271A Kalman Filter Project Report

Abstract

This report presents the implementation and analysis of a Kalman Filter designed for Calibration of an Accelerometer (IMU) Using GPS Measurements. We consider a vehicle accelerates in one dimension in an inertial frame, with the acceleration profile of

$$ a(t) = 10sin(\omega t) $$

where $\omega = 0.2 rad/s$.

We have an IMU that measures the acceleration at a sampling frequency of 200Hz. The IMU is modeled as

$$ a_c(t_j) = a(t_j) + b_a + w(t_j) $$

where b_a is the bias and has apriori statistics $b_a \sim N(0, 0.01 (m/s^2)^2)$, w is the IMU measurement noise and is modeled as additive white Gaussian noise with zero mean and variance $W=0.0004 (m/s^2)^2$

We have a GPS that measures position and velocity at a sampling frequency of 5Hz. The GPS is modeled as

$$ z_p(t_j) = p(t_j) + \eta _p(t_j)\ z_v(t_j) = v(t_j) + \eta _v(t_j) $$

where $\eta _p$ and $\eta _v$ are the GPS position and velocity measurement noise, modeled as independent additive white Gaussian noise

$$ \left[\begin{array}{ccc} \eta _p\ \eta *v \end{array}\right] \sim N(\left[\begin{array}{ccc} 0\ 0 \end{array}\right], V*{gps} = \left[\begin{array}{ccc} 1m^2 & 0\ 0 & (0.04m/s)^2 \end{array}\right]) $$

The initial position and velocity has apriori statistics $p_0 \sim N(0m, (10m)^2)$, $v_0 \sim N(100 m/s, (1m/s)^2)$.

Simulation results demonstrate the kalman filter's ability to converge the estimated state to the true state and converge the error variance matrix to the true state error variance matrix.

Theory

System Modeling

We want to have our state to be independent of the acceleration profile, such that this filter will perform the same way independent from the actual acceleration of the vehicle.

We model our IMU states as:

$$ a_c(t_k) = a(t_k) + b_a + w(t_k)\ v_c(t_{k+1}) = v_c(t_k) + a_c(t_k)\Delta t\ p_c(t_{t_k+1}) = p_c(t_k) + v_c(t_k)\Delta t + a_c(t_k)\frac{\Delta t^2}{2} $$

We model our true states in discreate time with Euler integration as:

$$ v_E(t_{k+1}) = v_E(t_k) + a(t_k)\Delta t\ p_e(t_{k+1}) = p_E(t_k) + v_E(t_k)\Delta t + a(t_k)\frac{\Delta t^2}{2} $$

- Note: It is assumed that the Euler integrated states are close enough to the actual true states from integrating the continuous acceleration profile.

By subtracting the pos and vel in the IMU model from the true states model, we get:

$$ \delta p_E = p_E - p_c\ \delta v_E = v_E - v_c\ \delta p_E(t_{k+1}) = \delta p_E(t_k) - \delta v_E(t_k)\Delta t - b_a\frac{\Delta t^2}{2} - w(t_k)\frac{\Delta t^2}{2}\ \delta v_E(t_{k+1}) = \delta v_E(t_k) - b_a\Delta t - w\Delta t $$

Let us have a new state $$ \delta x = [\delta p_E, \delta v_E, b_a]^T\ $$

We can model our system with the new state as

$$ \delta x(t_{k+1}) = \Phi \delta x(t_k) + \Gamma w(t_k)\ \Phi = \left[\begin{array}{ccc} 1 & \Delta t & -\frac{\Delta t^2}{2}\ 0 & 1 & -\Delta t\ 0 & 0 & 1 \end{array}\right], \Gamma = \left[\begin{array}{ccc} -\frac{\Delta t^2}{2}\ -\Delta t\ 0 \end{array}\right] $$

Our GPS measurement equation can also be updated as:

$$ \delta z = H\delta x + \eta\ \delta z = [\delta z_p, \delta z_v]^T,\ \eta = [\eta _p, \eta _v]^T, H = \left[\begin{array}{ccc} 1 & 0 & 0\ 0 & 1 & 0 \end{array}\right]\ \delta z_p = z_p - p_c\ \delta z_v = z_v - v_c $$

Kalman Filter Algorithm

From the System above that is independent from the acceleration profile, we can implement Kalman Filter Algorithm.

- Initialize true states based on apirori statistics of the states. Initial state prediction will be the expected values of the states from the apriori statistics. Initial apriori error variance will be the state variances from the apriori statistics.

- Begin loop:

  - Aposteriori estimation for t_k:

    - If measurement obtained: $$ P_k = M_k - M_kH^T(HM_kH^T + V_{gps})^{-1}HM_k\ K_k = P_kH^TV^{-1}\ r = \delta z - H\delta x_{predicted}\ (residual)\ \delta x_{estimate} = \delta x_{predicted} + K_k\ r $$

    - If measurement not obtained: $$ P_k = M_k\ \delta x_{estimate} = \delta x_{predicted} $$

  - Apriori prediction for t_{k+1}: $$ \delta x_{predicted} = \Phi \delta x_{estimate}\ M_k = \Phi P_k\Phi ^T + \Gamma W\Gamma^T $$

  - For simulation analysis, propagate true state for t_{k+1}: $$ \delta x_{k+1} = \Phi \delta x_k + \Gamma w_k $$

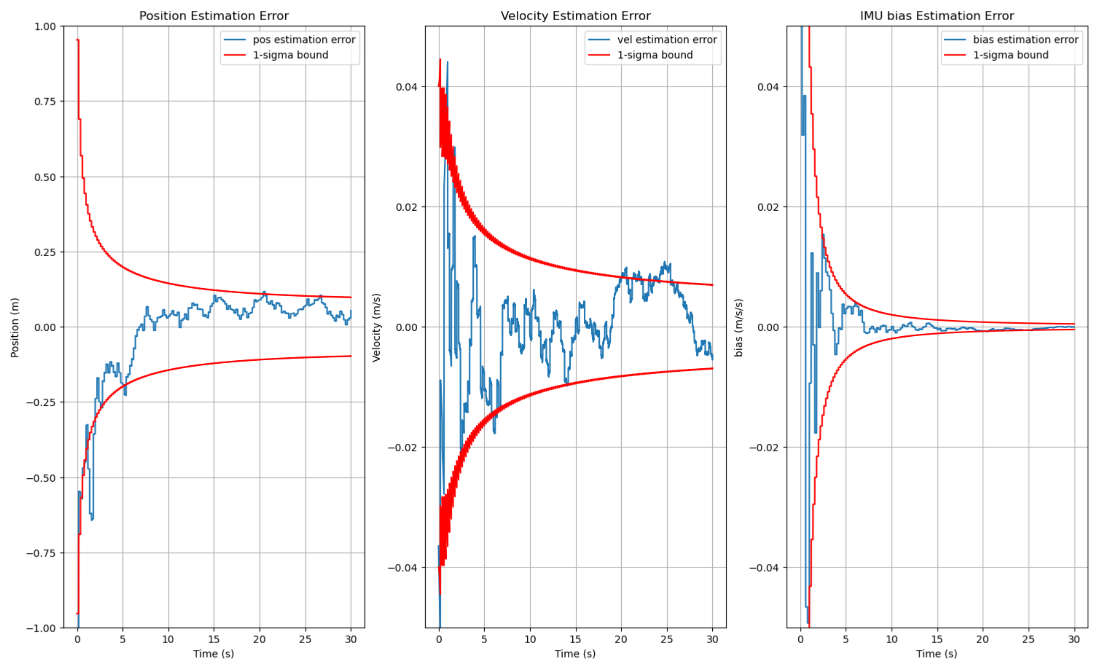
Results

- Note: all plots have units in SI. For example, position will have units of m, position variance will have units of m^2, velocity will have units of m/s, velocity variance will have (m/s)^2, covariance of position and velocity will have (m^2/s), etc.

One Realization Results

Shown below is the state estimation error and the error variance matrix error for one realization. We want our estimation error to be small to show that the kalman filter gives accurate estimation of the states.

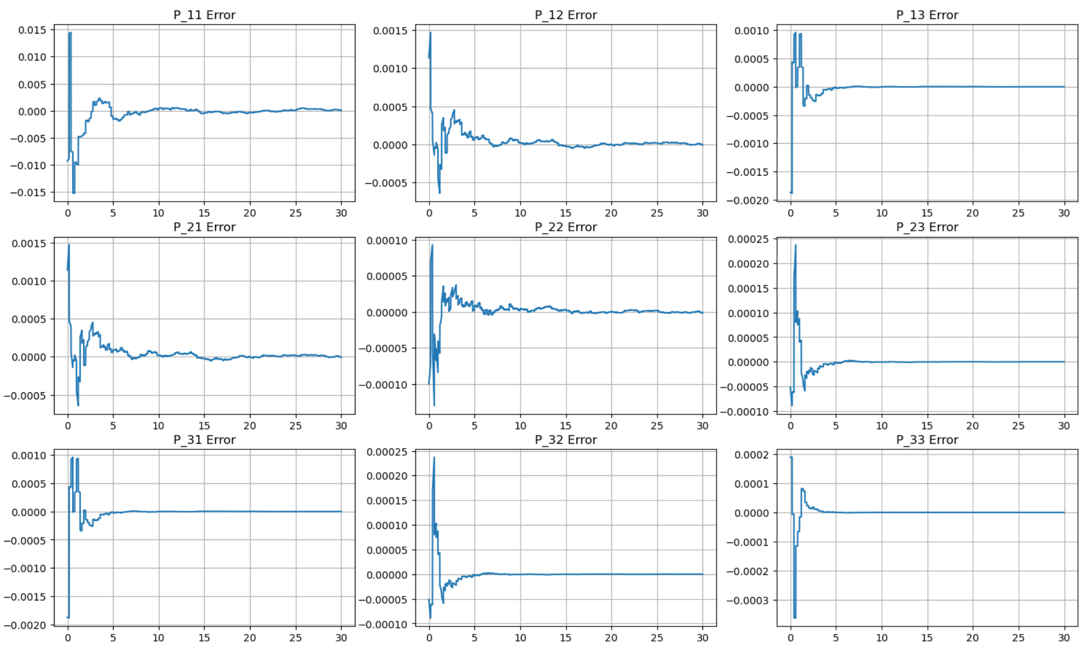
State estimation error is calculated as $$ e^l(t_k) = \delta x(t_k) - \delta x_{estimate}(t_k) $$



State Estimation Error Plot for 1 Realization

P error is calculated as $$ P_{error}(t_k) = P_{ave}(t_k) - P(t_k) $$

where P_{ave} is calculated from averaging over the outer product of the difference between e^l and $E[e^l]$ over the ensemble of realizations. $$ e^{ave}(t_k) = E[e^l(t_k)] = mean(e^l(t_k))\ (\sim 0)\ \tilde e_k = e^l_k - e^{ave} ✓ = *\tilde x)\ P*{ave}(t_k) = E[\tilde e_k^\ \tilde e_k^T] = mean(\tilde e_k\ \tilde e_k^T) $$

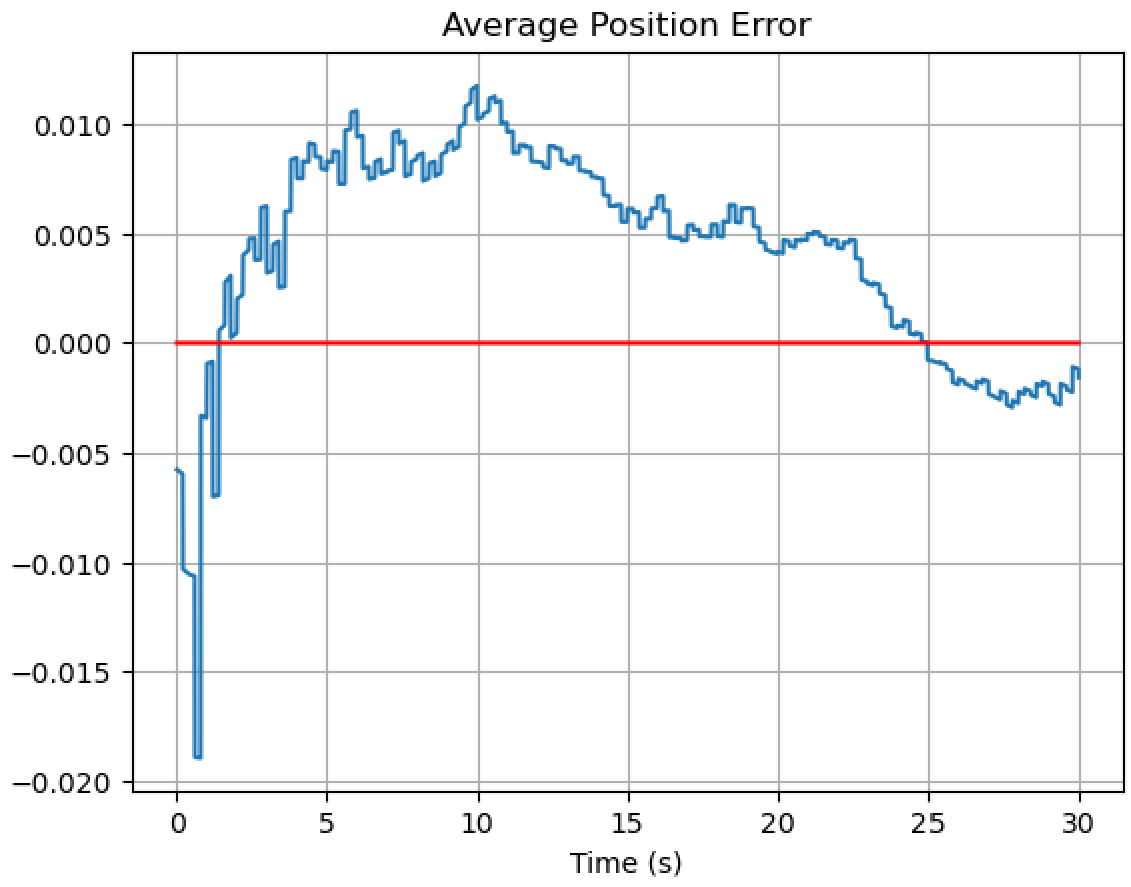


P Error Plot for 1 Realization

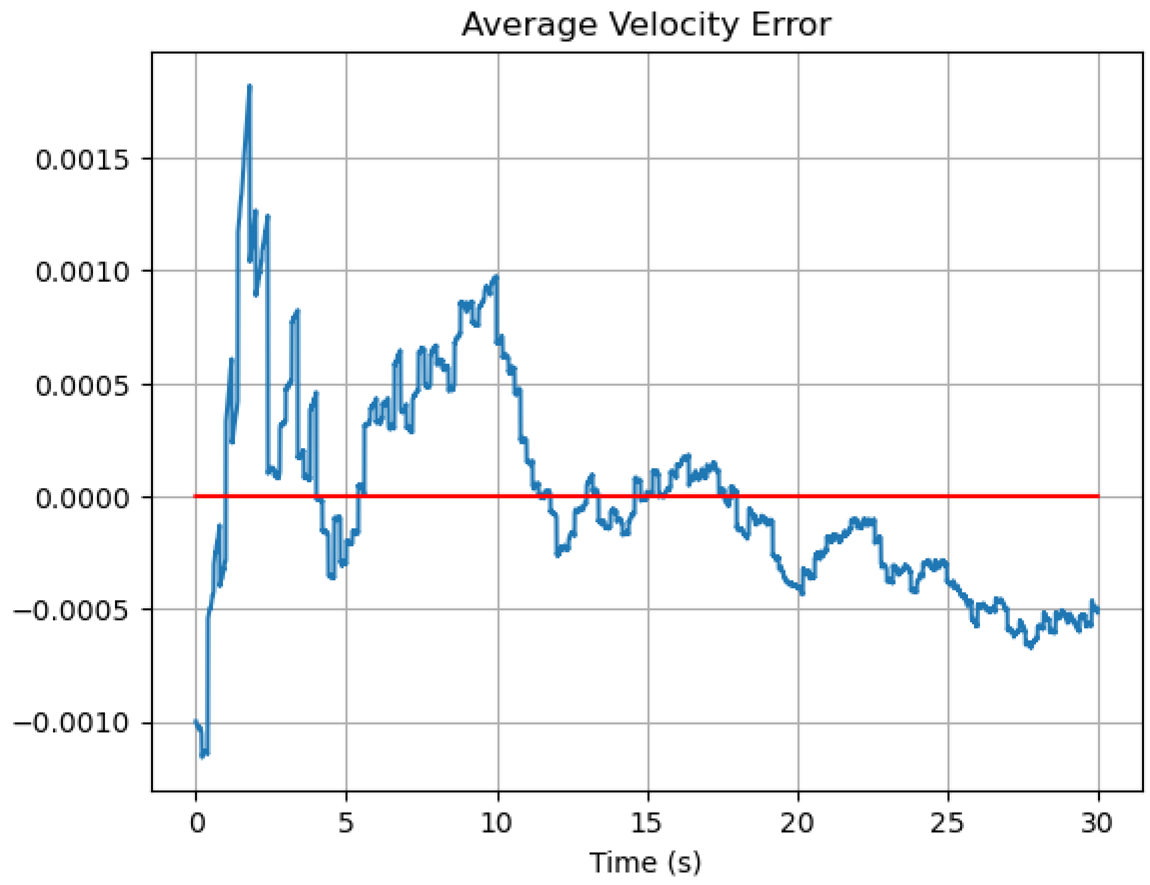
For this realization, the kalman filter is able to keep the states within the 1-sigma bound most of the time, showing its ability to converge the estimated states to the true states. The error covariance matrix is also really close to the true error covariance matrix calculated from averaging over 1000 realizations, showing that the kalman filter is programmed correctly.

Ensamble Realizations Estimation Error Results

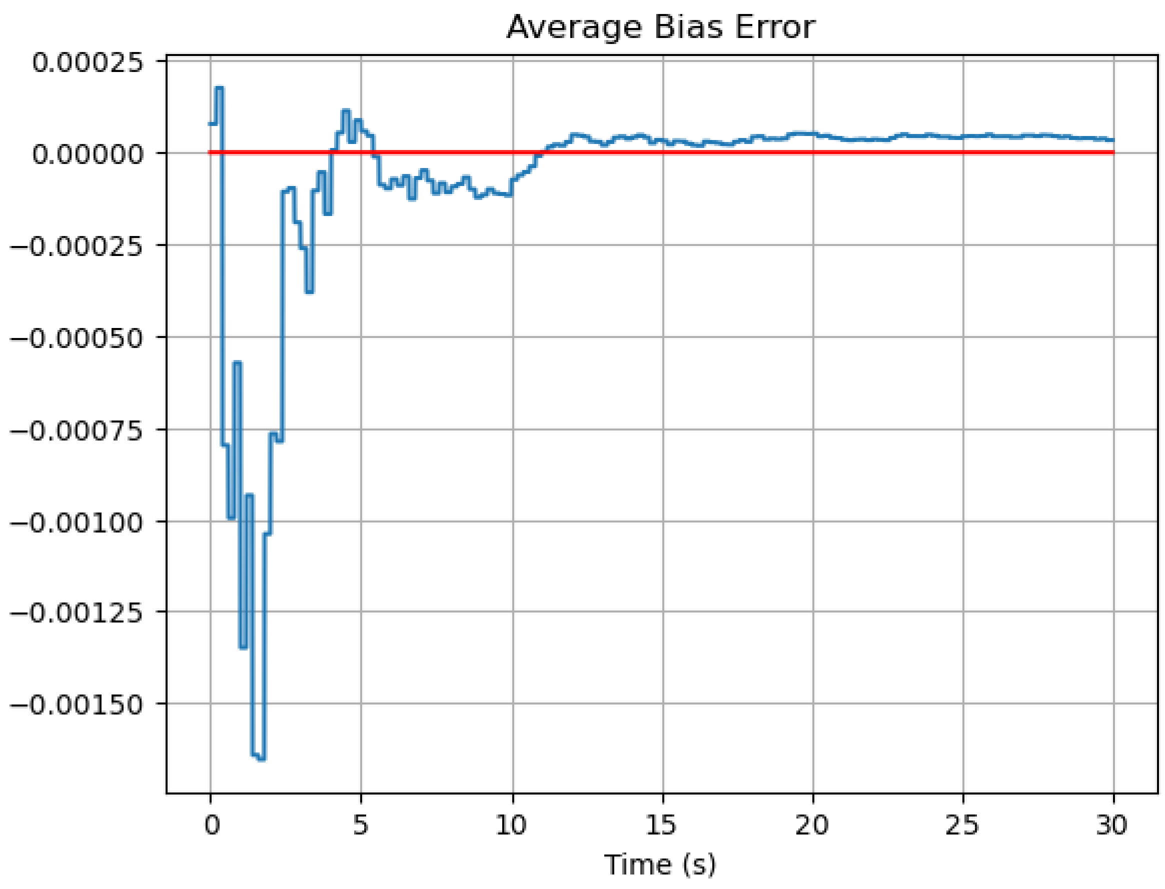
We plot the average state estimation error e^{ave} over 1000 realizations. They should be very close to 0 to show that there's no systematic error in the estimation process.



Average Position Estimation Error Plot for 1000 Realization



Average Velocity Estimation Error Plot for 1000 Realization

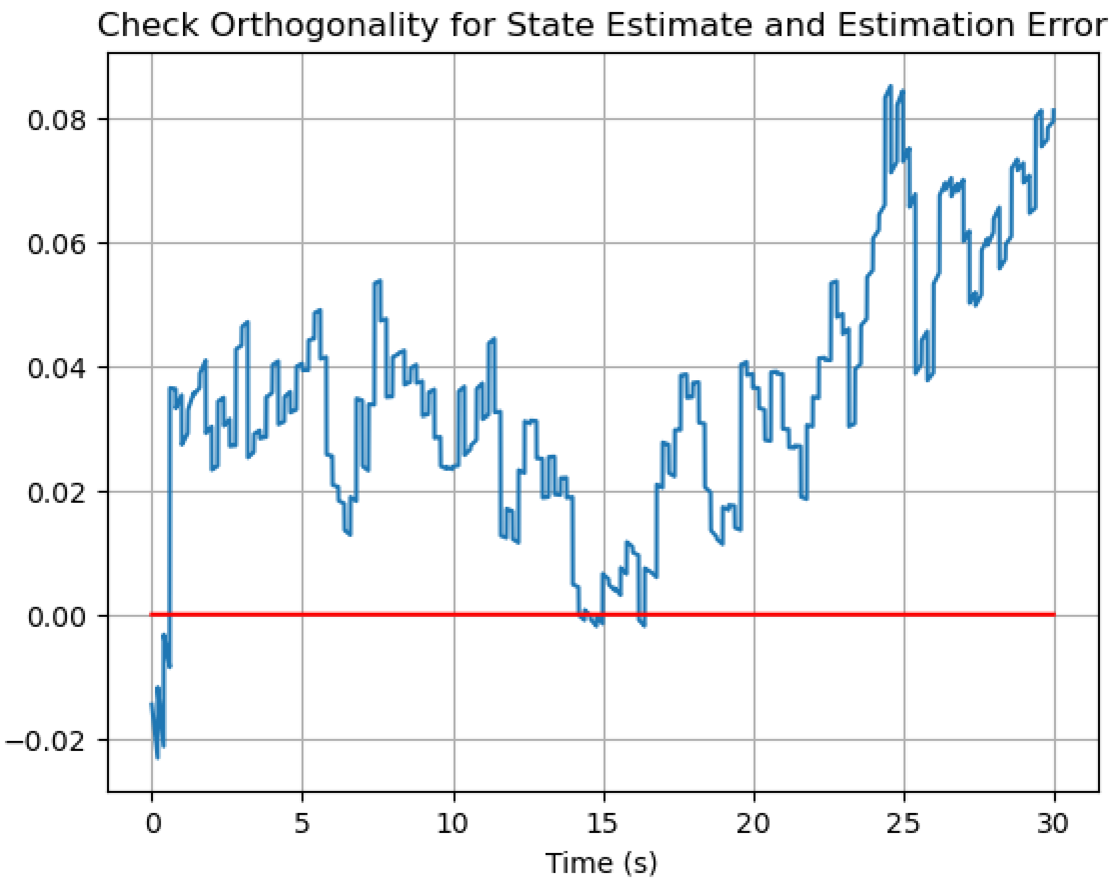


Average IMU Bias Estimation Error Plot for 1000 Realization

The above plots show that the average estimation error is reasonably small and close to 0. The average position error are relatively larger but could be explained by the large position error variance predicted by the kalman filter.

Orthogonality Check

We want to check that on average over the ensamble of realizations, $\tilde e$ and $\hat x\ (E[x])$ are orthogonal to each other. The orthogonality between estimation error and state estimate will show how optimal the filter is. We take the dot product of the 2 vectors and expect them to be close to 0 if orthogonal. $$ E[\tilde e_k^T\ \hat x_k] \sim 0\ \forall k\ mean(\tilde e_k^T\ \hat x_k) \sim 0\ \forall k $$

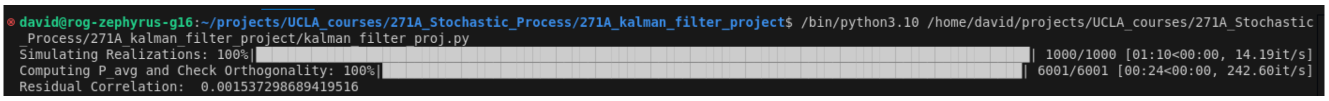


Check $\tilde e$ and $\hat x$ Orthogonality. Average over 1000 Realizations

From the plot above, we can see that the average values are relatively small and close to 0, indicating some optimality in the filter performance.

Check Residual Independence

We want to check that the residuals at different time are independent from each other because otherwise the process is not Markov. To show independence, the average of the dot product of two randomly picked residuals at different times should be close to 0. $$ r^l_k = \delta z_k - H\delta x_{predicted, k}\ mean(r_i^{lT}\ r_j^l) \sim 0\ \forall i \ne j $$



```
david@rog-zephyrus-g16:~/projects/UCLA_courses/271A_Stochastic_Process/271A_kalman_filter_project$ /bin/python3.10 /home/david/projects/UCLA_courses/271A_Stochastic
_Process/271A_kalman_filter_project/kalman_filter_proj.py
Simulating Realizations: 100%|██████████████████████████████████████████████████████████| 1000/1000 [01:10<00:00, 14.19it/s]
Computing P_avg and Check Orthogonality: 100%|██████████████████████████████████████████████| 6001/6001 [00:24<00:00, 242.60it/s]
Residual Correlation:  0.001537298689419516
```

Average Residual Correlation for 1000 Realization

The above result is calculated from randomly choosing two time index and compute the average over 1000 realizations the dot product of the residuals at those two times. We can see that the residual correlation is a small number and we can safely assume that the noise generated is Markov.

Conclusion

The implemented Kalman Filter effectively estimated position, velocity, and bias for a single-axis motion system. Simulation results verified the filter's accuracy and theoretical consistency. Future work may extend this implementation to multi-dimensional motion and possibly with other sensors for localization of the vehicle.