MAE 271A Project

1 Abstract

The goal of this project was to calibrate a vehicle's accelerometer using GPS measurements. A Kalman Filter was used to determine minimum variance estimates of the system. These estimates were compared to their corresponding true values over a span of 30 seconds. The one sigma bounds from the error variance were also plotted to determine if the estimates were relatively bounded by them. A Monte Carlo simulation was run for 10,000 realizations in order to check the theoretical properties of the system. These checks confirmed the Kalman Filter algorithm was functional and the model used was approximately correct.

2 Introduction

This project considers the calibration of a vehicle's accelerometer based on GPS measurements. The vehicle accelerates in one dimension in an inertial frame. The acceleration is assumed to be a harmonic of the form

$$a(t) = 10sin(\omega t) \quad \text{meters/sec}^2 \tag{1}$$

where $\omega = 0.1$rad/sec. The acceleration is measured by an accelerometer with a sample rate of 200 Hz at sample times t_j. The accelerometer is modelled with additive white Gaussian noise w with zero mean and variance V = .0004(meters/sec^2)^2. The accelerometer has a bias b_a with *a priori* statistics $b_a \sim$ N(0, .01(meters/sec^2)^2. The accelerometer a_c is modelled as:

$$a_c(t_j) = a(t_j) + b_a + w(t_j) \tag{2}$$

A GPS receiver is used to measure position and velocity in an inertial space. The measurements which are available at a 5 Hz rate (synchronized with the accelerometer) are

$$z_{1i} = x_i + \eta_{1i}$$

$$z_{2i} = v_i + \eta_{2i}$$

where x_i is the position and v_i is the velocity. Their *a priori* statistics are $x_0 \sim N(0 \text{ meters}, (10 \text{ meters})^2)$ and $v_0 \sim N(100 \text{ m/s}, (1 \text{ m/s})^2)$. The additive measurement noises are assumed to be white noise sequences and independent of each other with statistics

$$\begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \text{ meter}^2 & 0 \\ 0 & (4\text{cm/sec})^2 \end{bmatrix} \right)$$

A stochastic discrete time system, independent of the acceleration problem, was determined using the difference between the itegration of the accelerometer output and the GPS measurements.

The goal is to derive a Kalman Filter to determine minimum variance estimates and simulate the stochastic system over a 30 second interval.

By implementing the filter, the estimates of position, velocity, and accelerometer bias can be found as well as the filter error variance for one realization. A Monte Carlo simulation should be used to show that over an ensemble of realizations, the simulated error variance is close to the filter error variance. Finally, theoretical orthogonality properties should be proven in order to validate the filter.

3 Theory and Algorithm

3.1 Truth Model

Based on the assumption that the true acceleration acts as a harmonic given by Equation (1), the true velocity and position can be determined by integration.

$$\begin{aligned} v(t) &= v(0) + \frac{10}{\omega} - \frac{10}{\omega}\cos(\omega t) \\ p(t) &= p(0) + (v(0) + \frac{10}{\omega})t - \frac{10}{\omega^2}\sin(\omega t) \end{aligned} \tag{3}$$

where $v_0 \sim N(\bar{v}_0, M_0^v)$ and $p_0 \sim N(\bar{p}_0, M_0^p)$ with statistics given in Section 2.

3.2 Accelerometer Model

Using the given accelerometer model in Equation (2), the velocity and position can be calculated by a Euler formula where $\Delta t = 0.005$ seconds, the time step in between accelerometer measurements.

$$\begin{aligned} v_c(t_{j+1}) &= v_c(t_j) + a_c(t_j)\Delta t \\ p_c(t_{j+1}) &= p_c(t_j) + v_c(t_j)\Delta t + a_c(t_j)\frac{\Delta t^2}{2} \end{aligned} \tag{4}$$

with initial conditions $v_c(0) = \bar{v}_0,\ p_c(0) = \bar{p}_0 = 0$

3.3 Derivation of Dynamic Model

The Kalman filter should be independent of the true acceleration. In order to manipulate equations to remove true acceleration dependence, it is assumed that the true acceleration is integrated by the Euler equation rather than direct integration like in Section 3.1.

$$\begin{aligned} v_E(t_{j+1}) &= v_E(t_j) + a(t_j)\Delta t \\ p_E(t_{j+1}) &= p_E(t_j) + v_E(t_j)\Delta t + a(t_j)\frac{\Delta t^2}{2} \end{aligned} \tag{5}$$

To remove the $a(t_j)$ dependency, Equation (4) is subtracted from Equation (5). The dynamic equations can then be arranged in the following form to implement a Kalman Filter:

$$x_{k+1} = \Phi x_k + \Gamma w_k$$

$$\begin{bmatrix} \delta p_E(t_{j+1}) \\ \delta v_E(t_{j+1}) \\ b(t_{j+1}) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & -\frac{\Delta t^2}{2} \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta p_E(t_j) \\ \delta v_E(t_j) \\ b(t_j) \end{bmatrix} - \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \\ 0 \end{bmatrix} w(t_j) \tag{6}$$

where $\delta p_E(t_j) = p_E(t_j) - p_C(t_j)$ and $\delta v_E(t_j) = v_E(t_j) - v_C(t_j)$ and with initial states as follows:

$$\begin{aligned} \delta p(t_0) &= p_E(t_0) - p_c(t_0) \sim N(0, M_0^p) \\ \delta v(t_0) &= v_E(t_0) - v_c(t_0) \sim N(0, M_0^v) \\ b(0) &= b_a \sim N(0, M_0^b) \\ E[w(t_j)] &= 0, E[w(t_j)w(t_l)^T] = W\delta_{j,l} \end{aligned}$$

3.4 Measurement Equations

The GPS measurements are corrupted values of the actual position and velocity. GPS measurements are recorded a 5Hz, in other words they occur for every 40 accelerometer measurements.

$$z^p(t_i) = p(t_i) + \eta^p(t_i)$$
$$z^v(t_i) = v(t_i) + \eta^v(t_i)$$

The measurement equations are then manipulated to match the state variables of the dynamic model by subtracting by the position and velocity derived by the accelerometer measurements.

$$z_k = Hx_k + v_k$$
$$\begin{bmatrix} \delta z^p(t_i) \\ \delta z^v(t_i) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta p(t_i) \\ \delta v(t_i) \end{bmatrix} + \begin{bmatrix} \eta^p(t_i) \\ \eta^v(t_i) \end{bmatrix} \tag{7}$$

where $\delta p(t_i) = p(t_i) - p_c(t_i)$, and $\delta v(t_i) = v(t_i) - v_c(t_i)$

3.5 Kalman Filter

In order to use the measurement equations along with those of the dynamic model, the assumption is made that $\delta p(t_i) = \delta p_E(t_i)$ and $\delta v(t_i) = \delta v_E(t_i)$. The approximate *posteriori* conditional mean and the conditional *posteriori* error variance can be computed with a Kalman filter algorithm defined in Section 3.5.1 where

$$\hat{\delta x}(t_i) = \begin{bmatrix} \hat{\delta p}(t_i) \\ \hat{\delta v}(t_i) \\ \hat{b}(t_i) \end{bmatrix} = \begin{bmatrix} \hat{p}(t_i) - p_c(t_i) \\ \hat{v}(t_i) - v_c(t_i) \\ \hat{b}(t_i) \end{bmatrix}$$

3.5.1 Filter Algorithm

The Kalman Filter is implemented in a two step process: processing a measurement (update process), and propagating through the dynamics. The conditional mean, $\hat{x_k} = E[x_k|z_k]$ is based on the most recent data and is considered the "updated" estimate. The *a priori* estimate or "propagated" estimate, $\bar{x} = E[x_k|z_{k-1}]$ describes how the state evolves between measurements.

a priori propagation:

$$\bar{x}_{k+1} = \Phi \hat{x}_k \tag{8}$$

$$M_{k+1} = \Phi P_k \Phi^T + \Gamma W \Gamma^T \tag{9}$$

posteriori update:

$$\hat{x}_k = \bar{x}_k + K_k(z_k - H\bar{x}_k) \tag{10}$$

$$P_k = (M_k^{-1} + H^T V^{-1} H)^{-1} \tag{11}$$

$$K_k = P_k H^T V^{-1}$$

residuals:

$$r_k = z_k - H\bar{x}_k \tag{12}$$

where V is the covariance matrix of the additive measurement noises, W is the the variance of the additive white noise of the accelerometer. Their statistics are defined in Section 2:

$$V = \begin{bmatrix} 1 \text{ m}^2 & 0 \\ 0 & (0.04 \text{ m/s})^2 \end{bmatrix}, \quad W = 0.0004 \text{ (m/s}^2)^2$$

Because the GPS measurement is sampled at a slower rate than the accelerometer readings, the *a priori* measurement must be propagated between measurements. Propagation equations (8) and (9) then become the following for times without a GPS measurement:

$$\begin{aligned} \bar{x}_{k+1} &= \Phi \bar{x}_k \\ M_{k+1} &= \Phi M_k \Phi^T + \Gamma W \Gamma^T \end{aligned}$$

3.6 Orthogonality Properties

One validation technique for the implemented Kalman Filter is proving the property that the error of the estimate is uncorrelated to the estimate itself.

$$E[e_k \hat{x}_k^T] = E[(x_k - \hat{x}_k)\hat{x}_k^T] = 0 \tag{13}$$

In addition, another validation to be performed is to prove the residuals or "innovations" should be an independent sequence.

$$E[r_k r_j^T] = 0 \quad \forall j < k \tag{14}$$

3.6.1 Validation Algorithm

To validate the Kalman filter, a Monte Carlo simulation was used to compute an ensemble of realizations of the state and state estimates. Over the ensemble of realizations, the actual error and error variance can be calculated by averaging the errors of each realization. The orthogonality properties can also be simulated over the ensemble.

It is expected that the ensemble average of the actual error for realization l, $e^l(t_i)$, is approximately zero ($e_{ave}(t_i) \approx 0$) for all $t_i \in [0, 30]$).

$$e^{ave}(t_i) = \frac{1}{N_{ave}} \sum_{l=1}^{N_{ave}} e^l(t_i) \tag{15}$$

The ensemble average produces the actual error variance P^{ave}. The matrix $P^{ave}(t_i)$ should be close to $P(t_i)$ computed in the Kalman filter algorithm.

$$P^{ave}(t_i) = \frac{1}{N_{ave} - 1} \sum_{l=1}^{N_{ave}} [e^l(t_i) - e^{ave}(t_i)][e^l(t_i) - e^{ave}(t_i)]^T \tag{16}$$

The estimation error should be uncorrelated to the actual estimates. Thus, using the Monte Carlo simulation, the orthogonality property can be proven with Equation (17).

orthogonality of the error in estimates with the estimate check:

$$\frac{1}{N_{ave}} \sum_{l=1}^{N_{ave}} [e^l(t_i) - e^{ave}(t_i)]\hat{x}(t_i)^T \approx 0 \quad \forall\ t_i \tag{17}$$

The residuals should be an independent sequence and this orthogonality property can be proven with Equation (18).

independence of residuals check:

$$\frac{1}{N_{ave}} \sum_{l=1}^{N_{ave}} r^l(t_i) r^l(t_m)^T \approx 0 \quad \forall\ t_m < t_i \tag{18}$$

4 Results and Performance

4.1 State Estimates

The estimated position, velocity and bias were calculated using the Kalman filter algorithm (Section 3.5.1). The estimation errors, $x(t_i) - \hat{x}(t_i)$, are plotted in Figures 1, 2, and 3. The one sigma bounds are plotted against the error in order to visualize the decreasing filter error variance as time increases.

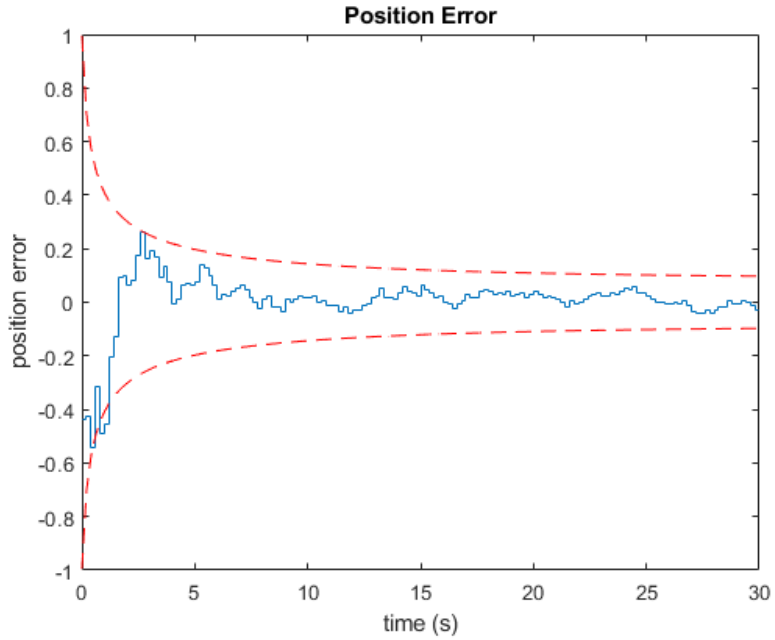


Figure 1: Position Error, $p(t_i) - \hat{p}(t_i)$

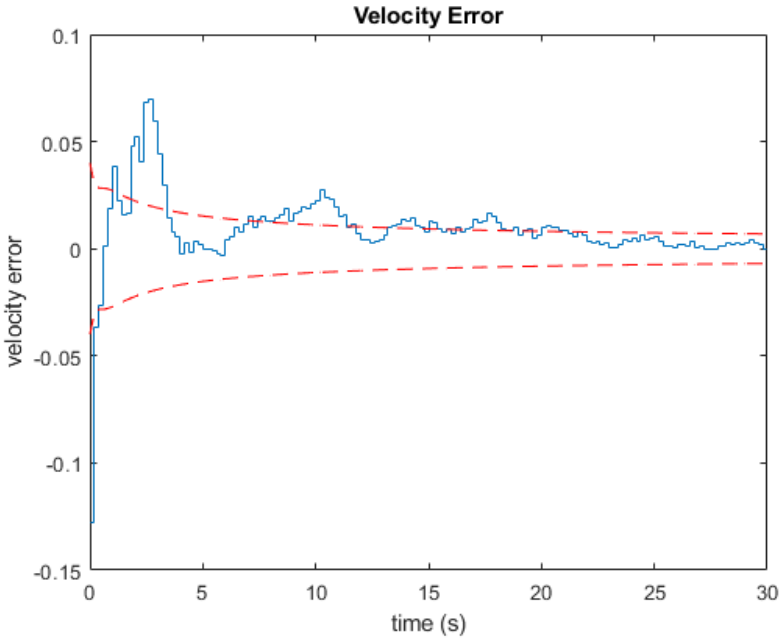
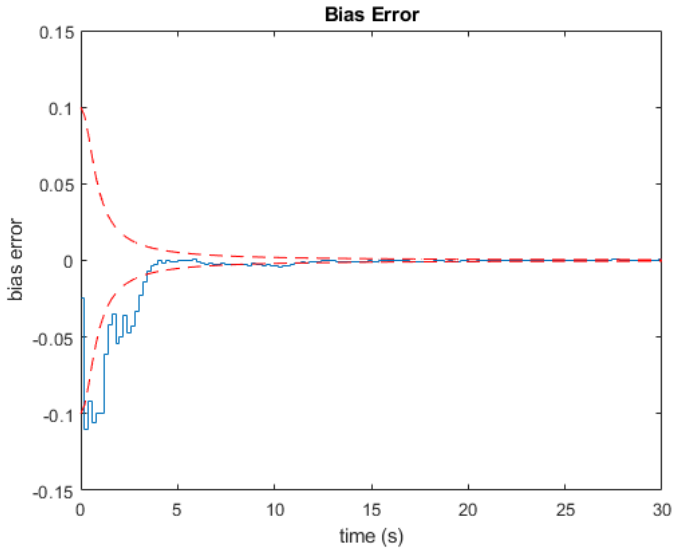
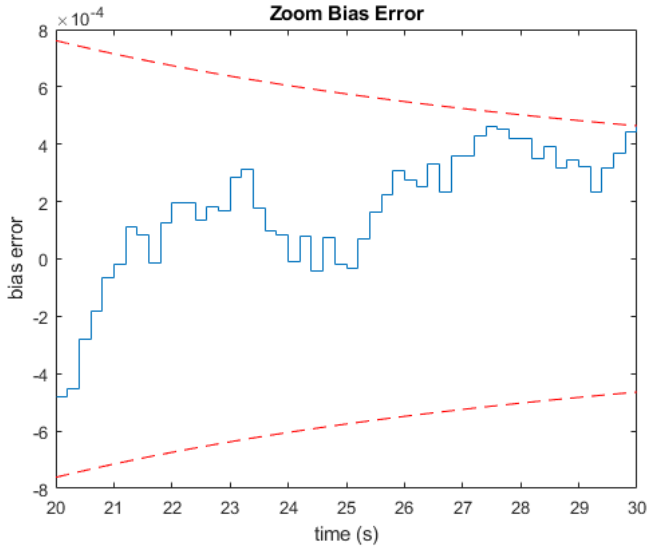


Figure 2: Velocity Error, $v(t_i) - \hat{v}(t_i)$

The bias error becomes very small as time increases. A zoomed in figure for the last 10 seconds of the bias error is provided in order to closely inspect values close to zero (See Figure 3b).



(a) Full time simulation



(b) Last 10 seconds

Figure 3: Bias Error, $b(t_i) - \hat{b}(t_i)$

In addition, the position, velocity, and bias estimates were plotted with their corresponding true values (See Figures 4, 5, 6). Section 3.1 provides the truth model equations for $p(t_i)$ and $v(t_i)$. As expected, the estimates track the true value over time. The estimate for the accelerometer's bias becomes more accurate as more measurements are taken from the GPS and the Kalman filter algorithm is updated.

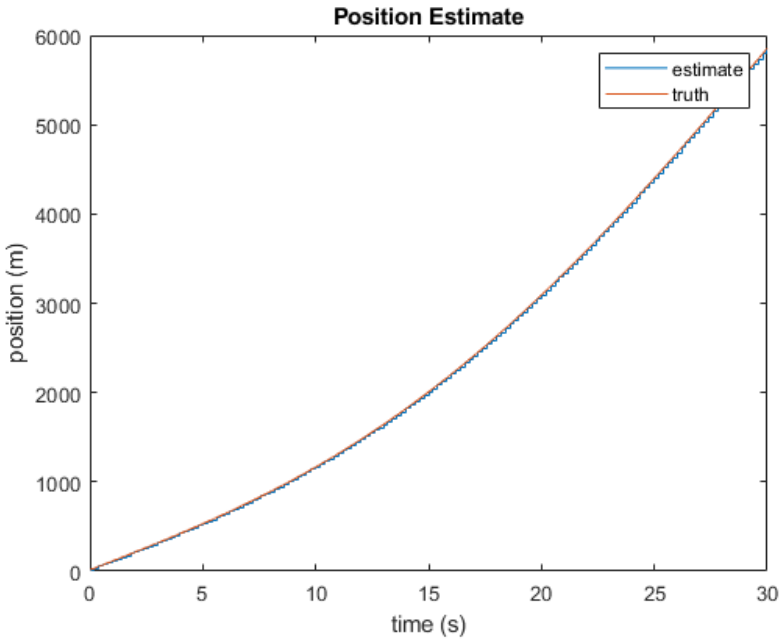


Figure 4: Position Estimate, $\hat{p}(t_i)$

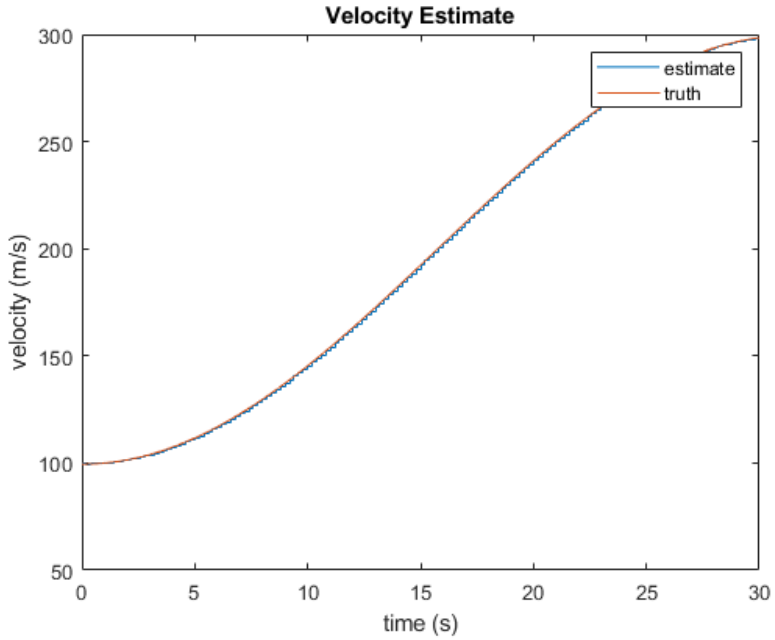


Figure 5: Velocity Estimate, $\hat{v}(t_i)$

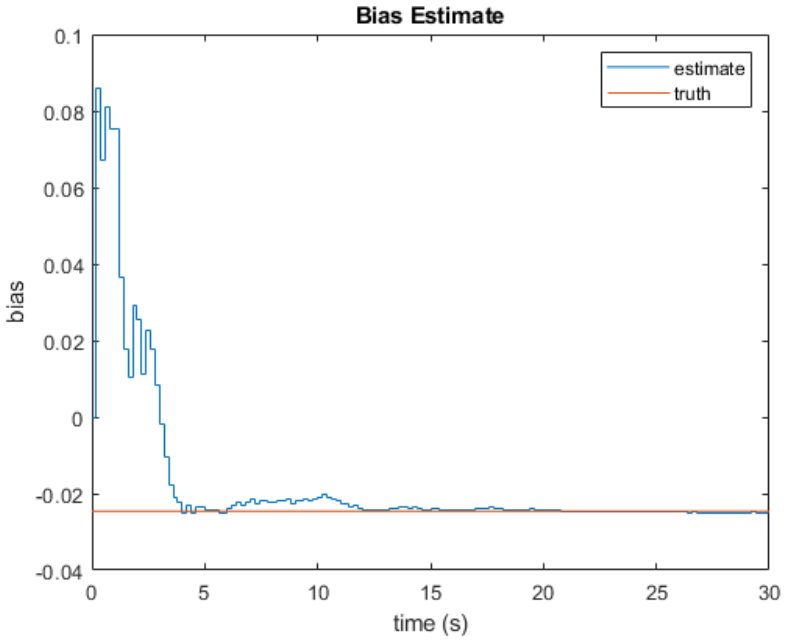


Figure 6: Bias Estimate, $\hat{b}(t_i)$

4.2 Filter Validation and Simulation

The derived Kalman Filter was validated by running a Monte Carlo Simulation with $N = 10,000$ realizations. Orthogonality properties were tested over the ensemble and a comparison of the error variance of one realization versus the ensemble average was made.

First, the filter was validated by comparing the simulated error variance, P^{ave} found with Equation (16), with the filter error variance derived from one realization, P. From Figure 7, one can see the difference is minimal with the max difference being $\max(P^{ave} - P) = -0.013$ for first term in the matrices. Over time however, the difference becomes even smaller as the Kalman Filter is updated with new measurement information.

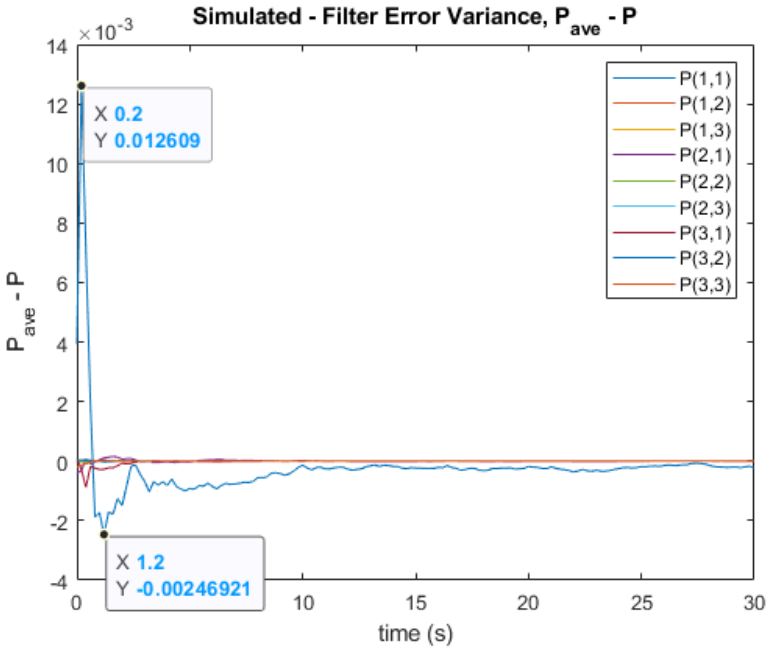


Figure 7: Simulated and Filter Variance Comparison

The orthogonality of the error in estimates with the estimate itself is theoretically zero. Using Equation (17) over the ensemble, the value at which the property was at a maximum at certain t_i was recorded. As shown below, the orthogonality property is approximately zero (even at it's maximal value), verifying the theoretical expectation.

```
orthogonality of error in estimates (at ti with max value):
   -0.0188    0.0012    0.0001
   -0.0016    0.0002    0.0001
    0.0007   -0.0002   -0.0001
```

Lastly, the independence of the residuals was checked using Equation (18). This was calculated for the residuals at the last time step ($t_i = 30s$) and second to last time step ($t_m = 29.80s$). The matrix is approximately zero, indicating the residuals form an independent sequence.

```
ensemble average for correlation of the residuals:
   -0.0034   -0.0005
   -0.0005   -0.0000
```

5 Conclusion

To conclude, the derived Kalman Filter was successful in determining the minimum variance estimates of position, velocity and accelerometer bias of the vehicle. The estimation error of each parameter generally stayed within the one sigma bounds with the exception of some short time periods as expected. The theoretical properties of the system also were validated using a Monte Carlo Simulation; namely checking the simulation error variance was approximately the filter error veriance, proving the orthogonality of estimation error and the estimate itself, and the independence of residuals. Proving these theoretical properties through simulation confirmed a functional Kalman Filter algorithm.

A Monte Carlo Simulation

```
%Monte Carlo Simulation
%Helene Levy
clc; clear; close all;

%Acceleration Model
ts = 1/200; %sample period, freq = 200 Hz
t = 0:ts:30;
t_meas = 0:40*ts:30; %meas freq = 5 Hz

%preallocations
e_sum = zeros(3,length(t_meas));
e_ave = zeros(3,length(t_meas));
r_sum = zeros(2,2);
% ortho_sum = zeros(3,3);

%cell preallocations
ortho_ave = {0}; ortho_sum = {0};
P_ave ={0}; P_sum = {0};

for i=1:length(t_meas)
    P_sum{i} = zeros(3,3);
    ortho_sum{i} = zeros(3,3);
end

%Monte Carlo simulation
N_ave = 10000;
for j = 1: N_ave
     [X_hat,e_hat,P,r] = kalman_filter();
     e_sum = e_sum + e_hat;
     e_ave = 1/j*e_sum;
```

```
31
32      for k = 1:length(e_hat)
33          P_sum{k} = ...
                P_sum{k}+(e_hat(:,k)-e_ave(:,k))*(e_hat(:,k)-e_ave(:,k))';
34          P_ave{k} = 1/(j-1)*P_sum{k};
35
36          ortho_sum{k} = ortho_sum{k}+(e_hat(:,k)-e_ave(:,k))*X_hat(:,k)';
37          ortho_ave{k} = 1/j*ortho_sum{k};
38      end
39      r_sum = r_sum + r(:,end)*(r(:,end-1))';
40      r_ave = 1/j*r_sum;
41  end
42  %displaying results
43
44  %error variance
45
46  figure;
47  dP_plot = zeros(9,length(P_ave));
48  for i = 1:length(P_ave{1})^2 %going 1 -> 9
49      for k = 1:length(P_ave) %going 1 -> 151
50          dP_plot(i,k) = P_ave{k}(i)-P{k}(i);
51      end
52      plot(t_meas,dP_plot(i,:)); hold on;
53  end
54  legend('P(1,1)','P(1,2)','P(1,3)','P(2,1)','P(2,2)','P(2,3)','P(3,1)',...
55          'P(3,2)','P(3,3)','Location','southeast');
56  title('Simulated - Filter Error Variance, P_{ave} - P');
57  xlabel('time (s)');
58  ylabel('P_{ave} - P');
59
60  %ortho
61
62  max_orth = zeros(3,3);
```

```
63 for k = 1:length(ortho_ave) %going 1 -> 151
64     if abs(ortho_ave{k}) > max_orth
65     max_orth = ortho_ave{k};
66     t_max = t_meas(k);
67     end
68 end
69 disp('orthogonality of error in estimates (at ti with max value):');
70 disp(max_orth)
71
72 %residuals
73 disp('ensemble average for correlation of the residuals:');
74 disp(r_ave);
```

B Kalman Filter Function

```
function [X_hat,e_hat,P,r] = kalman_filter()
%time definition
ts = 1/200; %sample period, freq = 200 Hz
t = 0:ts:30;
t_meas = 0:40*ts:30; %meas freq = 5 Hz

%initial position and velocity stats
p0_m = 0; v0_m = 100;
p0 = normrnd(p0_m,10); %m/s
v0 = normrnd(v0_m,1); %m/s

%bias and w stats
b = normrnd(0,sqrt(0.01));
w = normrnd(0,sqrt(0.0004),1,length(t));
W = 0.0004; %variance matrix of w

%measurement white noise stats
eta_1i = normrnd(0,1,1,length(t_meas)); %sd = 1m
eta_2i = normrnd(0,0.04,1,length(t_meas)); %sd = 4 cm/s = 0.4 m/s
eta = [eta_1i;eta_2i];
%variance matrix of eta
V = [1 0 ; 0 (0.04)^2];

%Acceleration Model
omega = 0.1; %rad/s
a = 10*sin(omega*t);
v = v0 + 10/omega-10/omega*cos(omega*t);
p = p0 + (v0+10/omega)*t-10/(omega^2)*sin(omega*t);

%Accelerometer model
```

```
31 a_c = a + b + w;
32
33 %state space model
34 Phi = [1 ts -ts^2/2; 0 1 -ts; 0 0 1];
35 Gamma = -[ts^2/2; ts; 0];
36
37 %preallocation
38 v_c = zeros(1,length(t));
39 p_c = zeros(1,length(t));
40
41 %initial states
42 v_c(1) = v0_m; p_c(1) = p0_m; %acclerometer initial value = mean of actual
43 v_e(1) = v0; p_e(1) = p0; %actual initial value is a random number w ...
       stats ^
44
45 %state space model X = [dp; dv; b]
46 X(:,1) = [p_e(1)-p_c(1);v_e(1)-v_c(1);b];
47 for j = 1:length(t)-1
48     v_c(j+1) = v_c(j) + a_c(j)*ts;
49     p_c(j+1) = p_c(j) + v_c(j)*ts + a_c(j)*ts^2/2;
50     %state space model propagation
51     X(:,j+1) = Phi*X(:,j)+Gamma*w(j);
52 end
53 %measured gps value propagation
54 H = [1 0 0; 0 1 0];
55 Z = H*X(:,1:40:end)+eta;
56
57 M0 = [10^2 0 0; 0 1 0; 0 0 0.01]; %variance matrix of x0
58
59 P = {0}; K = {0}; M ={0};
60 X_bar = zeros(size(X));
61 X_hat = zeros(3,length(t_meas));
62 %initial value of xbar = initial means of actual model
```

```
X_bar(:,1) = [p0_m; v0_m;0];
M{1} = M0;
for k = 1:length(t_meas)
    %measurements happen every 40 time steps ex, 1,41,81 ...
    %X_hat only has values at these indices of X_bar
    meas_ind = 40*(k-1)+1;

    %note matrices Phi, Gamma, H and V don't change because given
    P{k} = inv(inv(M{meas_ind})+H'*inv(V)*H);
    K{k} = P{k}*H'*inv(V);
    X_hat(:,k) = X_bar(:,meas_ind)+K{k}*(Z(:,k)-H*X_bar(:,meas_ind));

    %don't want to add another point and propagate 40 points if at end
    if k < length(t_meas)
        %now know X_bar and M at next index after the measurement index
        X_bar(:,meas_ind+1) = Phi*X_hat(:,k);
        M{meas_ind+1} = Phi*P{k}*Phi'+Gamma*W*Gamma';

        %because there aren't measurements for in-between time steps
        %must propagate from X_bar and M_k
        for j = 1:40-1
            X_bar(:,meas_ind+j+1) = Phi*X_bar(:,meas_ind+j);
            M{meas_ind+j+1} = Phi*M{meas_ind+j}*Phi'+Gamma*W*Gamma';
        end
    end
end
%plotting estimate of position and actual
e_bar = X(:,1:40:end) - X_bar(:,1:40:end);
e_hat = X(:,1:40:end) - X_hat;
%residual
r = Z-H*X_bar(:,1:40:end);
end
```

C Kalman Filter (One Realization)

```
%MAE 271A Project
%Helene Levy
clc; clear; close all;
%time definition
ts = 1/200; %sample period, freq = 200 Hz
t = 0:ts:30;
t_meas = 0:40*ts:30; %meas freq = 5 Hz

%initial position and velocity stats
p0_m = 0; v0_m = 100;
p0 = normrnd(p0_m,10); %m
v0 = normrnd(v0_m,1); %m/s

%bias and w stats
b = normrnd(0,sqrt(0.01));
w = normrnd(0,sqrt(0.0004),1,length(t));
W = 0.0004; %variance matrix of w

%measurement white noise stats
eta_1i = normrnd(0,1,1,length(t_meas)); %sd = 1m
eta_2i = normrnd(0,0.04,1,length(t_meas)); %sd = 4 cm/s = 0.4 m/s
eta = [eta_1i;eta_2i];
%variance matrix of eta
V = [1 0 ; 0 (0.04)^2];

%Acceleration Model
omega = 0.1; %rad/s
a = 10*sin(omega*t);
v = v0 + 10/omega-10/omega*cos(omega*t);
```

```
30 p = p0 + (v0+10/omega)*t-10/(omega^2)*sin(omega*t);
31
32 %Accelerometer model
33 a_c = a + b + w;
34
35 %state space model
36 Phi = [1 ts -ts^2/2; 0 1 -ts; 0 0 1];
37 Gamma = -[ts^2/2; ts; 0];
38
39 %preallocation
40 v_c = zeros(1,length(t));
41 p_c = zeros(1,length(t));
42
43 %initial states
44 v_c(1) = v0_m; p_c(1) = p0_m; %acclerometer initial value = mean of actual
45 v_e(1) = v0; p_e(1) = p0; %actual initial value is a random number w ...
       stats ^
46
47 %state space model X = [dp; dv; b]
48 X(:,1) = [p_e(1)-p_c(1);v_e(1)-v_c(1);b];
49 for j = 1:length(t)-1
50     v_c(j+1) = v_c(j) + a_c(j)*ts;
51     p_c(j+1) = p_c(j) + v_c(j)*ts + a_c(j)*ts^2/2;
52     %state space model propagation
53     X(:,j+1) = Phi*X(:,j)+Gamma*w(j);
54 end
55 %measured gps value propagation
56 H = [1 0 0; 0 1 0];
57 Z = H*X(:,1:40:end)+eta;
58
59 M0 = [10^2 0 0; 0 1 0; 0 0 0.01]; %variance matrix of x0
60
61 P = {0}; K = {0}; M ={0};
```

```
X_bar = zeros(size(X));
X_hat = zeros(3,length(t_meas));
%initial value of xbar = initial means of actual model
X_bar(:,1) = [p0_m; v0_m;0];
M{1} = M0;
for k = 1:length(t_meas)
    %measurements happen every 40 time steps ex, 1,41,81 ...
    %X_hat only has values at these indices of X_bar
    meas_ind = 40*(k-1)+1;

    %note matrices Phi, Gamma, H and V don't change because given
    P{k} = inv(inv(M{meas_ind})+H'*inv(V)*H);
    K{k} = P{k}*H'*inv(V);
    X_hat(:,k) = X_bar(:,meas_ind)+K{k}*(Z(:,k)-H*X_bar(:,meas_ind));

    %don't want to add another point and propagate 40 points if at end
    if k < length(t_meas)
        %now know X_bar and M at next index after the measurement index
        X_bar(:,meas_ind+1) = Phi*X_hat(:,k);
        M{meas_ind+1} = Phi*P{k}*Phi'+Gamma*W*Gamma';

        %because there aren't measurements for in-between time steps
        %must propagate from X_bar and M_k
        for j = 1:40-1
            X_bar(:,meas_ind+j+1) = Phi*X_bar(:,meas_ind+j);
            M{meas_ind+j+1} = Phi*M{meas_ind+j}*Phi'+Gamma*W*Gamma';
        end
    end
end
%plotting estimate of position and actual
e_bar = X(:,1:40:end) - X_bar(:,1:40:end);
e_hat = X(:,1:40:end) - X_hat;
%residual
```

```
95  r = Z-H*X_bar(:,1:40:end);
96
97  %getting sigma bounds from conditional error variance
98  sig = zeros(3,length(t_meas));
99  for k = 1:length(t_meas)
100     %sigma values for each error variance
101     sig(:,k) = sqrt(diag(P{k}));
102 end
103
104 figure
105 stairs(t_meas,e_hat(1,:));hold on;
106 plot(t_meas,sig(1,:),'r--',t_meas,-sig(1,:),'r--');
107 title('Position Error');
108 xlabel('time (s)');
109 ylabel('position error');
110
111 figure
112 stairs(t_meas,e_hat(2,:)); hold on;
113 plot(t_meas,sig(2,:),'r--',t_meas,-sig(2,:),'r--');
114 title('Velocity Error');
115 xlabel('time (s)');
116 ylabel('velocity error');
117
118 figure
119 stairs(t_meas,e_hat(3,:)); hold on;
120 plot(t_meas,sig(3,:),'r--',t_meas,-sig(3,:),'r--');
121 title('Bias Error');
122 xlabel('time (s)');
123 ylabel('bias error');
124
125 %zoomed in bias error
126 figure
127 stairs(t_meas(end-50:end),e_hat(3,end-50:end)); hold on;
```

```
128 plot(t_meas(end-50:end),sig(3,end-50:end),'r--', ...
129     t_meas(end-50:end),-sig(3,end-50:end),'r--');
130 title('Zoom Bias Error');
131 xlabel('time (s)');
132 ylabel('bias error');
133
134
135 figure
136 stairs(t_meas,X_hat(1,:)+p_c(1:40:end)); hold on;
137 plot(t,p);
138 title('Position Estimate');
139 xlabel('time (s)');
140 ylabel('position (m)');
141 legend('estimate','truth');
142
143 figure
144 stairs(t_meas,X_hat(2,:)+v_c(1:40:end)); hold on;
145 plot(t,v);
146 title('Velocity Estimate');
147 xlabel('time (s)');
148 ylabel('velocity (m/s)');
149 legend('estimate','truth');
150
151 figure
152 stairs(t_meas,X_hat(3,:)); hold on;
153 plot(t,b*ones(1,length(t)));
154 title('Bias Estimate');
155 xlabel('time (s)');
156 legend('estimate','truth');
157 ylabel('bias (m/s^2)');
```