Design and Implementation of a Modular Kalman Filter

Benjamin Rubinstein
bzr4@case.edu
William MacCormack
wjm102@case.edu

Abstract

We present an implementation of a modular Kalman filter, utilizing both linear and unscented Kalman filtering elements, optimized for robust localization in aerospace and robotics applications. The architecture utilizes modular predictor and corrector components, with flexibility for system-specific evolution and observation functions, ensuring adaptability. Two synthetic modalities were tested and residual values were found to be within acceptable ranges. Proposed enhancements include transitioning from Python to a compiled language for use on embedded systems and automatic noise covariance estimation to reduce the need for accurate a priori sensor characterization.

1 Introduction and Background

State estimation is critical to many control and analysis algorithms for applications in robotics, aerospace, and mechanical systems. Bayesian filters attempt to solve the problem of state estimation by propagating a probabilistic estimate of system state through an evolution model and correcting this estimate with a probabilistic model of sensor data through the application of Bayes theorem. The common approach of modeling the state and sensor estimates as gaussian distributions has led to a proliferation of Kalman filters[3]. These filters, which include the (linear) Kalman Filter, Extended Kalman Filter, and Unscented Kalman Filter, exploit the property a Gaussian distribution is fully defined by a mean and covariance.

The Linear Kalman Filter uses the linear nature of its system models to provide closed form update and correction equations for the mean and covariance of the evolution and observation model. The extended Kalman filter avoids the linear system requirement by employing the Jacobian of the system to linearize the evolution and observation models. The unscented filter opts to forego exact propagation of mean and covariance through approximate system models, and, instead, exactly propagates a weighted ensemble of 'sigma points' through evolution and observational models, which are used to estimate the true mean and covariance. Details for the implemented system can be found in section 2.

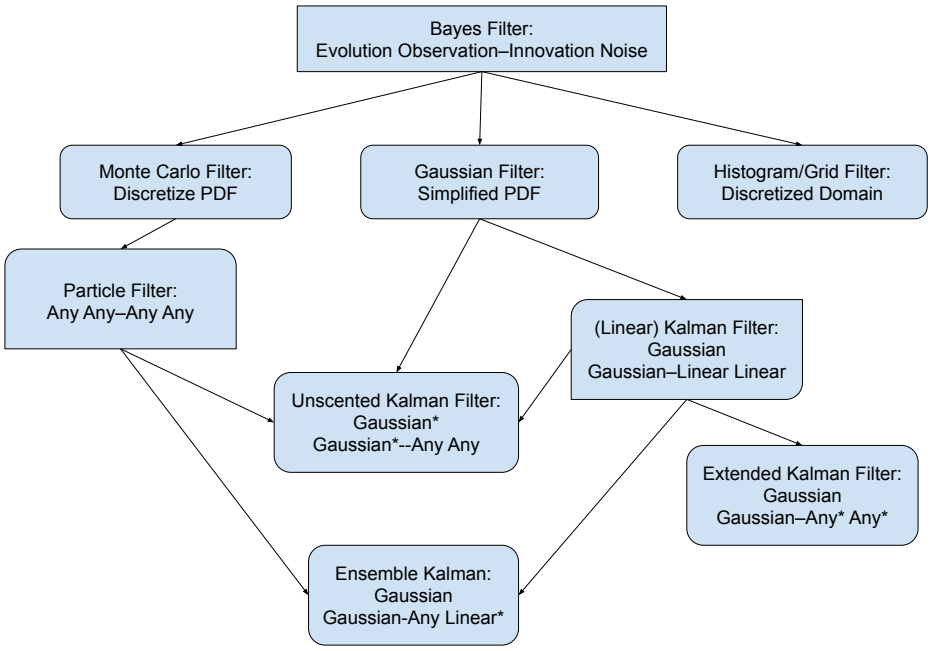

Figure 1: Taxonomy of the different types of Bayes Filters

2 Mathematical Modeling and Equations

All Bayesian filters require an observation model,

$$B_t = g(X_t) + U_t$$

where b_t represents the data received at time t and x_t is the state estimate at time t. g is the function that describes the expected sensor value given a system state. If that function is linear, we represent it as a matrix A.

Bayesian filters also require an evolution model,

$$X_{t+1} = f(X_t) + V_t$$

where f is the function that takes an observation at time t and predicts what the system will be at time $t+1$. If that function is linear, we represent it as a matrix F.

The errors $U_t \sim \mathcal{N}(\mu, \Sigma), V_t \sim \mathcal{N}(\mu, \Gamma)$ are the measurement (observation) and process (evolution) noise estimates.

Since the Kalman filter assumes that the error and process distributions are both gaussians, the entire probability can be defined by its mean $\bar{x}$ and covariance D. This gaussian is propagated forward in time through the evolution model to generate a gaussian a priori estimate of the system state and is corrected by gaussian observations into a gaussian a posteriori state estimate. Thus, the state estimate is evolved with time and corrected as new observational data is available.

2.1 Linear Kalman Filter Update

2.1.1 Linear Prediction

When Kalman filtering a system with a linear evolution model, the mean and covariance can be propagated exactly.

$$\hat{\bar{x}} = F\bar{x}, \quad \hat{D} = FDF^T + \Gamma$$

2.1.2 Linear Correction

The Linear Corrector adjusts the mean estimate to more closely match the observed data, proportional to the relative covariance of the prior and sensor models with the Kalman gain matrix, K.

$$\Delta = b - Ax, \quad K = DA^T(ADA^T + \Sigma)^{-1}$$

While the mean estimate correction is dependent upon the measurement, the covariance estimate correction is independent of the accuracy of the mean estimate.

$$\bar{x}' = \bar{x} + K\Delta, \quad D' = (I - KA)D$$

2.2 Extended Kalman Filter Update

2.2.1 Extended Prediction

Extended Kalman Prediction only extends linear Kalman Prediction by propagating the mean through the non-linear evolution function, and replacing the state-independent linear update transform with the state-dependent Jacobian of the non-linear function.

$$\hat{\bar{x}} = f(\bar{x}), \quad \hat{D} = FDF^T + \Gamma$$

2.2.2 Extended Correction

The Extended Corrector, likewise, replaces the mean observation estimate with the non-linear observational function, and the linear observational matrix is replaced with the Jacobian of the non-linear observation function.

$$\Delta = b - g(x), \quad K = DA^T(ADA^T + \Sigma)^{-1}$$

$$\bar{x}' = \bar{x} + K\Delta, \quad D' = (I - KA)D$$

2.3 Unscented Kalman Filter Update

2.3.1 Unscented Transform

The Unscented Kalman filter [6] uses the Unscented Transform, a weighted ensemble of "sigma points" to efficiently represent a probability distribution:

The distribution of the sigma points in the Scaled Set form are controlled by the parameters [1],

$$\alpha \in \mathbb{R}^+$$

$$\beta \in \mathbb{R}$$

$$\kappa \in \mathbb{R}^+$$

$$x^{(0)} = \bar{x}$$

and the ith and jth sigma points are sums of the mean and weightings of columns i or $j - n$ of the covariance:

$$x^{(i)} = \bar{x} + (\alpha\sqrt{\kappa D})_i \quad \forall i = 1, \ldots, n$$

$$x^{(j)} = \bar{x} - (\alpha\sqrt{\kappa D})_{j-n} \quad \forall j = n+1, \ldots, 2n$$

The sigma point at the mean estimate is weighted differently in mean and covariance estimates, while the remaining sigma points all have the same weight for both mean and covariance.

$$w_m^{(0)} = \frac{\alpha^2\kappa - n}{\alpha^2\kappa}, \quad w_c^{(0)} = w_m^{(0)} - \alpha^2 + \beta$$

$$w_m^{(i)} = w_c^{(i)} = \frac{1}{2\alpha^2\kappa} \quad \forall i = 1, \ldots, 2n$$

2.3.2 Unscented Prediction

Unscented prediction only requires the evaluation of the evolution function f, applying it to each sigma point and taking a weighted sum of the output to computed the mean and covariance:

$$\hat{x}_{(i)} = f(x^{(i)}),\ \hat{\bar{x}} = \sum_{i=0}^{2n} w_m^{(i)} \hat{x}^{(i)},$$

$$\hat{D} = \sum_{i=0}^{2n} w_c^{(i)} (x^{(i)} - \hat{\bar{x}})(x^{(i)} - \hat{\bar{x}})^T + \Gamma$$

2.3.3 Unscented Correction

The correction step first predicts the measurement from the state estimate using g and the sigma point ensemble:

$$z^{(i)} = g(x^{(i)}), \quad \bar{z} = \sum_{i=0}^{2n} w_m^{(i)} z^{(i)}$$

From the ensemble, two covariance matrices are found. First is the predicted covariance:

$$S = \sum_{i=0}^{2n} w_c^{(i)} (z^{(i)} - \bar{z})(z^{(i)} - \bar{z})^T + \Sigma$$

Second, is the cross covariance:

$$T = \sum_{i=0}^{2n} w_c^{(i)} (x^{(i)} - \bar{x})(z^{(i)} - \bar{z})^T$$

Then, the residual and Kalman gain are comupted:

$$\Delta = b - \bar{z}, \quad K = TS^{-1}$$

While the mean update is identical to the linear Kalman update, the covariance estimate update takes on a new form:

$$\bar{x}^{'} = \bar{x} + K\Delta, \quad D^{'} = D - KSK^T$$

3 Implementation

Our implementation of a modular Kalman filter [4] consists of two classes, a prediction provider and a correction provider, to evolve the state vector through time and apply a correction to the estimate based on received data, respectively. Abstract classes were created for the predictor and corrector, which provide the interface methods for prediction and correction.

For this project, two distinct filter implementations were created: the linear Kalman filter and the Unscented Kalman filter (N.B. an Extended Kalman filter can be implemented with this library by providing the non-linear function and the Jacobian to the linear filter implementation). Each filter's predictor and corrector implements methods to update mean and covariance. Because of this polymorphism, the user can call the same 'predict' or 'correct' function regardless of which Kalman update methodology the provider utilizes. This design choice enables interoperability of linear and Unscented filter providers across the prediction and correction steps. This interoperability allows for the use of computationally more efficient correctors for sensors that behave linearly, while Extended and Unscented correctors are available for non-linear sensors. A single predictor is used to evolve the system at each timestep. However, a distinct corrector is used for each atomic sensor interaction. This is critical to the success of the system, as it enables the filter to function optimally with whatever sensors provide new readings at each timestep. These, temporally sparse sensor updates can occur due to differing update rates across the array of different senors employed by the filter or through disconnection and malfunction of specific sensors.

By separating the prediction and correction steps, the filter can provide a state estimate more frequently than the fastest updating sensor, albeit with larger covariance than could be provided by a constantly corrected estimate. This multi-update, observation-split filter allows control and analysis systems built on top of the filter to be abstracted from the specifics of particular sensor noise and update rates, while still providing a full qualification of the estimate through the covariance.

4 Applications

The model was tested on two modalities. First, on a spring-mass oscillator. The filter tracked one degree of position, velocity and acceleration.

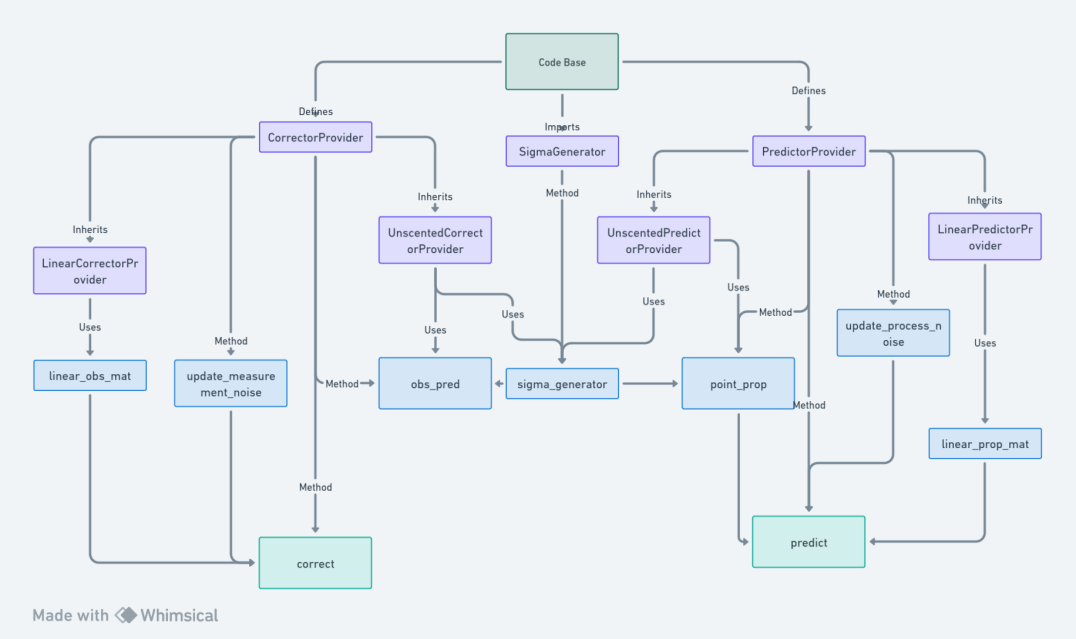

Figure 2: Structure of the Kalman Filter

Second, on a modeled rocket flight, where the state included three axes of position, velocity, acceleration and rotational velocity along with an orientation quaternion. Dropout rates of 10% through 99% were tested on the spring-mass system, and model predictions were only began to be seriously affected by dropout at around 80% dropout, see figure 7.

4.1 Spring Mass System

The first system, an undamped spring-mass oscillator, obeys the following dynamics:

Position (x):

$$x(t) = A\cos(\omega t)$$

Velocity (v):

$$v(t) = -A\omega\sin(\omega t)$$

Acceleration (a):

$$a(t) = -A\omega^2\cos(\omega t)$$

The natural frequency ω is defined by

$$\omega = \sqrt{\frac{k}{m}}$$

where k is the spring constant, m is the mass and A is the amplitude.

4.1.1 Linear State Estimation with Sensor Dropout

The linear Kalman Filter was capable of tracking the state of the oscillator without prior knowledge of the specific system dynamics. It employed the following evolution model:

Position (x):

$$x^{'} = x + v \cdot \Delta t + \frac{1}{2} a \cdot \Delta t^2$$

Velocity (v):

$$v^{'} = v + a \cdot \Delta t$$

Acceleration (a):

$$a^{'} = a$$

4.1.2 State and Parameter Estimation with the Unscented Kalman Filter

Additionally, an Unscented Kalman filter was employed to perform simultaneous state and parameter estimation on the spring mass oscillator system with the following evolution model:

Position (x):

$$x^{'} = x + v \cdot \Delta t + \frac{1}{2} \frac{k}{m} \cdot \Delta t^2$$

Velocity (v):

$$v^{'} = v + a \cdot \Delta t \quad | \quad a = -x \cdot \frac{k}{m}$$

Spring Mass ratio ($\frac{k}{m}$):

$$\frac{k}{m}^{'} = \frac{k}{m}$$

The model was provided only with position data, and successfully tracked and identified the spring mass ratio of the system.

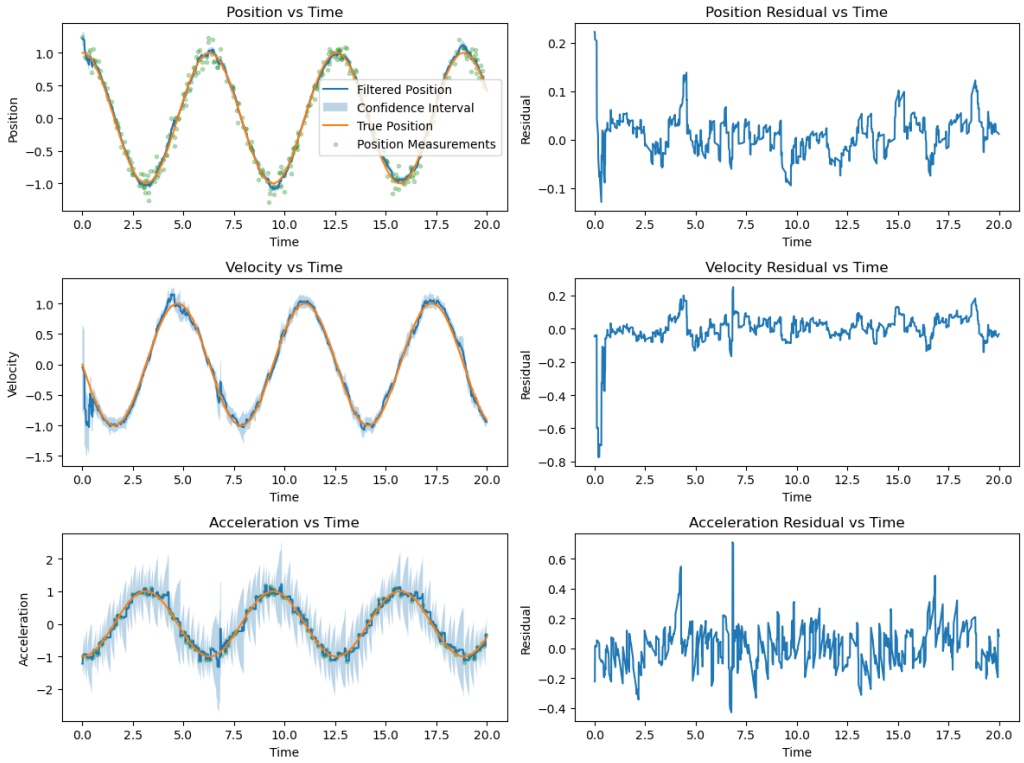

Figure 3: Oscillator with Linear filter and high dropout

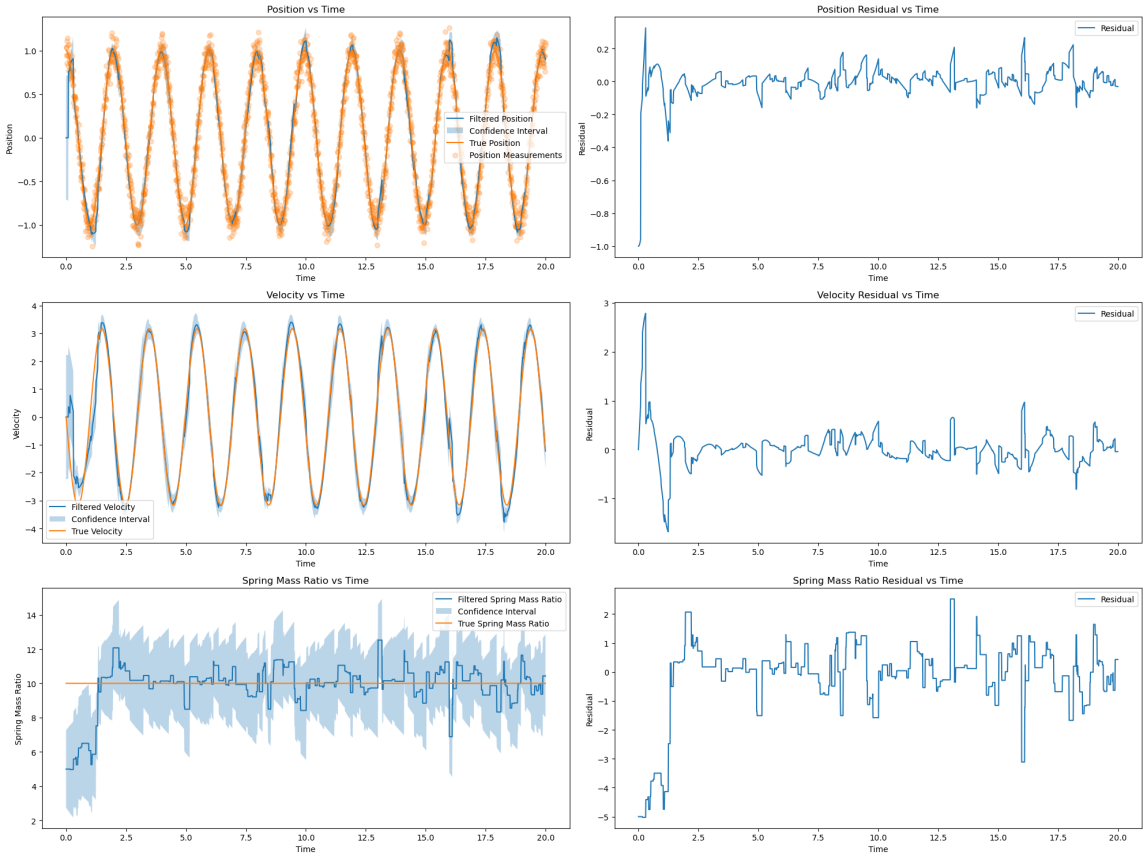

Figure 4: Oscillator with Unscented filter with 90% drop out.

4.2 Simplified Rocket Flight

An amateur rocket, similar to the one used by the Case Rocket Team, was modeled in flight. For the sake of this project, the following was assumed:

1. A constant coefficient of drag
2. a proportional relationship between impulse and mass lost
3. Point mass physics for the rocket.

4.2.1 Generating Data, Physics Simulation

First we calculated the forces

$$F_{\text{drag}} = -C_d \cdot A_s \cdot \rho(\vec{x}) \cdot \|\vec{v}\| \cdot \frac{\vec{v}}{\|\vec{v}\|}$$

Amateur rocket motors (the premade solid fuel used in model rockets) define thrust curves, a function that relates the thrust produced over the burn time of the motor. This is used to calculate the thrust produced, with the assumption that the rocket is pointed in the direction that it is headed due to aerodynamic stability.

$$\vec{F}_{\text{thrust}} = f_{\text{thrust}}(t) \cdot \frac{\vec{v}}{\|\vec{v}\|}$$

The total mass is calculated as a sum of the dry rocket mass and the motor mass. Since the motor burns over the course of flight, the motor looses fuel mass. We assume the motor looses fuel proportional to the rate at which thrust is produced, and derive the mass equations for the motor:

$$m_{motor}(t) = m_{motor}(0) \left(1 - \frac{\int_0^t f_{\text{thrust}}(s)\, ds}{\int_0^\infty f_{\text{thrust}}(t)\, dt} \right)$$

Then we can take the sum of the mass of the motor and the rocket (which is constant) and find that the force of gravity:

$$F_{\text{gravity}} = m_{\text{tot}} \cdot \vec{g}$$

The wind force was based on a perlin noise function scaled to the expected height. This, combined with the air density (calculated as an exponential decay relative to height: $\rho(z) = 1.225e^{-z/8500}$) generates the wind field.

$$F_{\text{wind}} = \|\vec{v}_{\text{relative}}\|^2 \cdot \rho(z) \cdot A_s \cdot \frac{\vec{v}_{\text{relative}}}{\|\vec{v}_{\text{relative}}\|}$$

We can then sum forces:

$$\vec{F}_{\text{total}} = F_{\text{drag}} + F_{\text{thrust}} + F_{\text{gravity}} + F_{\text{wind}}$$

Once we find the forces acting on our body, we update the state:

$$\vec{a}_{t+\Delta t} = \frac{\vec{F}_{\text{total}}}{m_{\text{tot}}}$$

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}_t \cdot \Delta t$$

$$\vec{x}_{t+\Delta t} = \vec{x}_t + \vec{v}_t \cdot \Delta t$$

The orientation quaternion is updated such that the rocket's orientation is in line with the velocity vector, then the change in the orientation $\dot{q}$ is computed by dividing the current orientation by the previous, which informs the new angular velocity.

$$\begin{bmatrix} 0 \\ \omega_{x,t+\Delta t} \\ \omega_{y,t+\Delta t} \\ \omega_{z,t+\Delta t} \end{bmatrix} = \frac{\dot{q}}{\Delta T}$$

4.2.2 Filter State Vector

The state vector of the rocket was constructed as follows:

1. $x \in \mathbb{R}^3$:= the position of the rocket in world space
2. $v \in \mathbb{R}^3$:= the velocity of the rocket in world space
3. $a \in \mathbb{R}^3$:= the acceleration of the rocket in world space
4. $q \in \mathbb{H}$:= the orientation of the rocket as a quaternion[5]
5. $\omega \in \mathbb{R}^3$:= the angular velocity of the rocket

4.2.3 Filter Evolution Model

This filter employs an Unscented Predictor Provider because the orientation quaternion portion of the state vector behaves non-linearly.

$$\vec{x}' = \vec{x} + \vec{v} \cdot \Delta t + \frac{1}{2}\vec{a} \cdot \Delta t^2$$

$$\vec{v}^{'} = \vec{v} + \vec{a} \cdot \Delta t$$
$$\vec{a}^{'} = \vec{a}$$

$$q^{'} = (1 + \frac{\Delta t}{2} \cdot \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}) \cdot q$$

$$\vec{\omega}^{'} = \vec{\omega}$$

4.2.4 Filter Sensor Overview

The rocket modality was equipped with four sensors: a GPS (3-axis position), a barometer (height), accelerometer (3-axis linear acceleration), gyroscope (3-axis rotational velocity), and magnetometer (3-axis orientation vector relative to the earth). For this test case, two polling rates were utilized:

- 2 Hz for the GPS, chosen because the GPS can poll at between 1Hz to 5Hz
- 200 Hz for the rest of the sensors, because the slowest sensor (barometer) polls at 200 Hz in high accuracy mode.

4.2.5 GPS

The GPS uses a linear observation matrix is defined as:

$$x_{\text{GPS}} = \begin{bmatrix} I_3 & O_{3\times 13} \end{bmatrix}$$

where I_3 is the 3×3 identity matrix and $O_{3\times 13}$ is a 3×13 zero matrix. This separates the three position observations from the rest.

4.2.6 Barometer

The barometric pressure is modeled using the following equations to model pressure and temperature:

$$T(h) = T_0 - 0.0065h$$

$$P(h) = P_0 \left(1 - \frac{0.0065h}{T_0}\right)^{\frac{gM}{R \cdot 0.0065}} \times \frac{1}{10^3}$$

where $T(h)$ is the temperature at height h, $P(h)$ is the pressure at height h, g is the acceleration due to gravity, M is the molar mass of Earth's air, R is the universal gas constant, T_0 is the standard temperature at sea level, and P_0 is the standard pressure at sea level.

4.2.7 Accelerometer

The acceleration corrector is non-linear, and has three different parts. First, the acceleration on the entire rocket is calculated.

$$a_{t+1} = a_t + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

Then, the acceleration is rotated to the reference frame of the rocket.

$$a_{\text{body}} = R(a_{t+1}, q)$$

where R represents the quaternion-based rotation of the vector.

Finally, the sensor acceleration is modeled, adding the apparent acceleration from the spin of the rocket.

$$a_{\text{sensor}} = a_{\text{body}} - \omega \times (\omega \times r_{\text{imu}}))$$

4.2.8 Gyroscope

The gyroscope corrector predicts the angular velocity in the body frame:

$$\omega_{t+1} = R(\omega_t, q)$$

4.2.9 Magnetometer

And the magnetometer predicts the magnetic field vector in the body frame:

$$\vec{\text{mag}} = R(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, q)$$

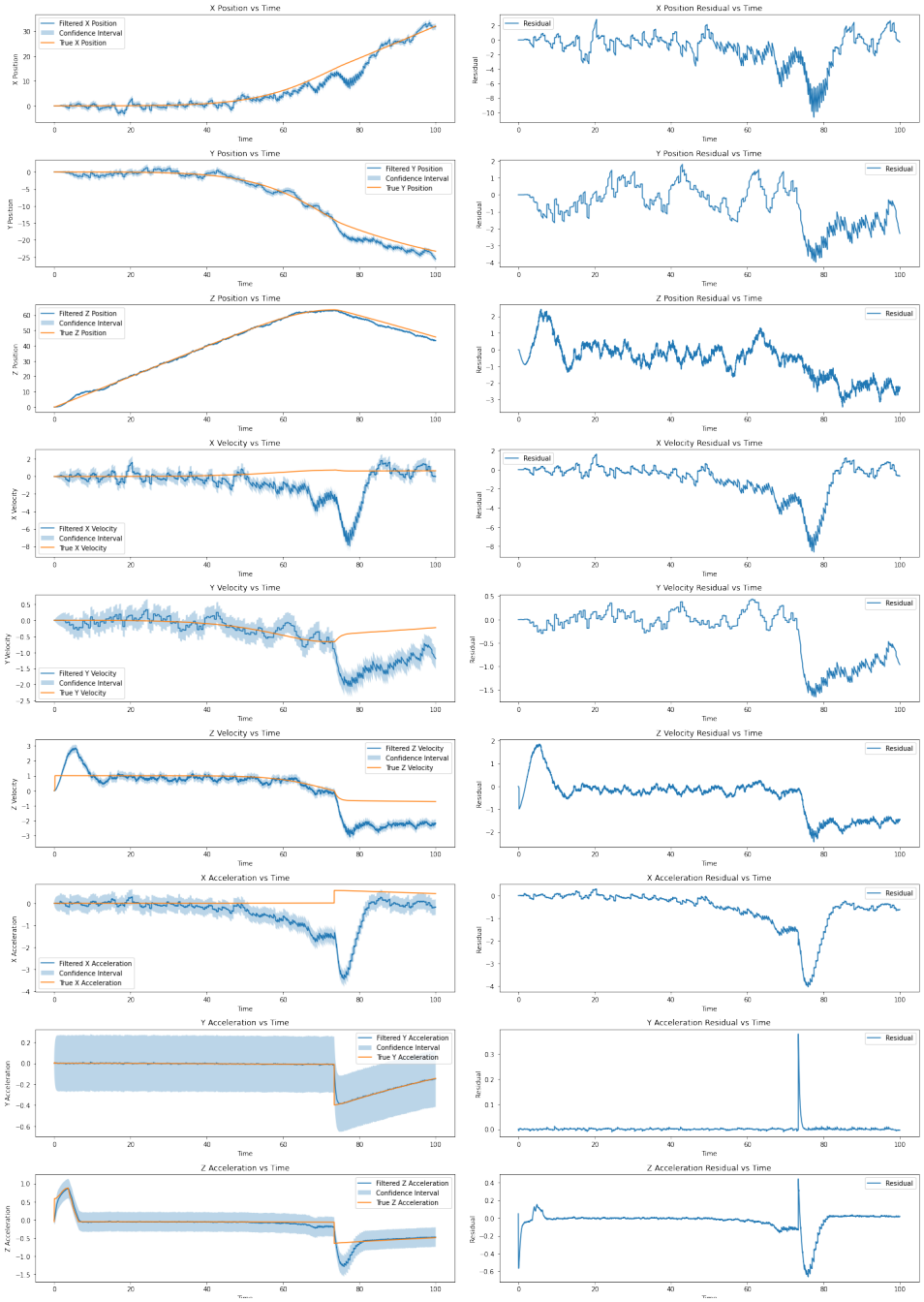

Figure 5: Rocket tested with a combination of filters, Position, Velocity and Acceleration

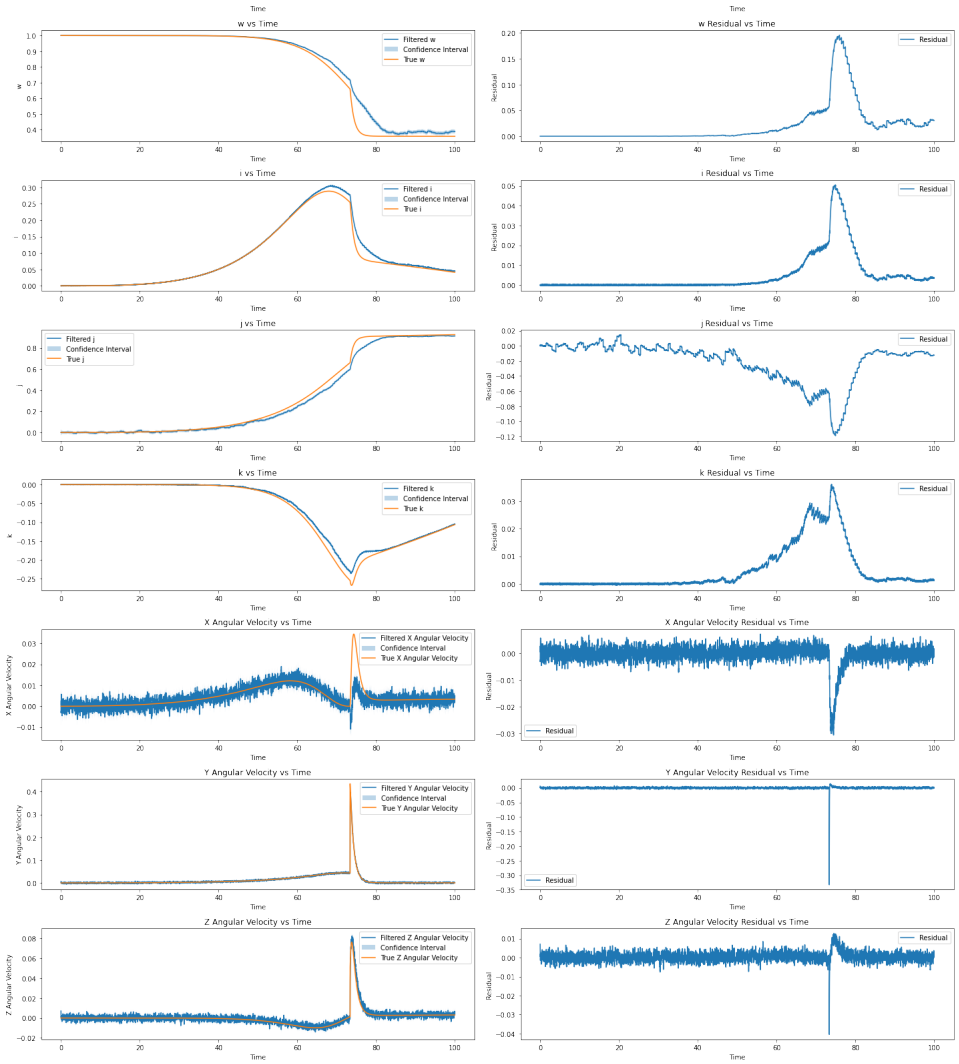

Figure 6: Rocket tested with a combination of filters, Orientation and Rotational Velocity

5 Results

The filters on both modalities were successful, maintaining residuals within acceptable tolerances during critical parts of the simulation. Namely, the oscillator was able to keep position residual within 2% of the amplitude, and the rocket was able to maintain position residual within 5% during ascent.

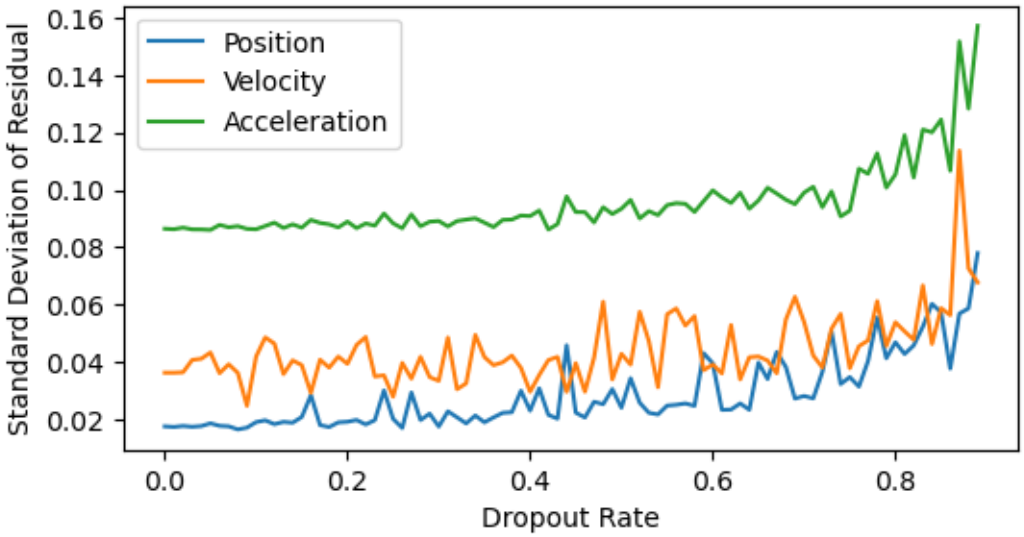

Figure 7: Dropout Rates and Scale of the Residual for The Linear Kalman tested on the oscillator

For the harmonic oscillator, average residual was plotted against the dropout rate. Dropout rate references the chance that a sensor measurement will be lost and subsequently neglected at a specific time step. For example, a drop out rate of 50% means that at each time step, there is a 50% chance that a correction will not occur, resulting in only a prediction for that time step. This is crucial in ensuring that the filter will remain robust even when sensors poll at less frequent rates.

6 Future Work

The filter applicability could be improved by transitioning from Python to a compiled language like C++ or Rust, accelerating performance and enabling implementation on embedded hardware for real-time, in situ localization. Additionally, integration efforts with new sensors and vehicles could be reduced by automatically approximating covariances at run time.

7 Conclusion

In this work we have implemented a modular Kalman filter, allowing for flexible construction of platforms with multiple, arbitrary sensors.

We have shown the filter's effectiveness on synthetic data which mimics the vehicles on which we aim to deploy this filter.

This work will enable us, as students on design teams, to more effectively localize our vehicles and apply corrective controls. With future work enabling us to run and tune this filter on hardware, this work clearly paves the way for tangible innovations in student design teams.

8 Code Availability

The code used for this project can be found at the following github repository:
https://github.com/Willmac16/fused-kalman-localization

9 Acknowledgements

We want to thank Dr. Daniela Calvetti for teaching the Bayesian Scientific Computing class, whose book[2] and work was instrumental in our ability to carry out this project.

References

[1] S Bitzer. *The UKF exposed: How it works, when it works and when it's better to sample*. Tech. rep. Zenodo, 2016. DOI: 10.5281/zenodo.44386.

[2] Daniela Calvetti and Erkki Somersalo. *Bayesian Scientific Computing*. 1st ed. Applied Mathematical Sciences. Springer Cham, 2023, pp. XVII, 286. ISBN: 978-3-031-23824-6. DOI: 10.1007/978-3-031-23824-6.

[3] KalmanFilter.net. *Kalman Filter*. Accessed: [December 1st, 2023]. 2023. URL: https://www.kalmanfilter.net/default.aspx.

[4] James Maley. *A Modular Approach to Kalman Filter Design and Analysis*. Technical Report. Pagination: 60 pages. DEVCOM Army Research Laboratory, Mar. 2021.

[5] F. Landis Markley. "Attitude Estimation or Quaternion Estimation?" In: *Flight Mechanics Symposium 2003*. NASA/CP-2003-212245. Greenbelt, MD, USA: National Aeronautics and Space Administration, 2003, N/A.

[6] Eric A. Wan and Rudolph van der Merwe. *The Unscented Kalman Filter for Nonlinear Estimation*. Tech. rep. Oregon Graduate Institute of Science & Technology.