Calibration of an Accelerometer Using GPS Measurements

Author:Yi Cui
ID number :605068398

University of California, Los Angeles

C271A - Probability and Stochastic Processes in Dynamical Systems

Final Project

Due Date

Dec 4, 2017

This thesis is dedicated to
UCLA

Abstract

- **Background**:

Our objective of this final project is to estimate the position and velocity of a vehicle. For instance, when the signals are not available or have some obstructions, we can take advantage of other methods. In this paper, we use one quite essential way. We can add a stuff like accelerometer and use the control system as well as the measurement to acquire a well-estimated vehicle's final information. Also, in order to get a relatively accurate result, we should calibrate the accelerometer.

- **General idea**:

We implement the GPS and accelerometer to get the measurement and dynamic system together through some algorithms. Meanwhile, the frequency of the accelerometer and the GPS is different, which is 200 Hz and 5Hz. Based on the error, we can use Kalman filter algorithm to analysis them. After that, we propagate according variables to update the system. Consequently, the result shows that although the Kalman filter exists some flaws, the overall results and the bound of error are quite convincing and nearly perfect.

Contents

List of Figures

Chapter 1

Introduction

In the project, what we have at first are two things: Accelerometer and GPS. The measurements of the vehicle are corrupted with bias and noise. Also, both accelerometer and GPS have their own demerits due to bias and noise. In order to get the perfect estimate of the vehicle, we should use some analysis algorithms. We first use Kalman Filter Algorithm through the error space. Then we propagate the variables and update the system. Meanwhile, what we should bear in mind is that the errors between the truth and our model are quite essential, which we can use to optimize the dynamic system and then calibrate the sensor. Another important process is the Monte Carlo simulation. We use it to ensure the availability and stability of the implemented Kalman Filter Algorithm, together with some other useful checking methods.

In my perspective, the Flow Chart obviously illustrates the whole process in this project and in the following sections I will pay more attention to deeply demonstrating my analysis and algorithm.

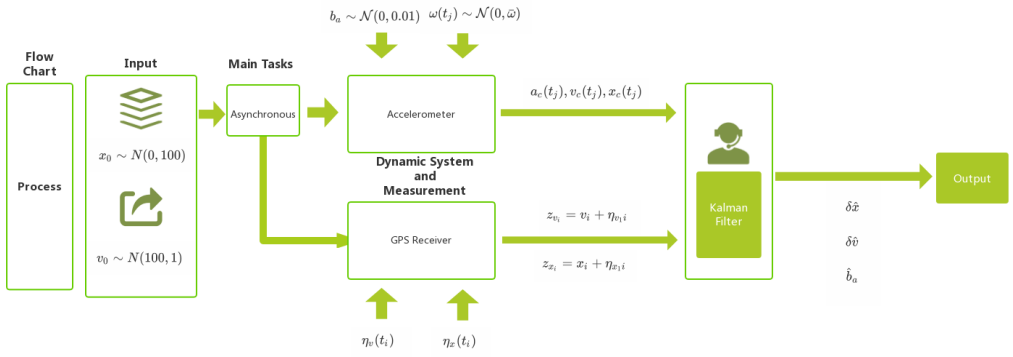

Figure 1.1: The Flow Chart

Chapter 2

Analysis and Algorithm

2.1 Basic Conditions

We know that the accelerometer a_c is modelled as $a_c(t_j) = a(t_j) + b_a + w(t_j)$, and we assume that the acceleration is a harmonic of the form $a(t) = 10sin(2\pi\omega t)$ $meters/sec^2$,where $\omega = 0.1rad/sec$.

Also, we know that the additive white Gaussian noise $\omega \sim (0, 0.0004)$, $b_a \sim \mathcal{N}(0, 0.01)$, $t_{sa} = \frac{1}{200}$.(sample accelerometer)

What we should bear in mind is that b_a is generated at the first time and remains unchanged all the time.

Then, we see the GPS receiver, from which we can see that

$$\begin{cases} z_{1i} = x_i + \eta_{1i} \\ z_{2i} = v_i + \eta_{2j} \end{cases}$$

where the $x_0 \sim \mathcal{N}(0, 10^2)$, $v_0 \sim \mathcal{N}(100, 1)$, $t_{sa} = \frac{1}{5}$.(sample accelerometer)

$$\begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0.0016 \end{bmatrix})$$

2.2 The True Model

From what we have known above, we can model our true acceleration system formula.

$$a(t) = asin(2\pi\omega t) \tag{1}$$

$$v(t) = v(0) + \frac{a}{2\pi\omega} - \frac{a}{2\pi\omega}cos(2\pi\omega t) \tag{2}$$

and we do not know the original beginning point.

$$x(t) = x(0) + (v(0) + \frac{a}{2\pi\omega})t - \frac{a}{(2\pi\omega)^2} sin(2\pi\omega t) \tag{3}$$

The above three formulas come from the relationship among $a(t), v(t), x(t)$,which means we can use integration and derivation to get the forms. And where $a = 10m/s^2, \omega = 0.1rad/s^2$.

Also, where the statistics for the initial values of velocity v and position x are $v_0 \sim \mathcal{N}(\bar{v_0}, M_0^v)$ and x is $\mathcal{N}(\bar{x_0}, M_0^x)$, respectively, with values of the mean and variance as have been given before.

Using these formulas, the actual acceleration, velocity and position of the vehicle over the period of 30 seconds for one realization are plotted as following.

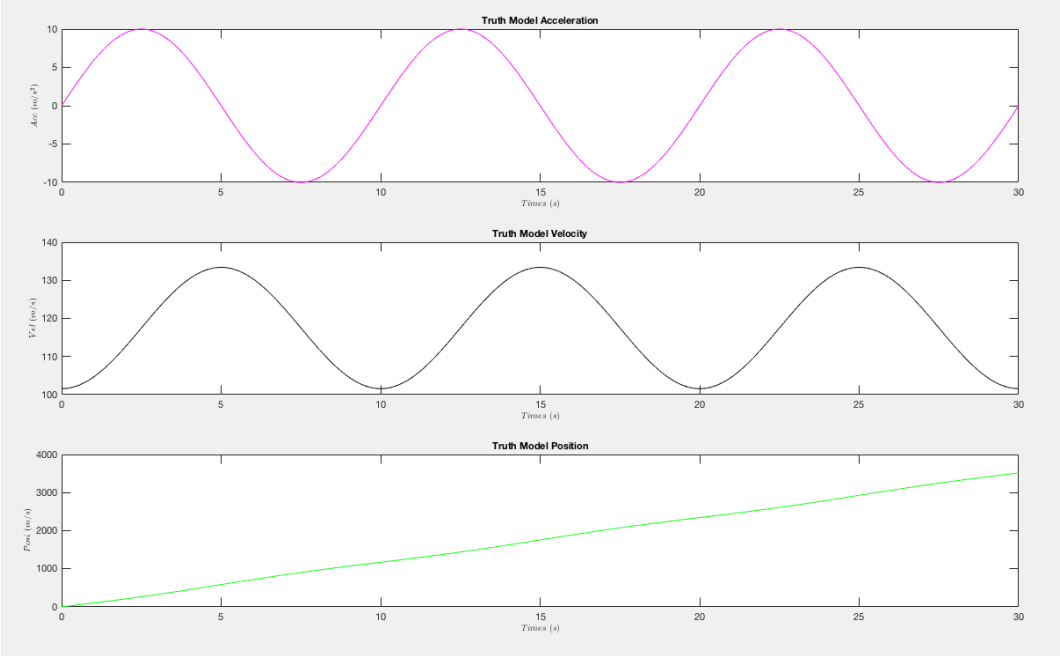

Figure 2.1: True Model's Parameters under the Condition that Second=30

```
%case
second = 30 ;
%General
%
%Truth Model
%

% loop for t=1:1:1000*second
frequency = 0.1 ; %rad/ s
harmonic= (2.*pi)*frequency;
```

Figure 2.1

```
a(t)= 10*sin(harmonic.*(t/1000));
v(t)= v0+(10/harmonic)-(10/harmonic).*cos(harmonic*(t/1000));
x(t)= x0+(v0+(10/harmonic))*(t/1000)-(10/(harmonic^2))*(sin(harmonic*(t
    /1000)));
% end loop

figure(1)
subplot(3,1,1)
plot(seconds,a,'m')
ylabel('$Acc$ $(m/s^2)$','FontSize',10 ,'Interpreter','Latex');
xlabel( '$Times$ $(s)$','FontSize',10,'Interpreter','Latex');
title ( '\bf{Truth Model Acceleration}','FontSize' ,11 );

subplot(3,1,2)
plot(seconds,v,'k')
ylabel('$Vel$ $(m/s)$','FontSize',10 ,'Interpreter','Latex');
xlabel( '$Times$ $(s)$','FontSize',10,'Interpreter','Latex');
title ( '\bf{Truth Model Velocity}','FontSize' ,11 );

subplot(3,1,3)
plot(seconds,p,'g')
ylabel('$Posi$ $(m/s)$','FontSize',10 ,'Interpreter','Latex');
xlabel( '$Times$ $(s)$','FontSize',10,'Interpreter','Latex');
title ( '\bf{Truth Model Position}','FontSize' ,11 );
```

Figure 2.1

2.3 The Accelerometer Model and Dynamic System

Meanwhile, we can then form the accelerometer model.

$$a_c(t_j) = a(t_j) + b_a + \omega(t_j) \tag{4}$$

where $\Delta t = \frac{1}{200}$, then we use a basic Euler's Method, we can get:

$$v_c(t_{j+1}) = v_c(t_j) + a_c(t_j)\Delta t \tag{5}$$

where $v_c(0) = \bar{v}_0 = 100$

$$x_c(t_{j+1}) = x_c(t_j) + v_c(t_j)\Delta t + a_c(t_j)\frac{\Delta t^2}{2} \tag{6}$$

where $x_c(0) = \bar{x}_0 = 0$

The following figures show the difference between the true model and the accelerometer.

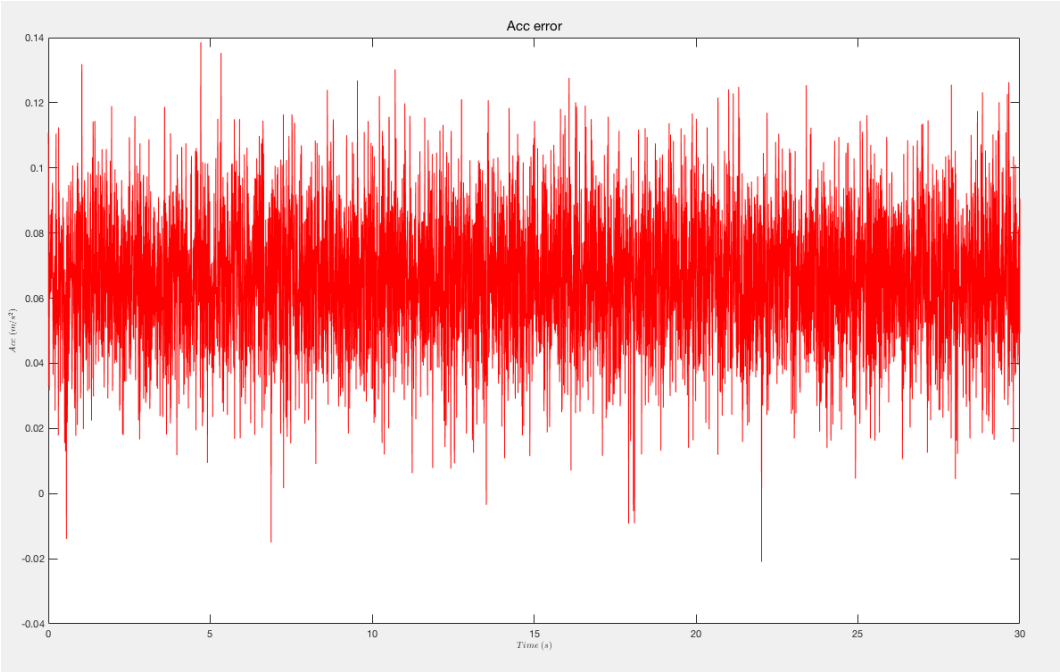

Figure 2.2: Acc Error

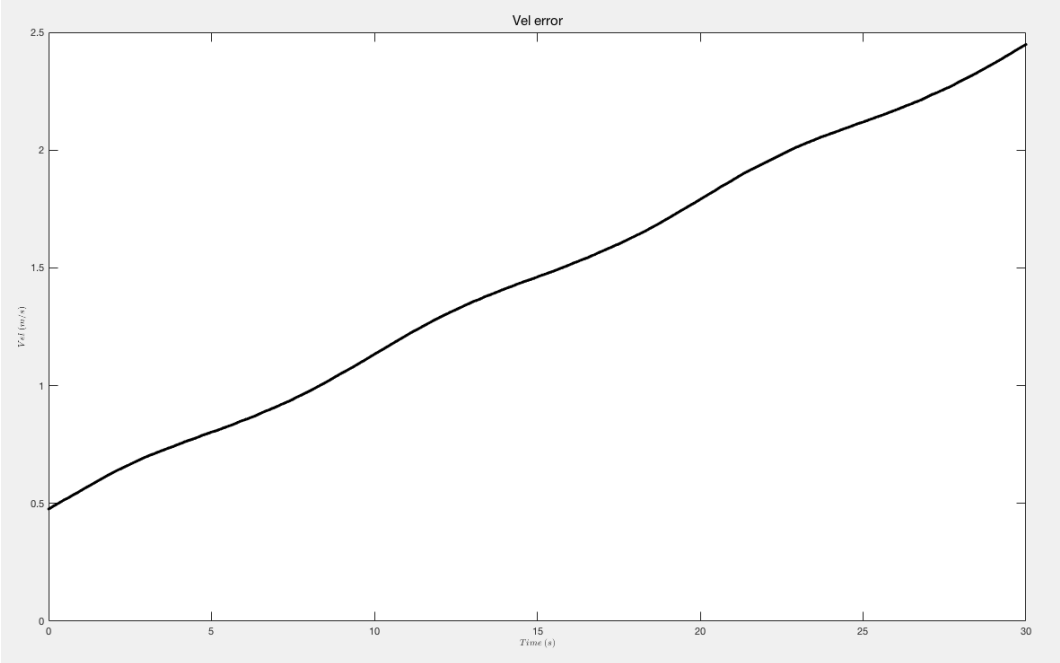

Figure 2.3: Vel Error

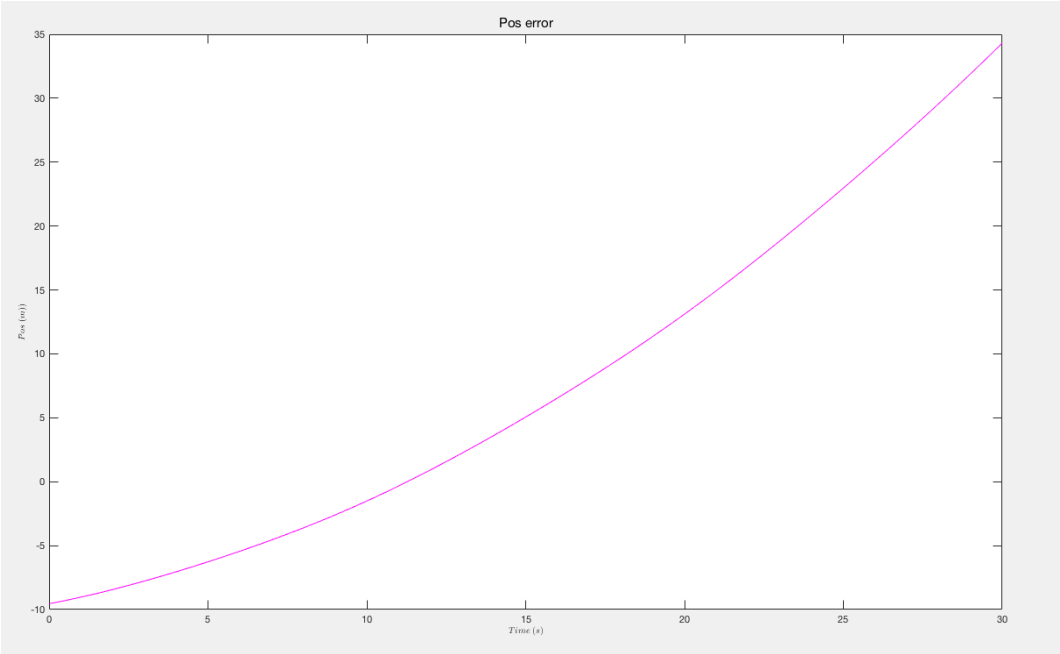

Figure 2.4: Pos Error

```
figure(2)
plot(clock_accelerometer,Aerr_truth_acc,'-r')
xlabel('$Time$ $(s)$' , 'FontSize' ,10 , 'Interpreter' , 'Latex' ) ;
ylabel('$Acc$ $ (m/s^2$)' , 'FontSize' ,10 , 'Interpreter' , 'Latex');
title('Acc error' , 'FontSize',15) ;

figure(3)
plot(clock_accelerometer,Verr_truth_acc,'k.')
xlabel('$Time$ $(s)$' , 'FontSize' ,10 , 'Interpreter' , 'Latex' ) ;
ylabel('$Vel$ $ (m/s$)' , 'FontSize' ,10 , 'Interpreter' , 'Latex');
title('Vel error' , 'FontSize',15) ;

figure(4)
plot(clock_accelerometer,Perr_truth_acc,'m-')
xlabel('$Time$ $(s)$' , 'FontSize' ,10 , 'Interpreter' , 'Latex' ) ;
ylabel('$Pos$ $ (m)$)' , 'FontSize' ,10 , 'Interpreter' , 'Latex');
title('Pos error' , 'FontSize',15) ;
```

Overall Codes

Our goal is to keep the system model for the Kalman Filter from the influence of $a(t)$, thus we assume the true acceleration can also be used by Euler Method.

Afterwards, we remove the average, which can be seen in the following sections

$$v(0) = v_E(0) \sim (100, 1)$$

$$x(0) = x_E(0) \sim (0, 100)$$

Then we use a basic Euler's Method, we can get:

$$v_E(t_{j+1}) = v_E t_j + a(t_j)\Delta t \tag{7}$$

$$x_E(t_{j+1}) = x_E(t_j) + v_E(t_j)\Delta t + a(t_j)\frac{\Delta t^2}{2} \tag{8}$$

We use (8)-(6), then get the following:

$$\begin{bmatrix} \delta X_E(t_{j+1}) \\ \delta V_E(t_{j+1}) \\ b(t_{j+1}) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & -\frac{\Delta t^2}{2} \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta X_E(t_j) \\ \delta V_E(t_j) \\ b(t_j) \end{bmatrix} - \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \\ 0 \end{bmatrix} \omega(t_j) \tag{9}$$

where $\delta X_E(t_0) = X_E(t_0) - X_C(t_0) \quad \sim \mathcal{N}(0, 100)$

$$\delta V_E(t_0) = V_E(t_0) - V_C(t_0) \quad \sim \mathcal{N}(0, 1)$$

$$E[\omega(t_j)] = 0, E[\omega(t_j)\omega(t_l)^T] = \mathbb{W}\delta_{j,l}$$

$$b(0) \sim \mathcal{N}(0, 0.01)$$

After finishing the basic modelling process, we come to the core process.

2.4 Measurement(GPS Model)

$$z_x(t_i) = x(t_i) + \eta_x(t_i) \tag{10}$$

$$z_v(t_i) = v(t_i) + \eta_v(t_i) \tag{11}$$

After filtering the deterministic factor(mean), we can get

$$\delta z_x(t_i) = \delta x(t_i) + \eta_x(t_i) \tag{12}$$

$$\delta z_v(t_i) = \delta v(t_i) + \eta_v(t_i) \tag{13}$$

And then, we can shift actual value of position and velocity to zero mean:

$$\delta x(t_i) = x(t_i) - x_c(t_i) \tag{14}$$

$$\delta v(t_i) = v(t_i) - v_c(t_i) \tag{15}$$

We should also note that we assume these two white noise sequences are independent of each other, which means that the off-diagonal elements of the corresponding covariance matrix are zero.

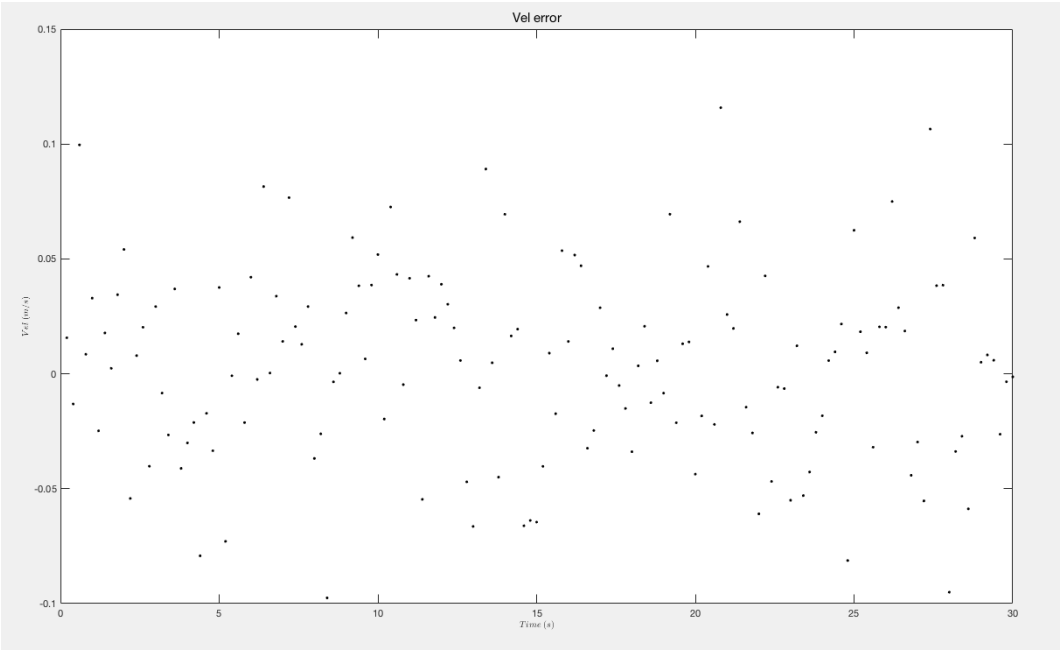

Figure 2.5: Vel Error between GPS and Truth Model.

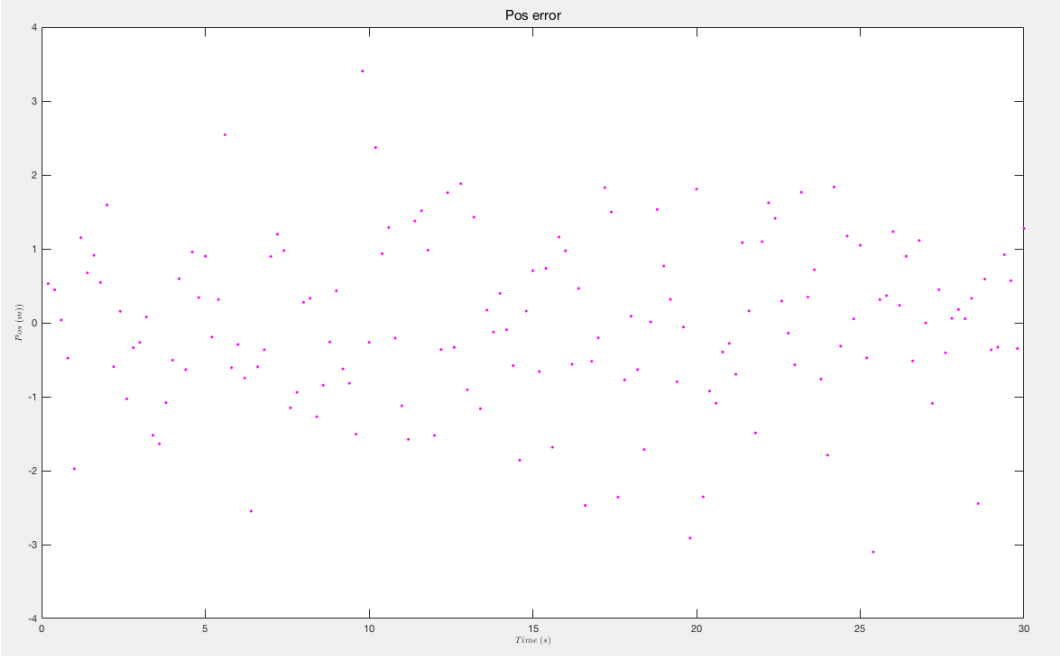

Figure 2.6: Pos Error between GPS and Truth Model.

Then, we generate the Acceleremeter and the GPS model together in the following figures. Also we should note that the accelerometer and GPS are synchronized which is very important and simplifies the problem.

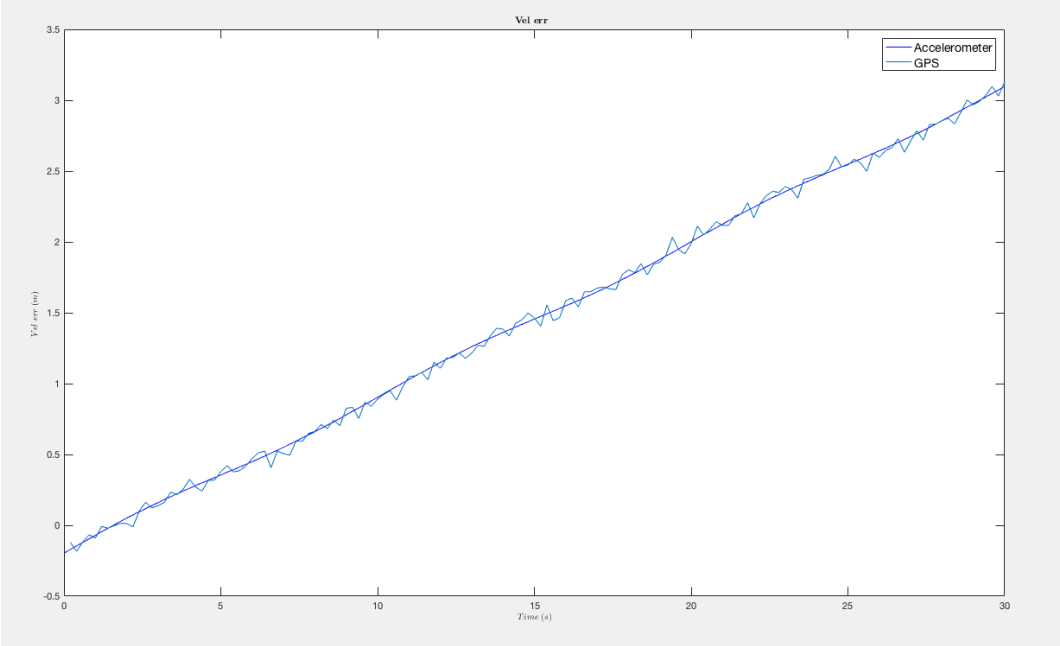

Figure 2.7: Vel Error between the Acceleremeter and the GPS.

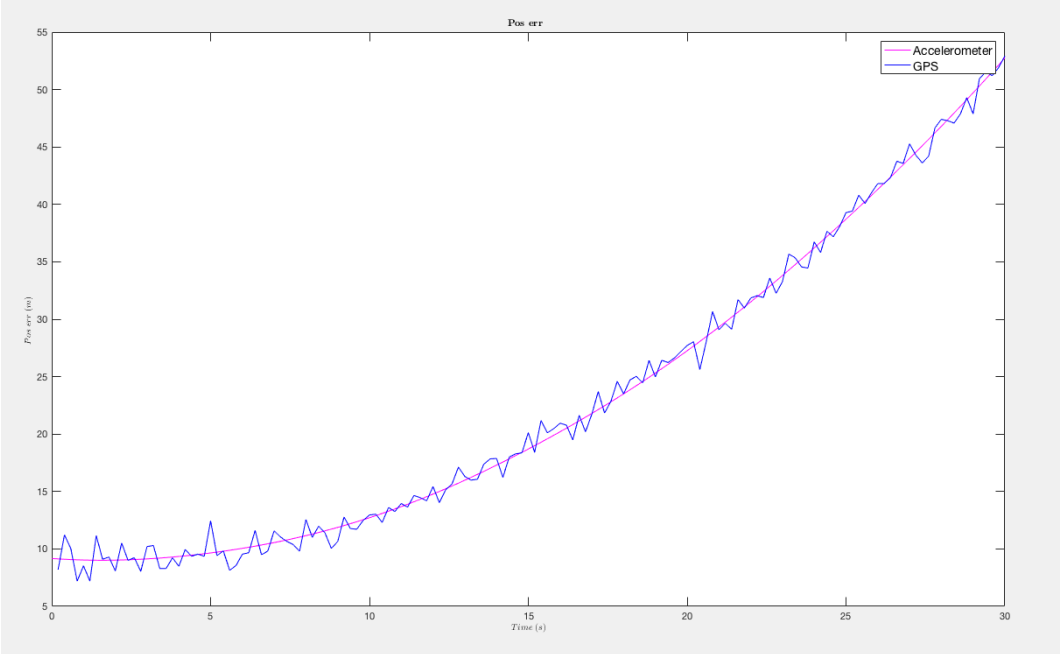

Figure 2.8: Pos Error between the Acceleremeter and the GPS.

The following Kalman Filter algorithm will combine these two along with the information of the bias and obtain a better estimate of these parameters.

2.5 Kalman Filter

Most importantly, we use the Kalman Filter to solve this model.

We assume that $\delta x(t_i) = \delta x_E(t_i)$ and $\delta v(t_i) = \delta v_E(t_i)$

Also assume that Euler formula is close to real value in GPS model.

Then, we use the system that is illustrated on the book.[1]

$$x_{k+1} = \Phi_k x_k + \Gamma_k \omega_k \quad , x_0 \sim \mathcal{N}(\bar{x_0}, M_0) \tag{16}$$

$$z_k = H_k x_k + v_k \tag{17}$$

We should notice that ω_k and v_k are zero mean, Gaussian, white-noise processes with covariances, W_k and V_k respectively.

w_k, v_k and x_0 are all independent of each other.

Between the time $k-1$ and time k, we know that at time $k-1$, we have the conditional mean estimate, $\hat{x}_{k-1}$ and conditional covariance P_{k-1}, and at time k, we will get another measurement, z_k

$$\begin{bmatrix} \delta X(t_{i+1}) \\ \delta V(t_{i+1}) \\ b(t_{i+1}) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & -\frac{\Delta t^2}{2} \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta X(t_i) \\ \delta V(t_i) \\ b(t_i) \end{bmatrix} - \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \\ 0 \end{bmatrix} \omega(t_i) \tag{18}$$

$$\begin{bmatrix} \delta z_x(t_i) \\ \delta z_v(t_i) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta x(t_i) \\ \delta v(t_i) \\ b(t_i) \end{bmatrix} + \begin{bmatrix} \eta_x(t_i) \\ \eta_v(t_i) \end{bmatrix} \tag{19}$$

We know the posteriori conditional mean $\delta\hat{\mathcal{X}}(t_i)$, and error variance is $\mathcal{P}(t_i)$[2]

Therefore, we can easily get the equation that

$$\delta\hat{\mathcal{X}}(t_i) = \begin{bmatrix} \delta\hat{X}(t_i) \\ \delta\hat{\mathcal{V}}(t_i) \\ \hat{b}(t_i) \end{bmatrix} = \begin{bmatrix} \hat{x}(t_i) - x_c(t_i) \\ \hat{v}(t_i) - v_c(t_i) \\ \hat{b}(t_i) \end{bmatrix} \tag{20}$$

We know that the accelerometer provides data every 0.005 seconds and GPS provides data every 0.2 seconds.[3]

Pseudo-code for the Kalman Filter Algorithm

If the data from accelerometer are available and others are not available, the following relations will be used to propagate the states and a priori covariance matrix:

The process of propagation

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

$M_{k+1} = \Phi M_k \Phi^T + \Gamma W \Gamma^T$

If the data from GPS are available and others are not available, the following relations will be used to propagate the states and then update them.

The process of propagation and updating

$P_k = (M_k^{-1} + H^T V^{-1} H)^{-1} = (I - K_k H) M_k (I - K_k H)^T + K_k V K_k^T$ (Matrix Lemma)

$K_k = P_k H^T V^{-1} = M_k H^T (H M_k H^T + V)^{-1}$

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

$\bar{x}_k = \Phi \hat{x}_k$

$M_k = \Phi P_k \Phi^T + \Gamma W \Gamma^T$

We here prove a lemma, which is quite useful for calculating and simplifying the inverse matrix.

Matrix Lemma

$$(A - BC^{-1}D)^{-1} = A^{-1} + A^{-1}B(C - DA^{-1}B)^{-1}DA^{-1}$$

which then is easy to prove

$$P = M - MH^T(HMH^T + V)^{-1}HM$$

where previously we defined P to be

$$P = (M^{-1} + H^T V^{-1} H)^{-1}$$

using the result to show that

$$PH^T V^{-1} = MH^T(HMH^T + V)^{-1}$$

Proof: We can first multiply $(A - BC^{-1}D)$ on both sides, we obtain

$I = ((A - BC^{-1}D))(A^{-1} + A^{-1}B(C - DA^{-1}B)^{-1}DA^{-1})$

Then, we obtain four terms

$I = I + B(C - DA^{-1}B)^{-1}DA^{-1} - BC^{-1}DA^{-1} - BC^{-1}DA^{-1}B(C - DA^{-1}B)^{-1}DA^{-1}$

Consider the middle two terms, we can get

$B(C - DA^{-1}DA^{-1}B)^{-1} - BC^{-1}DA^{-1} = BC^{-1}[C(C - DA^{-1}B)^{-1} - I]DA^{-1}$

$= BC^{-1}DA^{-1}B(C - DA^{-1}B)^{-1}DA^{-1}$

which is the same as the fourth term, thus we get the result.

Chapter 3

Check Theory

3.1 Monte Carlo Simulation

After that, we start the checks in theory.

After my consideration, the actual variance obtained from "Monte Carlo" simulation is close to the error variance $\mathcal{P}(t_i)$ in Kalman Filter.

We define

$$\delta\hat{\mathcal{X}}(t_j) = \begin{bmatrix} \delta X(t_j) \\ \delta\mathcal{V}(t_j) \\ b(t_j) \end{bmatrix} = \begin{bmatrix} x(t_j) - x_c(t_j) \\ v(t_j) - v_c(t_j) \\ b(t_j) \end{bmatrix} = \mathcal{X}_{true} - \mathcal{X}_c \tag{20}$$

a priori error

$$\bar{\theta}(t_j) = \delta x(t_j) - \delta\bar{x}(t_j) = \begin{bmatrix} x(t_j) - \bar{x}(t_j) \\ v(t_j) - \bar{x}(t_j) \\ b(t_j) - \bar{b}(t_j) \end{bmatrix} \tag{21}$$

$(\delta\bar{x}(t_j) = \Phi(t_{j-1})\delta\hat{x}(t_{j-1}))$

At time $t_i = t_j$

$$\bar{e}(t_i) = \delta x(t_i) - \delta\bar{x}(t_i^-) \qquad a \quad priori \tag{22}$$

$$e(t_i) = \delta x(t_i) - \delta\hat{x}(t_i^+) \qquad posteriori \tag{23}$$

However, the above method is only used in the case $i = j$. Instead, we use Monte Carlo Averaging:

The simulation runs multiple times where each time all the random noise generation have a new seed.

$EX = \delta X - \delta\hat{X}$, and $(\delta X - \delta\hat{X})^2$ are what we are interested in.

We know that $\frac{1}{N}\sum_{l=1}^{N}(\delta X_{\sum}^{l} - \delta\hat{X}_{\sum}^{l}) = (\sigma_{X,\delta})^2$

And we have this in our Kalman Filter $X(1,1)$ at time δ
Therefore, we try to get appropriate l until it convergence.

$$e^{avg}(t_i) = \frac{1}{N_{avg}} \sum_{l=1}^{N} e^l(t_i)$$

where $e^l(t_i) = \delta x^l(t_i) - \delta \hat{x}^l(t_i)$

$$X_{avg}(t_i) = \frac{1}{N_{avg} - 1} \sum_{l=1}^{N} [e^l(t_i) - e^{avg}(t_i)][e^l(t_i) - e^{avg}(t_i)]^T$$

$X_{avg}(t_i) \approx X(t_i)$ for all t_i

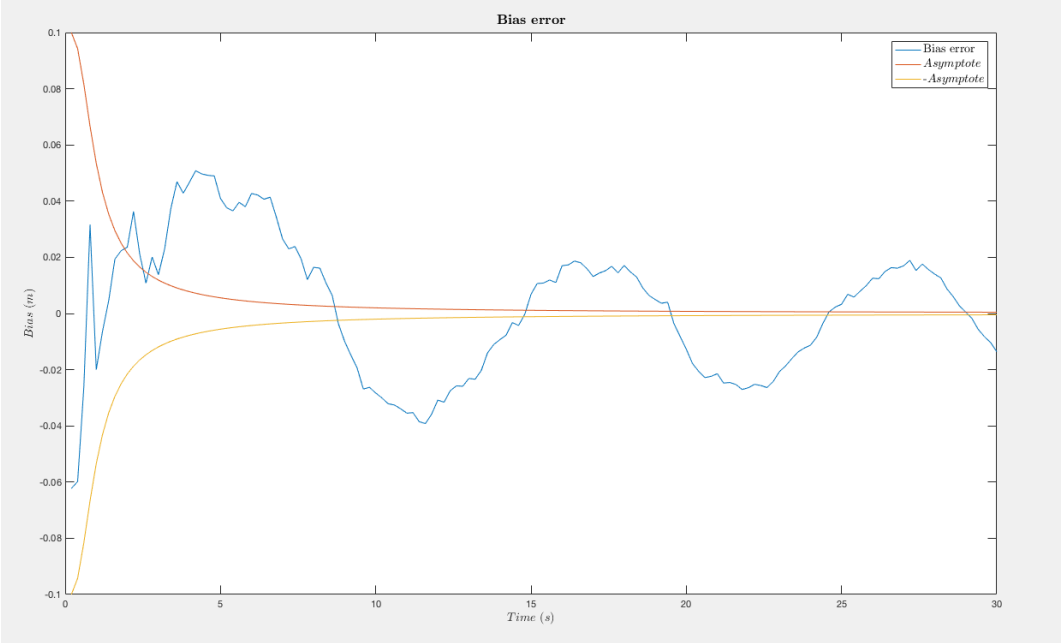

Figure 3.1: Bias Error.(Original)

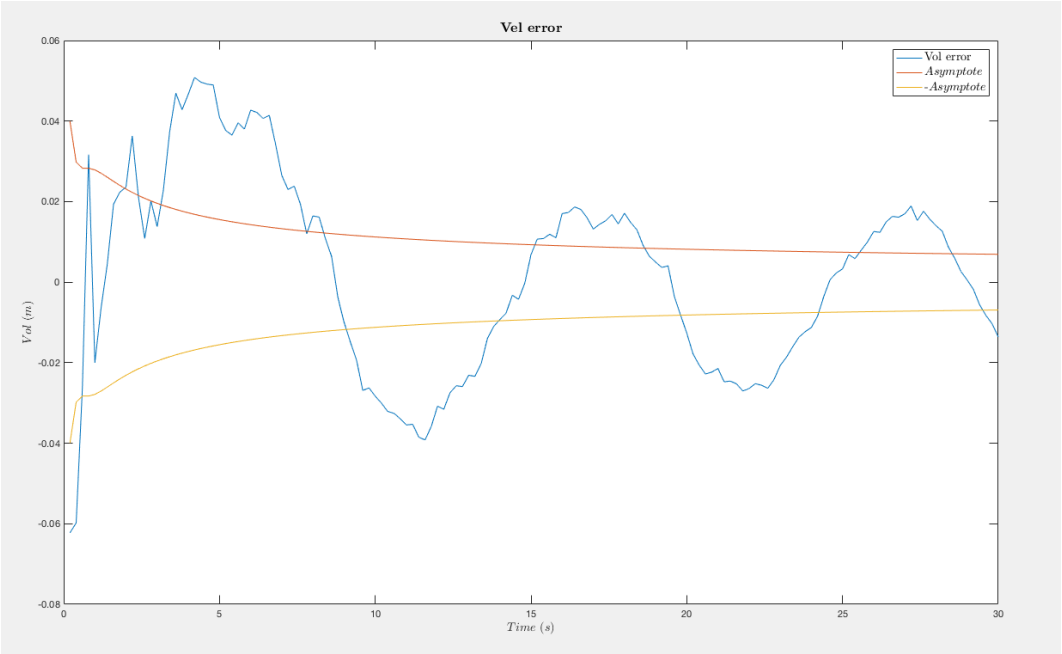

Figure 3.2: Vel Error.(Original)

However, we can see the above two figures are not simulated very well. After my consideration and analysis. The reason is N_{ave} that I choose is too small in the first time. Therefore, after my adjustment of the parameter of N_{ave} to the 6000 and above, I see the relative stable figures.[Error in the Kalman lteration estimates from a single run.]

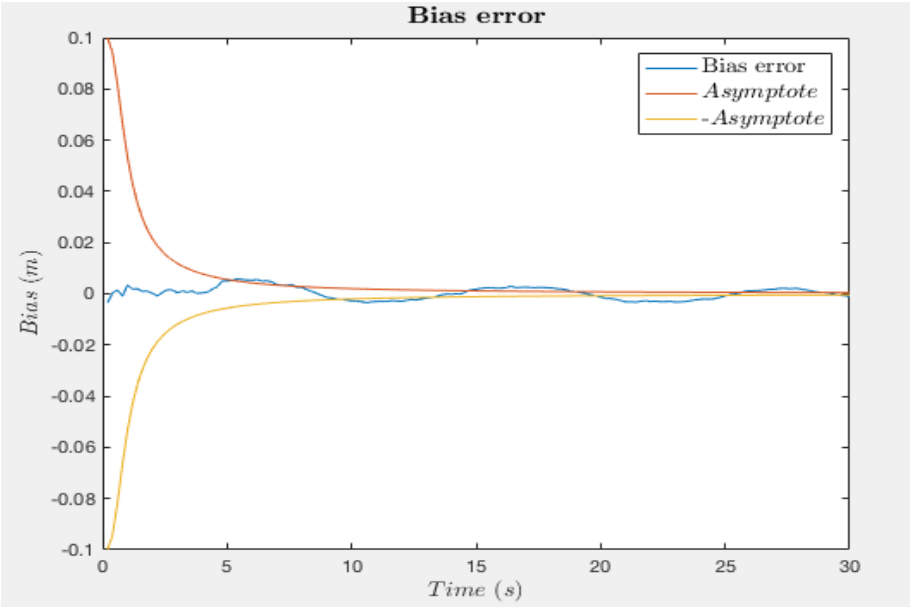

Figure 3.3: Bias Error.(After Adjusting)

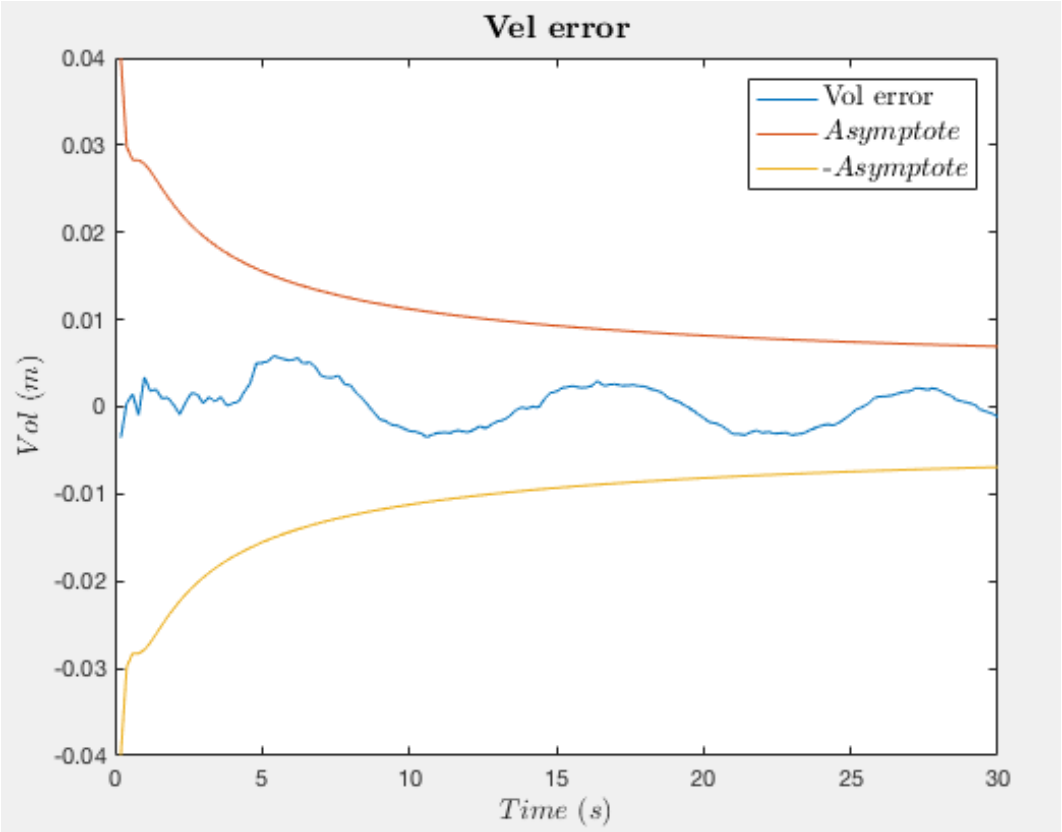

Figure 3.4: Vel Error.(After Adjusting)

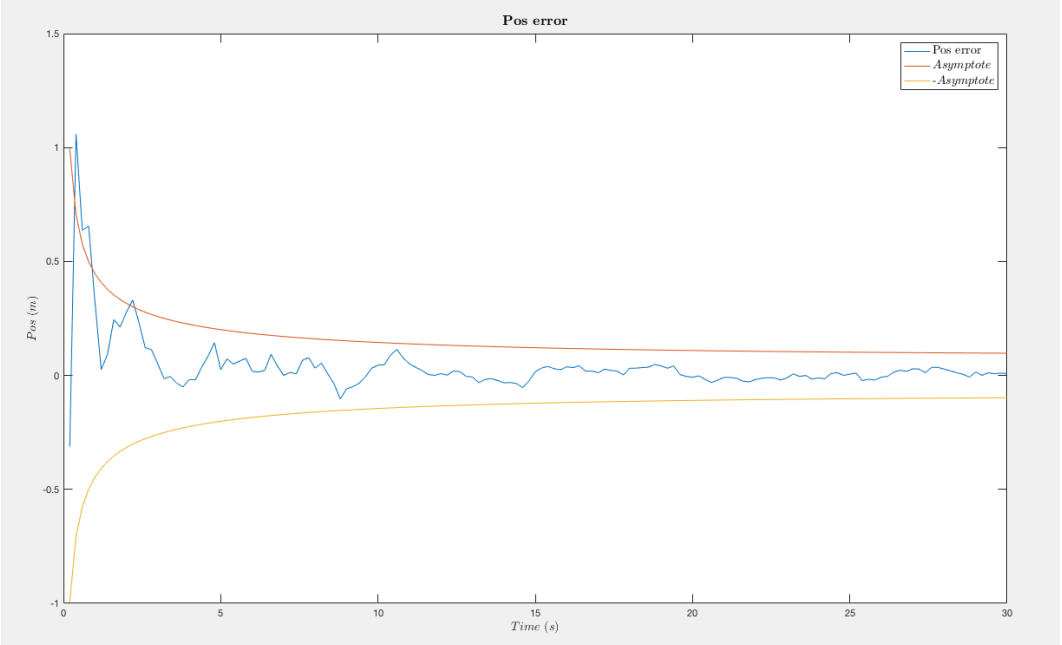

Figure 3.5: Pos Error.(After Adjusting)

The K matrix

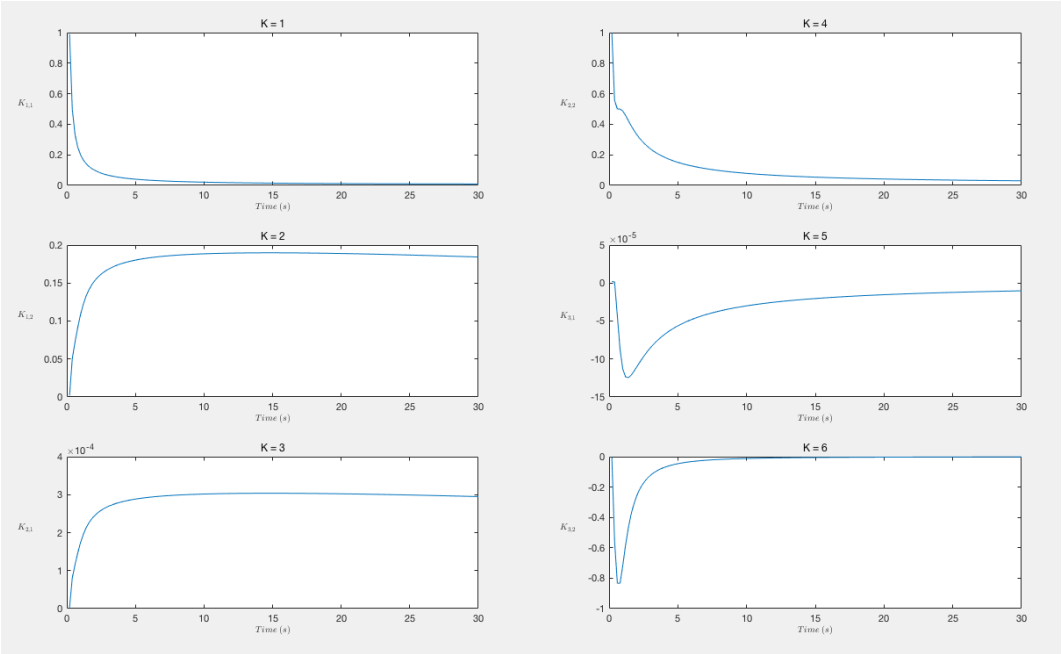

Figure 3.6: The K Matrix.

The P matrix

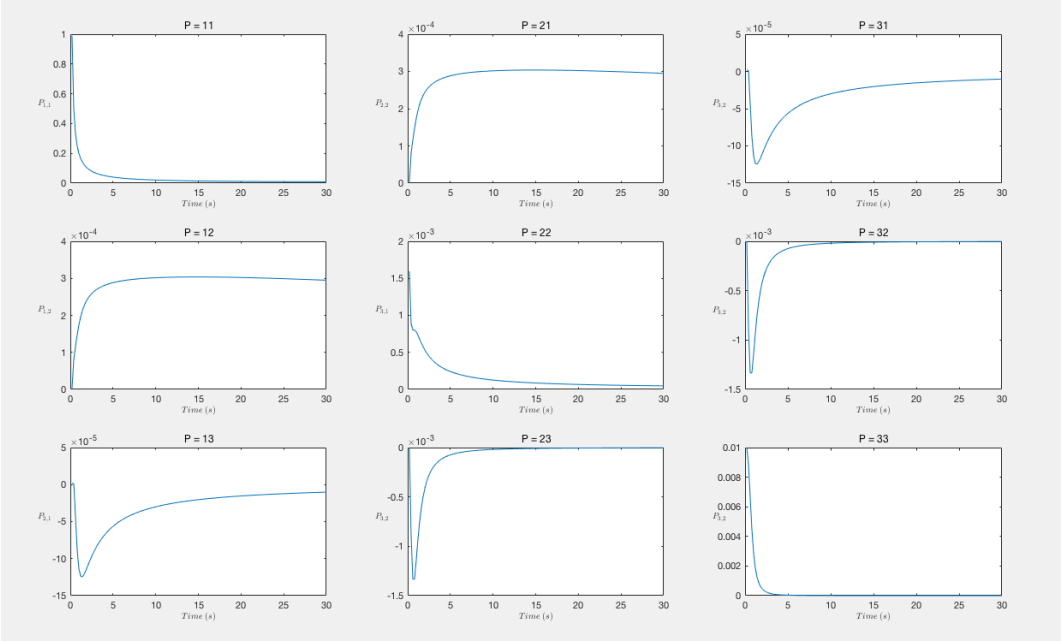

Figure 3.7: The P Matrix.

Difference between Ensemble Covariance and Average Propagated Covariance.

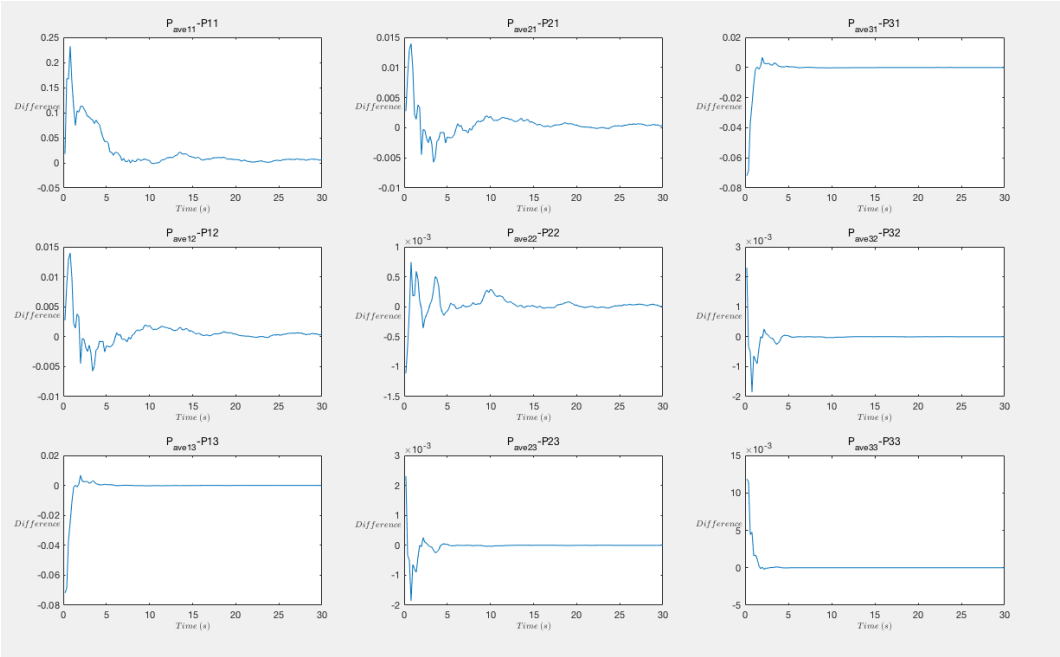

Figure 3.8: Difference between Ensemble Covariance and Average Propagated Covariance.

After the simulation and calculation, we get the above figures, including elements of Kalman gain matrix, the actual error variance P_{avg} and the error of them.

That is an important check to verify that the modeling is roughly correct and the Kalman Flter has been programmed correctly.The matrix $P_{ave}(t_i)$ should be close to $P(t_i)$ which is a posteriori error covariance matrix in the Kalman Filter algorithm for a single run. This can be expressed as: $P_{ave}(t_i) - P(ti) \approx 0$; for all $t_i \in [0, 30]$

3.2 Orthogonality

$Orth_{avg}(t_i) = \frac{1}{N}\sum_{l=1}^{N}[e^l(t_i) - e_{avg}(t_i)]\hat{x}(t_i)^T = 0$, for all t_i

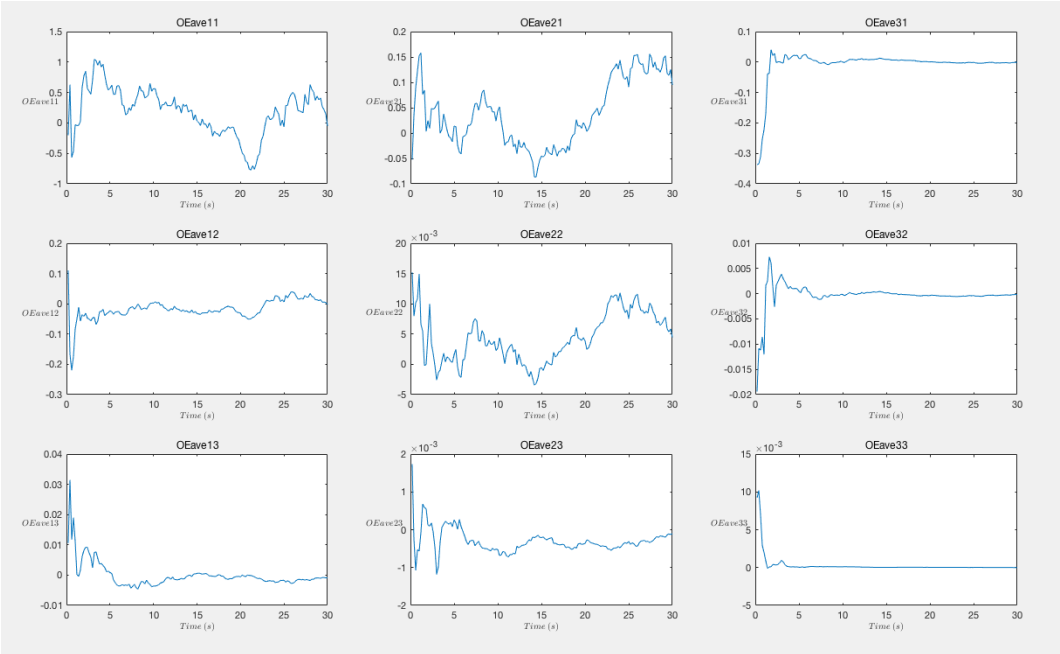

Figure 3.9: Orthogonality Check.

3.3 Independence of Residuals

$r^l(t_i) = \delta z^l(t_i) - \delta \bar{z}^l(t_i) = H\bar{e}^l(t_i) + \eta^l(t_i)$

$r_{avg}(t_i) = \frac{1}{N}\sum_{l=1}^{N} r^l(t_i) r^l(t_m)^T = 0 \ , \forall \ t_m < t_i$

After careful calculation and analysis, finally I get the res_{error}

$res_{error} =$

```
0.0912    0.0131
0.0128    0.0008
```

Result

As it can be seen above, the ensemble average for the correlation of residuals are not zero, but approximately near 0, which means there remains some flaws in the Kalman filter construction. However, based on the other checks, we can know that these flaws do not matter much, and the pure independence need the further analysis and calculation.

Chapter 4

Conclusion

After the comprehensive analysis and stimulation with different checks, we can easily see the results proved successful in matching the algorithm for the reality. From the figure 3.3, 3.4 and 3.5, we can see the Kalman Filter performs quite well in estimating the system and different states when the process experience 6000 or more realizations.

We can combine the measurements and dynamic model, which makes the Kalman Filter algorithm successfully estimate the position, velocity, and the accelerometer's bias. Figure 3.6 and figure 3.7 show the P matrix and Kalman gain matrix. Figure 3.8 shows the difference between Ensemble Covariance and Average Propagated Covariance. The error is small especially as time goes toward 30 seconds. Moreover, as we know the posteriori error values from the covariance matrix, which are added in figure 3.7 and 3.8, provides an error bound on the posteriori estimation at each time step t_i in the Kalman Filter.

To evaluate the performance of the Kalman Filter, three methods are used in my paper.The first one is a Monte Carlo simulation. From the results of the Monte Carlo simulation, we can roughly conclude the Kalman Filter is performing properly. We can firstly see the fact that mean of the estimate is close to zero for all times. In fact, as it was shown in figure 3.9, the error for each realization is just a difference, which is reasonably small and close to zero. Also, for a Kalman Filter to be performed correctly, the error conditional variance check, the orthogonality check, and the residual check should all approximately equal zero, which I have actually done in the modelling and calculation.

Appendix A

Bibliography

1.Stochastic Processes, Estimation and Control, Jason L. Speyer, Walter H. Chung, Society for Industrial and Applied Mathematics, 2008
2.https://www.gaussianwaves.com/2013/11/simulation-and-analysis-of-white-noise-in-matlab/, 2017/11/30
3.https://en.wikipedia.org/wiki/Kalmanfilter, 2017/11/28