**MAE 593I**

Homework 3

Ryan Watson

# Problem 1

*Design and implement a Kalman filter to process dataSet3 and dataSet4 that were used in homework #2.*

For this homework assignment I decided to use and Unscented Kalman Filter (UKF). This decision was made for several reasons. The first reason that I decided to use a UKF was because I would like to become more familiar with the implementation and this was a great exercise to practice with. The second and more relevant reason is because a UKF allows for the use of non-linear equations in the model. That means that there is no need to calculate a Jacobian Matrix.

* ***What should be the process model***

The process model used for this UKF can be seen below. As can be seen, for data sets 3 and 4 the position is modeled as a random walk event. The clock bias was modeled as the bias plus the drift. This can be seen in Figure 1. The entire code for problem 1 can be seen in Appendix A.

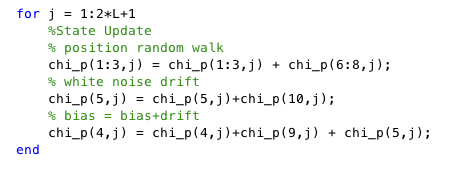
******

Figure 1: Matlab Section

* **Process Noise And Measurement Noise Covariance**

After experiment with several process and measurement noise covariance values I decided to select Q=I\*0.01 and R=I\*2. These values were selected through a trial and error process. By having a small Q value and a relatively large R value I am telling the Kalman Filter to way more heavily my process model than my measurements.

* **Atmospheric Delays**
  + **Ionosphere**

To mitigate the Ionospheric errors I used the dual frequency Ionospheric free pseudorange combination. The equation for this combination is shown below.

Eq. 1

* + **Tropospheric**

To model the troposphere I used several mapping. The mapping function that worked best for me can be seen below as Eq. 2.

M(el) = 2.42

Eq. 2

## Data Set 3

In Figure 2 an error plot for data set 3 can be seen. From Table 1 it can be seen that the largest error average error was in the “Up” direction with a magnitude of roughly 2.2 meters. This is reasonable because satellite geometry makes it so that height is the hardest axes to precisely measure with GPS.

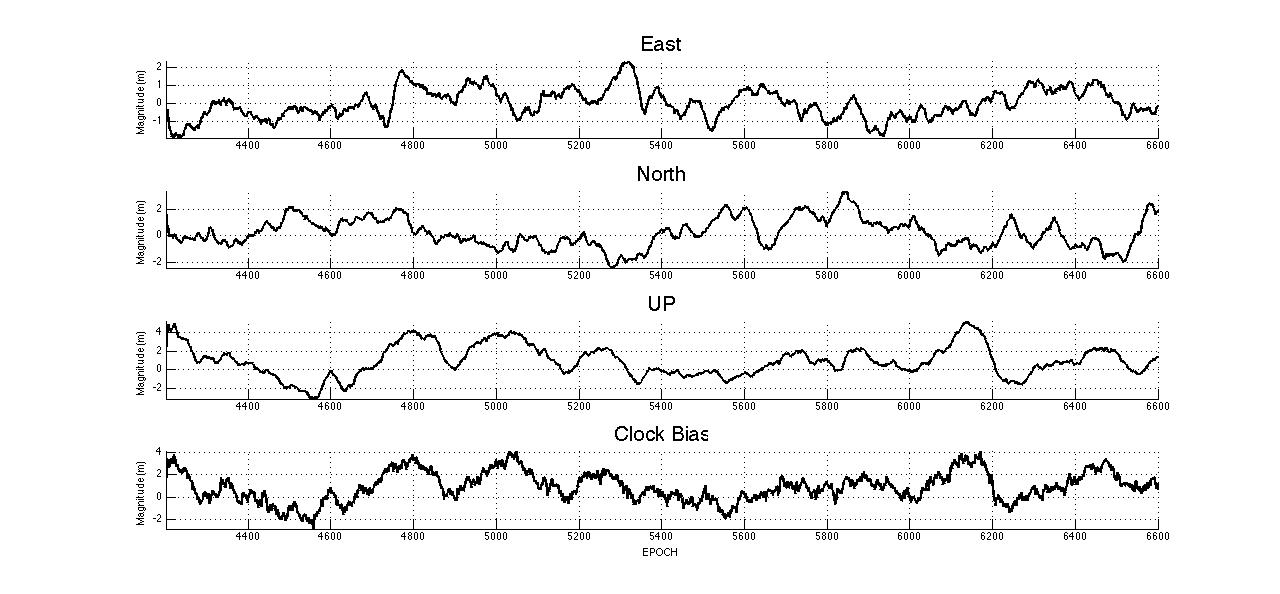


Figure 2: Error Plot Data Set 3

Table 1: Error Values for Data Set 3

|  |  |  |  |
| --- | --- | --- | --- |
| East (m) | North (m) | Up (m) | Clock (m) |
|
| 0.084 | -0.205 | 2.19 | 1.86 |
|

## Data Set 4

Unlike Data Set 3, for Data Set 4 the largest error is the clock. This can be seen in Table 2 where the average error magnitudes are presented. Like in Data Set 3, the errors in the East and North direction are on the centimeter level.

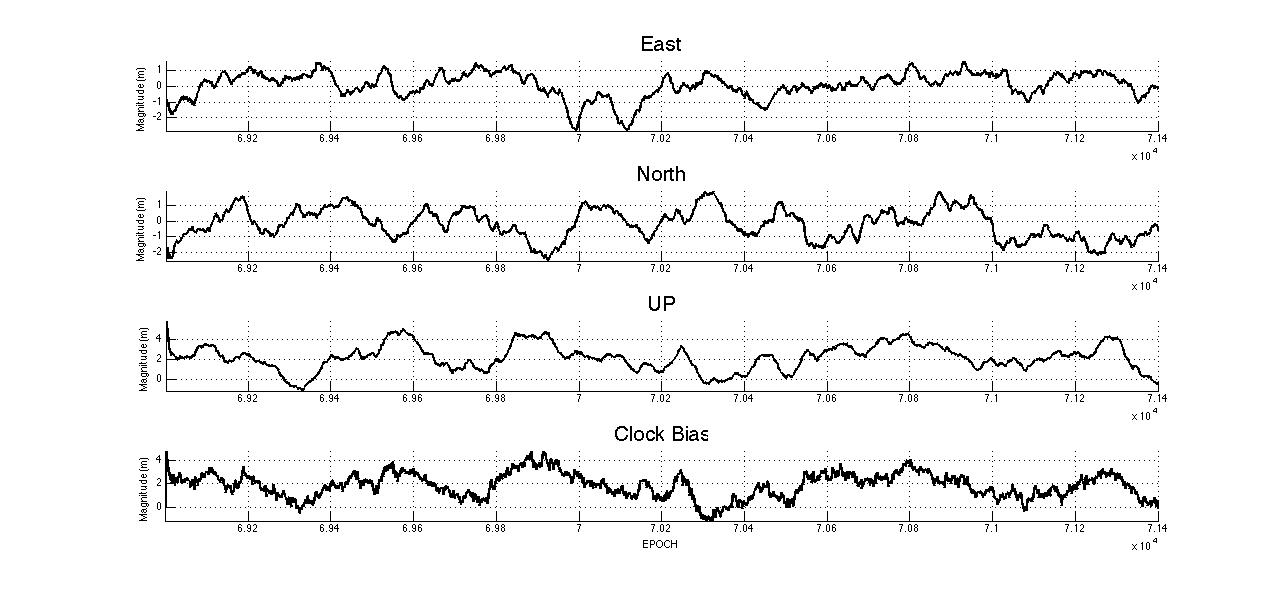


Figure 3: Error Plot for Data Set 4

Table 2: Error Values For Data Set 4

|  |  |  |  |
| --- | --- | --- | --- |
| East (m) | North (m) | Up (m) | Clock (m) |
|
| 0.022 | 0.199 | 0.923 | 1.86 |
|

# Problem 2

*Use the carrier phase data to smooth pseudorange using the approach discussed in class. Assume that M=100.*

Carrier Phase Smoothing (CSC) is a valuable tool that allows for the noisy range data to be “smoothed” with the more precise carrier data. The implementation of carrier smoothing is fairly simple, all that needs to be done is to apply Eq. 3 to the data.

= +[+(-)]

Eq. 3

One major draw back to the implementation of carrier phase smoothing is that the algorithm must be initialized every time that a carrier phase cycle slip occurs.

The smoothed data can be seen in Figure 4. It can easily been seen from the plot that the smoothed data gives a much cleaner plot. This would be very valuable if you are planning on using the data in a filter where large data spikes could cause issues elsewhere.

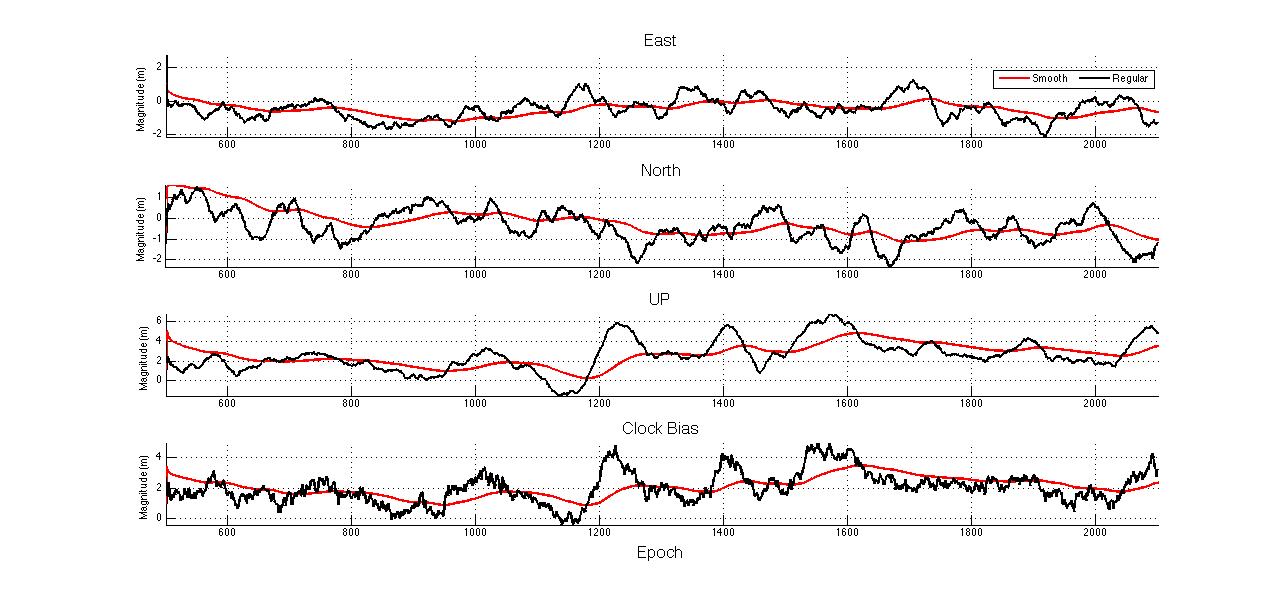


Figure 4: Smoothed vs Non-Smoothed Data

# Supplemental Questions:

1. *What error sources are being mitigated in problem 2 when carrier smoothing pseudorange?*

By applying carrier smoothing you are reducing the effects of multipath error. Also because of the utilization of the dual frequency data the Ionosphere error was mitigated to the first order.

1. *What are the assumption regarding atmospheric delays needed to use pseudorange-rate observables for velocity estimation and to use carrier-smoothed pseudorange?*

We make the assumption that the errors due to changes during the measurement interval in the ionosphere and tropospheric delays are generally small. This leaves the largest error in estimating velocity using the Doppler measurement as the satellites velocity. This is because any error in the satellites predicted position will show up in the velocity estimate.

*Given that we have dual frequency observations, devise a strategy that will help you determine if the assumptions needed to use these two estimation strategies are valid.*

To check the assumption that the change over the time interval for the ionosphere and tropospheric delays are small, I would calculate the error for each epoch. I would then do an epoch-by-epoch comparison to see if there are any points in the data that do not conform to the assumption.

# Appendix

% 5-State GPS Unscented Kalman Filter

%

% The state vector is:

% x1 = delta x position

% x2 = delta y position

% x3 = delta z position

% x4 = delta clock offset

% x5 = delta clock drift

% Authors: Ryan M. Watson,

clear all

%close all

load dataSet5.mat % static data

nom1=ones(1,length(satsXYZ));

nom1=truthXYZ(1,1).\*nom1;

nom2=ones(1,length(satsXYZ));

nom2=truthXYZ(2,1).\*nom2;

nom3=ones(1,length(satsXYZ));

nom3=truthXYZ(3,1).\*nom3;

nomXYZ=[nom1;nom2;nom3];

prData=(2.546\*prDataP1)-(1.546\*prDataP2);

Ntot=length(prData);

%UKF Parameters and Weights

L = 10; % Size of state vector

alpha = 1; % Primary scaling parameter

beta = 2; % Secondary scaling parameter (Gaussian assumption)

kappa = 0; % Tertiary scaling parameter

lambda = alpha^2\*(L+kappa) - L;

wm = ones(2\*L + 1,1)\*1/(2\*(L+lambda));

wc = wm;

wm(1) = lambda/(lambda+L);

wc(1) = lambda/(lambda+L) + 1 - alpha^2 + beta;

speedoflight = 299792458; % Speed of Light (m/s)

Q=[.01; .01; .01; .01; .01];

Q = diag(Q);

sigmaz = 2; % standard deviation of pseudorange measurement

% noise in meters

x\_pre\_ukf = zeros(10,Ntot); % initial state prediction

P\_pre = 50\*eye(5); % initial prediction error covariance matrix

P=P\_pre;

x\_ukf=x\_pre\_ukf;

Ntot=length(prData);

for i = 1:Ntot

if i==1

estusr = olspos(prData(1:nSat(i),i),satsXYZ(1:nSat(i),:,i));

% disp(estusr)

x\_ukf(1,1)=estusr(1);

x\_ukf(2,1)=estusr(2);

x\_ukf(3,1)=estusr(3);

x\_ukf(4,1)=estusr(4);

x\_ukf(5,1)=0;

else

R = sigmaz\*eye(nSat(i)); % form the measurement error covariance matrix

[sP,dummy] = chol(P,'lower'); % Calculate square root of error covariance

sQ = sqrtm(Q);

sPQ = blkdiag(sP,sQ);

% chi\_p = "chi previous" = chi(k-1)

chi\_p = [x\_ukf(:,i-1), x\_ukf(:,i-1)\*ones(1,L)+sqrt(L+lambda)\*sPQ,x\_ukf(:,i-1)\*ones(1,L)-sqrt(L+lambda)\*sPQ];

for j = 1:2\*L+1

%State Update

% position random walk

chi\_p(1:3,j) = chi\_p(1:3,j) + chi\_p(6:8,j);

% white noise drift

chi\_p(5,j) = chi\_p(5,j)+chi\_p(10,j);

% bias = bias+drift

chi\_p(4,j) = chi\_p(4,j)+chi\_p(9,j) + chi\_p(5,j);

end

x\_ukf(1:5,i) = chi\_p(1:5,:)\*wm; % Calculate mean of predicted state

% Calculate covariance of predicted stats

P\_m = zeros(5,5);

for k = 1:2\*L+1

P\_m = P\_m + wc(k)\*(chi\_p(1:5,k) - x\_ukf(1:5,i))\*(chi\_p(1:5,k) - x\_ukf(1:5,i))';

end

L\_1 = 5+nSat(i); % Size of state vector

lambda\_1 = alpha^2\*(L\_1+kappa) - L\_1;

wm\_1 = ones(2\*L\_1 + 1,1)\*1/(2\*(L\_1+lambda\_1));

wc\_1 = wm\_1;

wm\_1(1) = lambda\_1/(lambda\_1+L\_1);

wc\_1(1) = lambda\_1/(lambda\_1+L\_1) + 1 - alpha^2 + beta;

[sP\_m, dum] = chol(P\_m,'lower'); % Calculate square root of error covariance

sR = sqrtm(R);

sPR = blkdiag(sP\_m,sR);

x\_m = zeros(L\_1,1);

x\_m(1:5,1)=x\_ukf(1:5,i);

chi\_m = [x\_m, x\_m\*ones(1,L\_1)+sqrt(L\_1+lambda\_1)\*sPR,x\_m\*ones(1,L\_1)-sqrt(L\_1+lambda\_1)\*sPR];

for j=1:nSat(i)

TropoDel(j,i)= tropocorr(satsXYZ(j,:,i),x\_ukf(1:3,i));

end

Y\_sig = zeros(nSat(i),1+2\*L\_1);

for k = 1:2\*L\_1+1

for l=1:nSat(i)

Y\_sig(l,k) = norm(satsXYZ(l,:,i) - chi\_m(1:3,k)')+chi\_m(4,k)+chi\_m(5+l,k)+TropoDel(l,i);

end

end

y\_m(1:nSat(i),1) = Y\_sig(1:nSat(i),:)\*wm\_1;

Pyy = zeros(nSat(i));

for k = 1:2\*L\_1+1

Pyy = Pyy + wc\_1(k)\*(Y\_sig(1:nSat(i),k) - y\_m(1:nSat(i),1))\*(Y\_sig(1:nSat(i),k) - y\_m(1:nSat(i),1))';

end

Pxy = zeros(5,nSat(i));

for k = 1:2\*L\_1+1

Pxy = Pxy + wc\_1(k)\*(chi\_m(1:5,k)-x\_m(1:5,1))\*(Y\_sig(1:nSat(i),k) - y\_m(1:nSat(i),1))';

end

K = Pxy/Pyy;

prz=prData(1:nSat(i),i);

x\_ukf(1:5,i) = x\_m(1:5,1)+K\*(prz(1:nSat(i))-y\_m(1:nSat(i)));

P = P\_m - K\*Pyy\*K';

end

end

for i = 1:length(satsXYZ)

ENU\_UKF = xyz2enu(x\_ukf(1:3,i),nomXYZ(:,i));

ENU\_Truth = xyz2enu(truthXYZ(:,i),nomXYZ(:,i));

ERROR(:,i)=ENU\_UKF-ENU\_Truth;

Clock = x\_ukf(4,:)-truthClockBias;

end

Clock(1,1)=0; ERROR(3,1)=0;

Mean\_Error\_East=mean(ERROR(1,:))

Mean\_Error\_North=mean(ERROR(2,:))

Mean\_Error\_UP=mean(ERROR(3,:))

Mean\_Clock\_Error=mean(Clock)

subplot(4,1,1)

plot(time,ERROR(1,:),'LineWidth',2, 'Color', [0,0,0])

title('East')

axis tight

subplot(4,1,2)

plot(time,ERROR(2,:),'LineWidth',2, 'Color', [0,0,0])

title('North')

axis tight

subplot(4,1,3)

plot(time,ERROR(3,:),'LineWidth',2, 'Color', [0,0,0])

title('UP')

axis tight

subplot(4,1,4)

plot(time,Clock,'LineWidth',2, 'Color', [0,0,0])

title('Clock Bias')

axis tight