MAE 271B Stochastic Estimation Project

Zhi Li 104944953

Mar 23rd, 2018

Abstract: This project completed estimation of relative position, velocity and acceleration of a missile with respect to a moving target. We developed Kalman Filter with reasonable assumptions to estimate the kinematics parameters of the missile. We showed that the Kalman Filter gives us reasonable results by using Monte Carlo analysis. Besides, we also considered complicated target dynamics, such as random telegraph signal acceleration and Kalman Filter also gives reasonable estimation of parameters of interests.

Introduction

Kalman Filter is one of the most important and common data fusion algorithms in use today. The great success of Kalman Filter is due to its small computational requirement, elegant recursive properties and its status as the optimal estimator for linear systems with Gaussian error statistics.

In this project, we implemented Kalman Filter to estimate the relative position, velocity and acceleration of a missile with respect to a moving target. The basic assumption of this system is that the acceration of the target acceleration is Gauss-Markov process. Therefore, the whole system is Gauss-Markov which Kalman Filter will give a optimal estimation for the relative position, velocity and acceleration of the missile. We also considered more realistic situation, the dynamics of target is described by random telegraph signal rather than Gauss-Markov. The perfoemance of Kalman Filter in this situation is also discussed.

Theory and Algorithm

Model

The missle intercept illustration is shown in Figure 1 below.

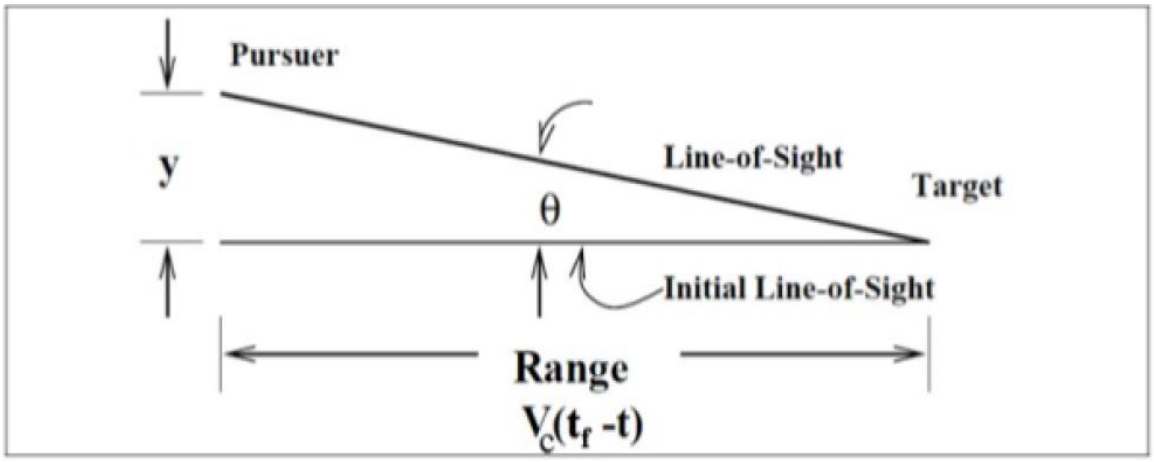

Figure 1. Missle Intercept Illustration.

The dynamics of the problem are

$$\dot{y} = v$$

$$\dot{v} = a_p - a_T$$

where a_p, the missile acceleration, is known and assumed to be zero. The input, a_T, is the target acceleration and is treated as a random forcing function with an exponential cerrelation,

$$E[a_T] = 0$$

$$E[a_T(t)a_T(s)] = E[a_T^2]e^{\frac{-|t-s|}{\tau}}$$

The scalar τ, is the correlation time. The statistics of initial lateral position $y(t_0)$ and initial lateral velocity $v(t_0)$ are:
$E[y(t_0)] = 0$, $E[y(t_0)^2] = 0$, $E[v(t_0)] = 0$, $E[y(t_0)v(t_0)] = 0$
The measurement z consist of a line-of-sight angle θ. For $|\theta| \ll 1$,

$$\theta \approx \frac{y}{V_c(t_f - t)}$$

It will also be assumed that z is corrupted by fading and scintillation noise so that

$$y = \theta + n$$

$$E[n(t)] = 0$$

$$E[n(t)n(\tau)] = V\delta(t-\tau) = [R_1 + \frac{R_2}{(t_f - t)^2}]\delta(t-\tau)$$

The process noise spectral density W is

$$W = GE[a_T^2]G^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & E[a_T^2] \end{bmatrix}$$

The parameters of the problem are $V_c = 300ft/sec$, $E[a_T^2] = [100ft.sec^{-2}]^2$, $t_f = 10sec$, $R_1 = 15 \times 10^{-6} rad^2.sec$, $R_2 = 1.67 \times 10^{-3} rad^2.sec^3$,$\tau = 2sec$.
The initial covariance is

$$P(0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & (200sec^{-1})^2 & 0 \\ 0 & 0 & (100sec^{-2})^2 \end{bmatrix}$$

System Dynamics

Based on the model and assumptions made before, we could write system dynamics in state-space form

$$\begin{bmatrix} \dot{y} \\ \dot{v} \\ \dot{a}_T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix} \begin{bmatrix} y \\ v \\ a_T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_{a_T} \tag{1}$$

The measurement z is

$$z = \begin{bmatrix} \frac{1}{V_c(t_f - t)} & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ v \\ a_T \end{bmatrix} + n \tag{2}$$

We can rewrite the above equations in the form of continuous-time stochastic linear sustem of differential equations:

$$dx(t) = F(t)x(t)dt + G(t)d\beta_t \tag{3}$$

$$dz = H(t)x(t)dt + d\eta_t \tag{4}$$

We could see that the matrices $F(t)$ and $G(t)$ are time-invariant while $H(t)$ is a time varying matrix. The statistics of $d\beta_t$ and $d\eta_t$ are

$$E[d\beta_t] = 0$$

$$E[d\eta_t] = 0$$

$$E[\omega(t)\omega t(\tau)] = E[\frac{\beta_t}{dt}\frac{\beta_\tau}{d\tau}] = \frac{W_{PSD}dt}{(dt)^2} = \frac{W_{PSD}}{dt}, t = \tau$$

$$E[n(t)n(\tau)] = E[\frac{\eta_t}{dt}\frac{\eta_\tau}{d\tau}] = \frac{V_{PSD}dt}{(dt)^2} = \frac{V_{PSD}}{dt}, t = \tau$$

$$W_{PSD} = E[a_T^2] = (100ft/s^2)^2$$

$$V_{PSD} = E[n(t)^2]dt = [R_1 + \frac{R_2}{(t_f - t)^2}]rad^2s$$

W_{PSD} is process noise PSD, V_{PSD} is measurement noise PSD. Note that W_{PSD} is time-invariant while V_{PSD} is time-varying.

Kalman Filter

Since we are using a continuous-time Kalman Filter, there is no separation between the propagation and measurement update stages.

The error covariance is obtained from the Riccati equation.

$$\dot{P(t)} = F(t)P(t) + P(t)F(t)^T - P(t)H(t)^TV(t)^{-1}H(t)P(t) + G(t)W(t)G(t)^T, P(0) = P_0 \quad (5)$$

To solve the above Riccati equation numerically, discrete-time mode is used to recursively solving P We assume $dt = 0.01sec$.

$$P(t + dt) = P(t) + dPdt, P(0) = P_0 \quad (6)$$

From the error covariance matrix, P(t), calculate from 6, the Kalman gains can be computed according to

$$K(t) = P(t)H(t)^TV(t)^{-1} \quad (7)$$

Simulation of Dynamics and Measurement

State

Since the dynamic eqaution and measurement equation are driven by stochastic random functions, the system dynamics and measurements model do change in each realization. Therefore, the error covariance and Kalman gain should be computed and stored on-line.

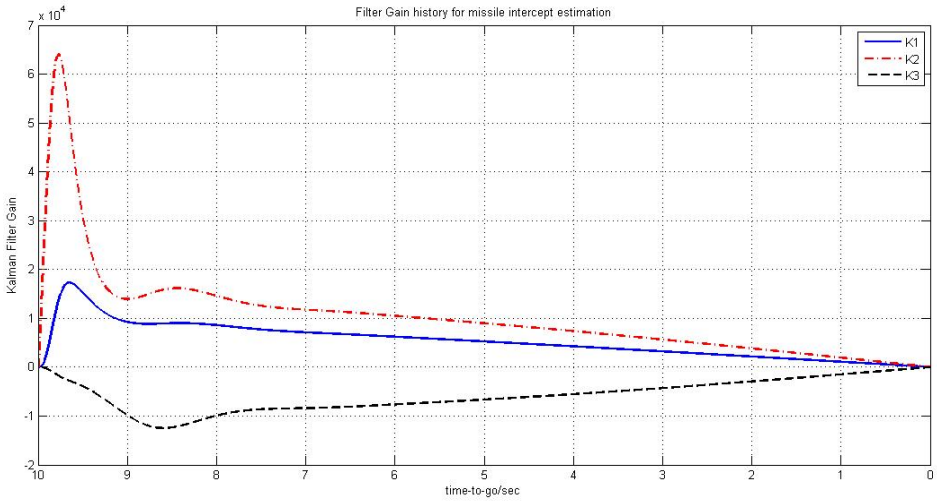

Figure 2. Kalman Filter gains.

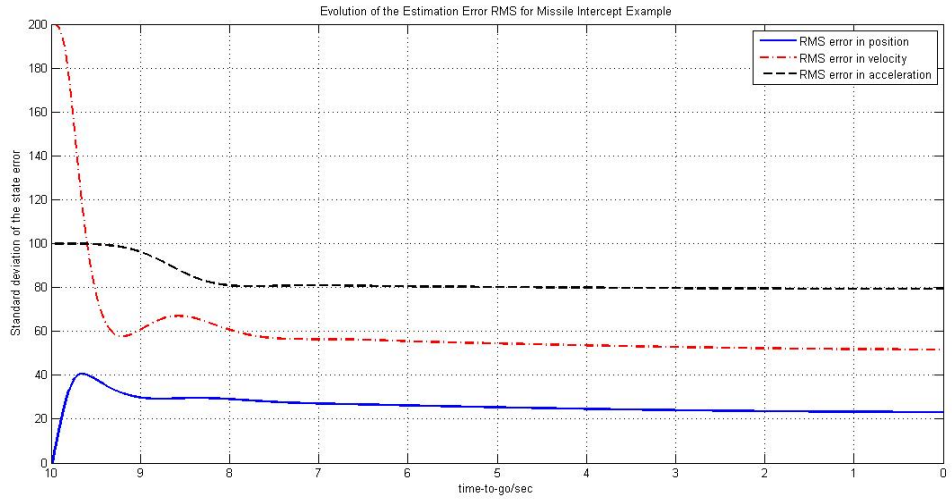

Figure 3. RMS of error in estimation of vertical position, velocity and acceleration of the missile.

For simulations of the dynamics and the measurements, the equation (1) and (2) are solved numerically.

$$\dot{x(t)} = F(t)x(t) + G(t)\frac{d\beta(t)}{dt}$$

$$x(t+1) = x(t) + \dot{x(t)}dt$$

$$x(0) = \begin{bmatrix} 0 \\ N(0,200) \\ N(0,100) \end{bmatrix}$$

The simulation results are as follows

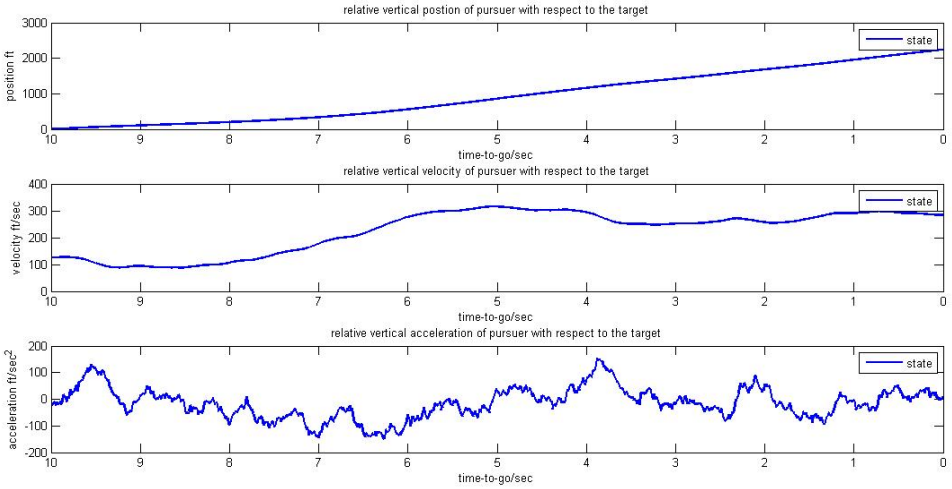

Figure 4. The relative position, velocity and acceleration of the missile for one realization

State Estimation

The state estimation also change in each realization. For this continuous Kalman Filter, there is no seperation between propagation and measurement update. Therefore the algorithm of state estimation is:

$$d\hat{x}(t) = F(t)\hat{x}(t)dt + K(t)[dz(t) - H(t)\hat{x}(t)dt] \tag{8}$$

We also discretisized the state estimation to solve numerically by matlab.

$$\dot{\hat{x}}(t) = F(t)\hat{x}(t)dt + K(t)[dz(t) - H(t)\hat{x}(t)dt]$$

$$\hat{x}(t+1) = \hat{x}(t) + \dot{\hat{x}}(t)dt$$

Figure 5 shows the results of the simulation for the estimation of states and states generated from dynamic equation.

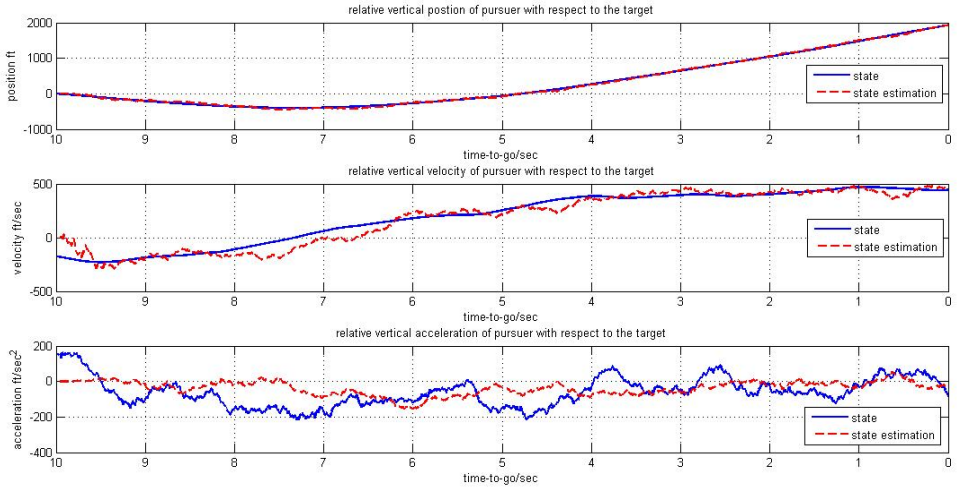

Figure 5. States estimation and States dynamics

Figure 6 shows the error between the state generated from dynamic equations and state estimation using Kalman Filter. The error is bounded by the RMS of the error calculated from the error covariance matrix(diagnal elements). We could see that over 67% of the errors are within the range of RMS of error.

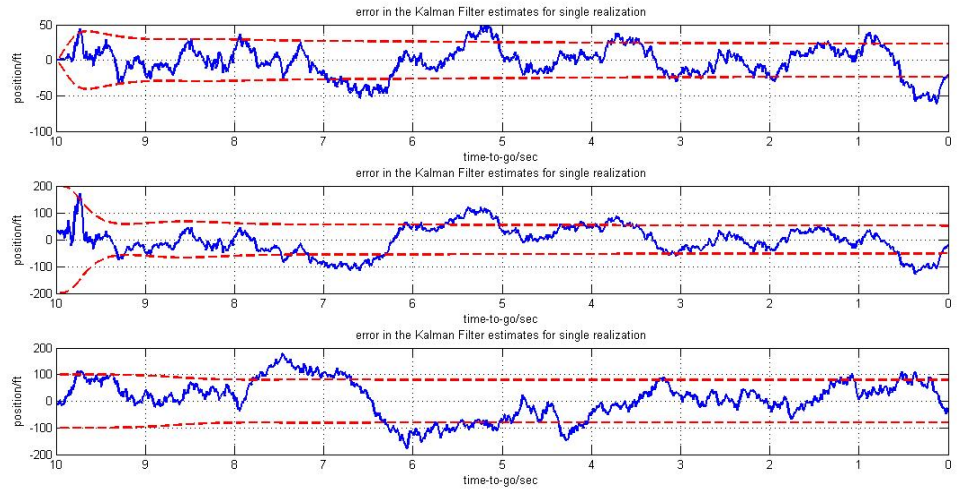

Figure 6. Error in the Kalman Filter estimation

Monte Carlo Simulation

There are basically two validation in this project by monte carlo. One is to show the error variance obtained from the simulation and averaged over an ensemble of realization is close to the error variance used in Kalman Filter, the other is to show theoretical orthogonality properties.

Error Variance

A Monte Carlo simulation is needed to find the ensemble averages over a set of realizations. Denote $e^l(t_i)$ as the actual error for realization l.

The ensemble average of $e^l(t_i)$ which produces the actual mean is

$$e^{ave}(t_i) = \frac{1}{N_{ave}} \sum_{l=1}^{N_{ave}} e^l(t_i) \tag{9}$$

where N is the number of realizations.

The expected value of $e^{ave}(t_i)$ is approximately 0 for all $t_i \in [0, T]$. Now, we consider the ensemble average producing the actual error variance P^{ave} as

$$P^{ave}(t_i) = \frac{1}{N_{ave} - 1} \sum_{l=1}^{N_{ave}} [e^l(t_i) - e^{ave}(t_i)][e^l(t_i) - e^{ave}(t_i)]^T \tag{10}$$

where $N_{ave} - 1$ is used for unbiased variance from small sample theory. The matrix $P^{ave}(t_i)$ (3×3) should be close to $P(t_i)$, i.e. $P^{ave}(t_i) - P(t_i) \approx 0$ for all t_i. Therefore, this is an important check of whether the Kalman Filter works correct.

Residual Orthogonality

Ensemble average for the correlation of the residuals:

$$residual = \frac{1}{N_{ave}} \sum_{l=1}^{N_{ave}} r^l(t_i) r^l(t_m)^T \tag{11}$$

where

$$r^l(t_i) = z^l(t_i) - H(t)\hat{x}^l(t) \tag{12}$$

Results of Monte Carlo Analysis

Verify $e_{ave} \approx 0$

Figure 7 shows the ensemble average of $e^l(t_i)$ which produces the actual mean e_{ave}

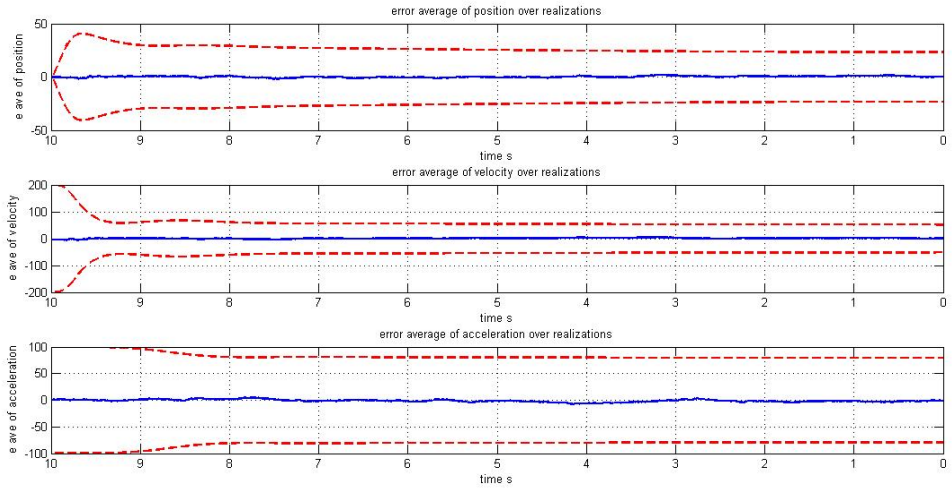

Figure 7. Ensemble average of errors

Verify $P^{ave}(t_i) - P(t_i) \approx 0$

Figure 8 shows the actual error variance from Monte Carlo simulation(in blue) and priori error variance in Kalman Filter(in red).

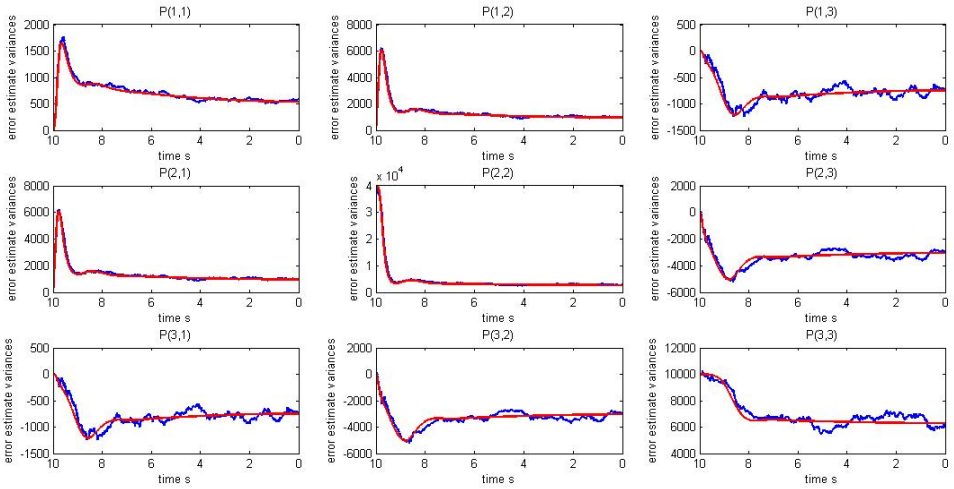

Figure 8. Actual Error Variance and Priori Error Variance

The diagonal elements of P matrix is the variance of position, velocity and acceleration, respectively. From Figure 8, we find that the variances are nearly the same between actual error variance and priori error variance. The Kalman Filter effectively gives estimation of kinematics of the missile.

Verify independence of residual

Pick $i = 9s$ and $m = 4s < i$. The resulting orthogonality check results:

$$\frac{1}{N_{ave}} \sum_{l=1}^{N_{ave}} r^l(t_i) r^l(t_m)^T \approx 9.1464 \times 10^{-5} \approx 0$$

This result shows the independence of residuals and also gives us a check that the Kalman Filter algorithm is basically correct.

Kalman Filter for Random Telegraph Signal

The Gauss-Markov model is an approximation to the random telegraph signal model that is more realistic. In this model the value a_T changes sign at random times given by a Poisson probability. We assume that $a_T(0) = \pm a_T$ with probability 0.5 and $a_T(t)$ changes polarity at Poisson times. The probability of k sign changes in a time interval of length T, $P(k(T))$ is $P(k(T)) = \frac{(\lambda T)^k e^{\lambda T}}{k!}$, where λ is the rate. The probability of an even number of sign changes in T, $P(even\#inT)$ is

$$P(even\#inT) = \sum_{k=0,keven}^{\infty} \frac{(\lambda T)^k e^{\lambda T}}{k!} = e^{-\lambda T} \sum_{k=0}^{\infty} \frac{(1+(-1)^k)(\lambda T)^k}{2k!} \tag{13}$$

Since $e^{\lambda T} = \sum_{k=0}^{\infty} \frac{(\lambda T)^k}{k!}$, then

$$P(even\#inT) = \frac{1}{2}[1 + e^{-2\lambda T}] \tag{14}$$

By a similar process, $P(odd\#inT) = \frac{1}{2}[1 - e^{-2\lambda T}]$. Then the probability that $a_T(t)$ is positive is

$$\begin{aligned} P(a_T(t) = a_T) &= P(a_T(t) = a_T | P(a_T(0) = a_T)) P(a_T(0) = a_T) \\ &+ P(a_T(t) = a_T | P(a_T(0) = -a_T)) P(a_T(0) = -a_T) \\ &= \frac{1}{2} P(even\#inT) + \frac{1}{2} P(odd\#inT) \\ &= \frac{1}{2}\{\frac{1}{2}[1 + e^{-2\lambda T}] + \frac{1}{2}[1 - e^{-2\lambda T}] = \frac{1}{2} \end{aligned} \tag{15}$$

Then the mean acceleration $\bar{a}_T = 0$. The autocorrelation $R_{a_T a_T}(t_1, t_2) = a_T^2 e^{-2\lambda|t_2 - t_1|}$.

If $\frac{1}{2} = 2\lambda$ then the autocorrelation of the Gauss-Markov process is the same as the random telegraph signal $(t_2 - t_1) = t - s$ and the means of both are zero.

We need a method of generating the random switching times. Let $T = t_{n+1} - t_n$ be the time between two switch times. The probability that the switch occured after t_{n+1} is

$$P(T' > T | t = t_n) = 1 - P(T' < T | t = t_n)$$

Now the probability that no switch occured in T, but occured after t_{n+1} is

$$P(T' > T | t = t_n) = P(\#of signchanges in T is zero) = e^{-\lambda T}$$

Then

$$P(T' \leq T | t = t_n) = 1 - e^{-\lambda T} \tag{16}$$

which is the probability that at least one change occured. To produce the random time t_{n+1}, set $P(T' \leq T | t = t_n)$ equal to U, the output of [0,1] from a uniform density function. Then

$$t_{n+1} = t_n - \frac{1}{\lambda} ln(U) \tag{17}$$

The objective of the above analysis is to use the random telegraph signal in the simulation rather than the Gauss-Markov process that was used to ensure that the Kalman Filter was implemented correctly. The results of the simulation for a sigle run using random telegraph signals for the acceleration and implementing the Kalman Filter for estimation the states are presented in Figure 9.

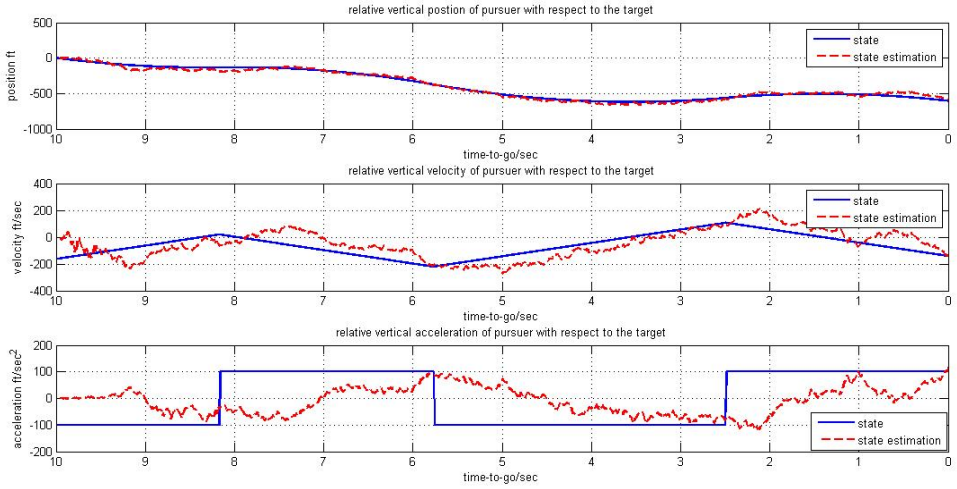

Figure 9. simulation of states generated from dynamics and states estimation of Kalman Filter for random telegraph signals

By running a Monte Carlo analysis using the random telegraph signal with $a_T = 100 ft.sec^{-2}$ and $\lambda = 0.25 sec^{-1}$ The ensemble average of error for 1000 realizations are shown in Figure 10. The actual error variance and priori error variance are shown in Figure 11.

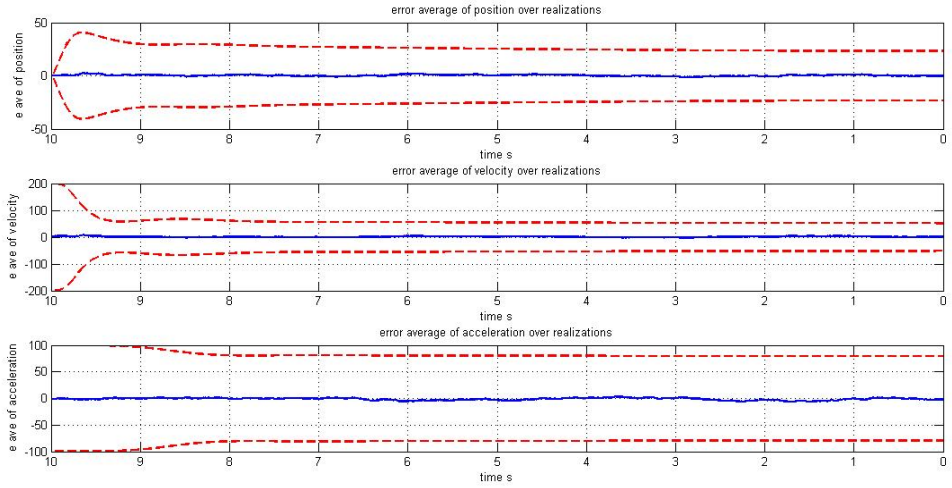

Figure 10. The ensemble average of error for 1000 realizations

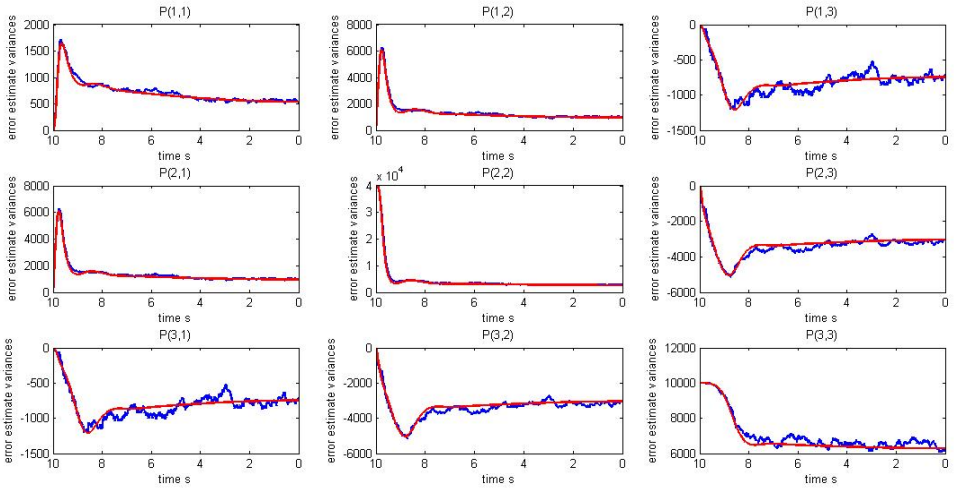

Figure 11. The actual error variance and priori error variance of Kalman Filter

Since $\lambda = 0.25sec^{-1} = \frac{1}{2\tau}$. The autocorrelation of the Gauss-Markov process is the same as the random telegraph signal.

The comparation of priori error variance of Kalman Filter, actual error variance of Gauss-Markov process and actual error variance of random telegraph signal are shown in Figure 12.

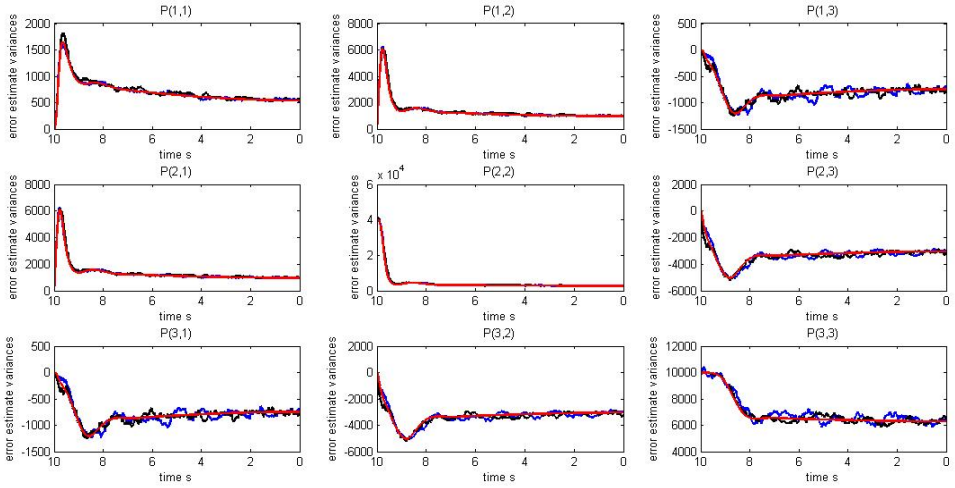

Figure 12. The actual error variance of Gauss-Markov process(blue) , actual error variance of random telegraph signal(black) and priori error variance of Kalman Filter(red)

Conclusion

The Kalman Filter gives reasonable estimation of relative position, velocity and acceleration of a missile with respect to a moving target. In order to further verify the algorithm, Monte Carlo analysis is used. Besides, I also checked the variances from Kalman Filter, actual variances and their differences and the independence of residuals. Both of them shows that the Kalman Filter algorithm is correct. We also consider more realistic situation that the target acceleration is replaced by random telegraph signals. The results also show that Kalman Filter was implemented correctly.

Appendix

Project271B.m

```
close all;
clear all;
clc;
rng('shuffle');


%% initials
tau=2;%sec
Vc=300;%300 ft/sec
tf=10;%sec
R1=15e-6;%rad^2.sec
R2=1.67e-3;%rad^2.sec^3

F=[0,1,0;0,0,-1;0,0,-1/tau];
B=[0;1;0];
G=[0;0;1];

W=[0,0,0;0,0,0;0,0,10000];

P0=[0,0,0;0,40000,0;0,0,10000];
P=P0;
K_=[];P_=P0;dt=0.01;

%% dynamic simulation and Kalman Filter
state0=[0;normrnd(0,200);normrnd(0,100)];
state=state0;state_=[state0];
s_estimate0=[0;0;0];
s_estimate=s_estimate0;s_estimate_=[s_estimate0];
error=[s_estimate0-state0];
residual=[];

for t=0:0.01:9.99
    H=[1/(Vc*(tf-t)),0,0];
    V=R1+R2/(tf-t)^2;

    d_P=F*P+P*F'-1/V*P*H'*H*P+W; %Riccati Equation
    K=P*H'*inv(V); %Kalman Gain
    K_=[K_ K];
    P=d_P*dt+P; % Propagate P
    P_=[P_ P];

    wat=normrnd(0,sqrt(10000/dt));%process noise
    n=normrnd(0,sqrt(V/dt));%measurement noise
    d_state=F*state+G*wat;%dynamic propagate
    z=H*state+n;%measurenment
    state=state+d_state*dt;%state update
    state_=[state_ state];

    d_s_estimate=F*s_estimate+K*(z-H*s_estimate);%estimation update
    residual=[residual z-H*s_estimate];%calculate the residual
    s_estimate=s_estimate+d_s_estimate*dt;%state estimation update
    s_estimate_=[s_estimate_ s_estimate];
    error=[error s_estimate-state];%store the errors
end

figure(1);
plot(K_(1,:),'b','Linewidth',2);hold on
plot(K_(2,:),'r-.','Linewidth',2);hold on;
plot(K_(3,:),'k--','Linewidth',2);grid on;
title('Filter Gain history for missile intercept estimation');
legend('K1','K2','K3');
xlabel('time-to-go/sec');ylabel('Kalman Filter Gain');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);

```

```
figure(2);
i=1:3:size(P_,2);
plot(sqrt(P_(1,i)),'b','Linewidth',2);hold on;
plot(sqrt(P_(2,i+1)),'r-.','Linewidth',2);hold on;
plot(sqrt(P_(3,i+2)),'k--','Linewidth',2);grid on;
title('Evolution of the Estimation Error RMS for Missile Intercept Example');
legend('RMS error in position','RMS error in velocity','RMS error in acceleration');
xlabel('time-to-go/sec');ylabel('Standard deviation of the state error');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);

figure(3);
subplot 311
plot(state_(1,:),'b','Linewidth',2);hold on;
plot(s_estimate_(1,:),'r--','Linewidth',2);grid on;
title('relative vertical postion of pursuer with respect to the target');
legend('state','state estimation');
xlabel('time-to-go/sec');ylabel('position ft');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 312
plot(state_(2,:),'b','Linewidth',2);hold on;
plot(s_estimate_(2,:),'r--','Linewidth',2);grid on;
title('relative vertical velocity of pursuer with respect to the target');
legend('state','state estimation');
xlabel('time-to-go/sec');ylabel('velocity ft/sec');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 313
plot(state_(3,:),'b','Linewidth',2);hold on;
plot(s_estimate_(3,:),'r--','Linewidth',2);grid on;
title('relative vertical acceleration of pursuer with respect to the target');
legend('state','state estimation');
xlabel('time-to-go/sec');ylabel('acceleration ft/sec^2');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);

figure(4);
subplot 311
plot(error(1,:),'LineWidth',2);hold on;
plot(sqrt(P_(1,i)),'r--','LineWidth',2);hold on;
plot(-sqrt(P_(1,i)),'r--','LineWidth',2);grid on;
xlabel('time-to-go/sec');ylabel('position/ft');
title('error in the Kalman Filter estimates for single realization');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 312
plot(error(2,:),'LineWidth',2);hold on;
plot(sqrt(P_(2,i+1)),'r--','LineWidth',2);hold on;
plot(-sqrt(P_(2,i+1)),'r--','LineWidth',2);grid on;
xlabel('time-to-go/sec');ylabel('position/ft');
title('error in the Kalman Filter estimates for single realization');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 313
plot(error(3,:),'LineWidth',2);hold on;
plot(sqrt(P_(3,i+2)),'r--','LineWidth',2);hold on;
plot(-sqrt(P_(3,i+2)),'r--','LineWidth',2);grid on;
xlabel('time-to-go/sec');ylabel('position/ft');
title('error in the Kalman Filter estimates for single realization');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
```

project271B5.m

```
close all;
clear all;
clc;
rng('shuffle');

%% initials
tau=2;%sec
Vc=300;%300 ft/sec
tf=10;%sec
R1=15e-6;%rad^2.sec
R2=1.67e-3;%rad^2.sec^3

F=[0,1,0;0,0,-1;0,0,-1/tau];
B=[0;1;0];
G=[0;0;1];

W=[0,0,0;0,0,0;0,0,10000];

P0=[0,0,0;0,40000,0;0,0,10000];
P=P0;
K_=[];P_=P0;dt=0.01;

%% dynamic simulation and Kalman Filter
    switchtime=0;stime=[];lambda=0.25;
    while switchtime≤9.99
        stime=[stime switchtime];
        switchtime=switchtime-1/lambda*log(1-unifrnd(0,1));
    end
    prob=unifrnd(0,1,1,length(stime)-1);

    if unifrnd(0,1)≤0.5
        at=100;
    else
        at=-100;
    end
    state0=[0;normrnd(0,200);at];
    state=state0;state_=[state0];
    s_estimate0=[0;0;0];
    s_estimate=s_estimate0;s_estimate_=[s_estimate0];
    error=[s_estimate0-state0];
    residual=[];
    index=2;temp=at;
    for t=0:0.01:9.99
        H=[1/(Vc*(tf-t)),0,0];
        V=R1+R2/(tf-t)^2;

        d_P=F*P+P*F'-1/V*P*H'*H*P+W; %Riccati Equation
        K=P*H'*inv(V); %Kalman Gain
        K_=[K_ K];
        P=d_P*dt+P; % Propagate P
        P_=[P_ P];

        wat=normrnd(0,sqrt(10000/dt));%process noise
        n=normrnd(0,sqrt(V/dt));%measurement noise

        if index≤length(stime)
            if t≤stime(index)%&&prob(index-1)≤0.5
                state(3)=at;
                temp=state(3);
            else
                index=index+1;
                at=-at;
                state(3)=at;
                temp=state(3);
            end

        end
```

```
        d_state=F*state+G*wat;%dynamic propagate
        z=H*state+n;%measurenment
        state=state+d_state*dt;%state update
        state(3)=temp;
        state_=[state_ state];

        d_s_estimate=F*s_estimate+K*(z-H*s_estimate);%estimation update
        residual=[residual z-H*s_estimate];%calculate the residual
        s_estimate=s_estimate+d_s_estimate*dt;%state estimation update
        s_estimate_=[s_estimate_ s_estimate];
        error=[error s_estimate-state];%store the errors
    end

figure(3);
subplot 311
plot(state_(1,:),'b','Linewidth',2);hold on;
plot(s_estimate_(1,:),'r--','Linewidth',2);grid on;
title('relative vertical postion of pursuer with respect to the target');
legend('state','state estimation');
xlabel('time-to-go/sec');ylabel('position ft');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 312
plot(state_(2,:),'b','Linewidth',2);hold on;
plot(s_estimate_(2,:),'r--','Linewidth',2);grid on;
title('relative vertical velocity of pursuer with respect to the target');
legend('state','state estimation');
xlabel('time-to-go/sec');ylabel('velocity ft/sec');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 313
plot(state_(3,:),'b','Linewidth',2);hold on;
plot(s_estimate_(3,:),'r--','Linewidth',2);grid on;
title('relative vertical acceleration of pursuer with respect to the target');
legend('state','state estimation');
xlabel('time-to-go/sec');ylabel('acceleration ft/sec^2');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
```

Verification271B.m

```
Num=1000;
for i=1:Num
    project271B;%for problem 1-4
    project271B_5;%for problem 5
    if i==1 %define useful variables
    error_ave=zeros(3,length(error));
    error_store=zeros(3,length(error),Num);
    residual_store=zeros(1,length(residual),Num);
    var_sum=zeros(3,3);
    end
    error_ave=error_ave+error; % compute error sum
    error_store(:,:,i)=error;
    residual_store(:,:,i)=residual;
end
error_ave=error_ave/Num;  %compute e_ave

%% compute P_ave ======================
P_ave=[];
for tt=1:size(P_,2)/3
    for i=1:Num
        term=error_store(:,tt,i)-error_ave(:,tt);
        sum_p=term*term';
        var_sum=var_sum+sum_p;%3x3
    end
    var_sum=var_sum/(Num-1);
    P_ave=[P_ave var_sum];
    var_sum=zeros(3,3);
end
%% =======================================

%% residual orthogonality ==================
sum=0;
k=101;j=689;
        for i=1:Num
            sum=sum+residual_store(:,k,i)*residual_store(:,j,i)';
        end
sum=sum/Num;
%% ========================================

figure(5);
i=1:3:size(P_,2);
subplot 311
plot(error_ave(1,:),'linewidth',2);hold on;
plot(sqrt(P_(1,i)),'r--','linewidth',2);hold on;
plot(-sqrt(P_(1,i)),'r--','linewidth',2);grid on;
title('error average of position over realizations')
xlabel('time s');ylabel('e ave of position');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 312
plot(error_ave(2,:),'linewidth',2);hold on;
plot(sqrt(P_(2,i+1)),'r--','linewidth',2);hold on;
plot(-sqrt(P_(2,i+1)),'r--','linewidth',2);grid on;
title('error average of velocity over realizations')
xlabel('time s');ylabel('e ave of velocity');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);
subplot 313
plot(error_ave(3,:),'linewidth',2);hold on;
plot(sqrt(P_(3,i+2)),'r--','linewidth',2);hold on;
plot(-sqrt(P_(3,i+2)),'r--','linewidth',2);grid on;
title('error average of acceleration over realizations')
xlabel('time s');ylabel('e ave of acceleration');
set(gca,'xticklabel',[10 9 8 7 6 5 4 3 2 1 0]);
xlim([0,1000]);


```

```
%% P variace and actual variance
figure(8);
ind=1:3:length(P_);
legend('Actual Error Variance from Monto Carlo Simulation','Priori Error Variance in ...
    Kalman Filter')
title('Variance Check for 1000 realizations for Pave and P');
subplot 331
plot(P_ave(1,ind),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(1,ind),'r','linewidth',2); %% estimate variance of one realization
title('P(1,1)');
xlabel('time s');ylabel('error estimate variances');
set(gca,'xticklabel',[10 8 6 4 2 0]);
xlim([0,1000]);


subplot 332
plot(P_ave(1,ind+1),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(1,ind+1),'r','linewidth',2); %% estimate variance of one realization
title('P(1,2)');
xlabel('time s');ylabel('error estimate variances');
set(gca,'xticklabel',[10 8 6 4 2 0]);
xlim([0,1000]);


subplot 333
plot(P_ave(1,ind+2),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(1,ind+2),'r','linewidth',2); %% estimate variance of one realization
title('P(1,3)');
xlabel('time s');ylabel('error estimate variances');
set(gca,'xticklabel',[10 8 6 4 2 0]);
xlim([0,1000]);


subplot 334
plot(P_ave(2,ind),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(2,ind),'r','linewidth',2); %% estimate variance of one realization
title('P(2,1)');
xlabel('time s');ylabel('error estimate variances');
set(gca,'xticklabel',[10 8 6 4 2 0]);
xlim([0,1000]);


subplot 335
plot(P_ave(2,ind+1),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(2,ind+1),'r','linewidth',2); %% estimate variance of one realization
title('P(2,2)');
xlabel('time s');ylabel('error estimate variances');
set(gca,'xticklabel',[10 8 6 4 2 0]);
xlim([0,1000]);


subplot 336
plot(P_ave(2,ind+2),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(2,ind+2),'r','linewidth',2); %% estimate variance of one realization
title('P(2,3)');
xlabel('time s');ylabel('error estimate variances');
set(gca,'xticklabel',[10 8 6 4 2 0]);
xlim([0,1000]);

subplot 337
plot(P_ave(3,ind),'k','linewidth',2);hold on; %%simulation and the average over an ...
    ensemble of realizations
plot(P_(3,ind),'r','linewidth',2); %% estimate variance of one realization
title('P(3,1)');
```

```
130 xlabel('time s');ylabel('error estimate variances');
131 set(gca,'xticklabel',[10 8 6 4 2 0]);
132 xlim([0,1000]);
133
134 subplot 338
135 plot(P_ave(3,ind+1),'k','linewidth',2);hold on; %%simulation and the average over an ...
        ensemble of realizations
136 plot(P_(3,ind+1),'r','linewidth',2); %% estimate variance of one realization
137 title('P(3,2)');
138 xlabel('time s');ylabel('error estimate variances');
139 set(gca,'xticklabel',[10 8 6 4 2 0]);
140 xlim([0,1000]);
141
142 subplot 339
143 plot(P_ave(3,ind+2),'k','linewidth',2);hold on; %%simulation and the average over an ...
        ensemble of realizations
144 plot(P_(3,ind+2),'r','linewidth',2); %% estimate variance of one realization
145 title('P(3,3)');
146 xlabel('time s');ylabel('error estimate variances');
147 set(gca,'xticklabel',[10 8 6 4 2 0]);
148 xlim([0,1000]);
```