# Kalman Filter Design

The Kalman filter (KF) that was designed is attached in Appendix A. When running the filter, 3, yes or no, user inputs are required. Those inputs are if the user wants to include pseudorange rate to estimate velocity, if the user wants to include carrier phase smoothing, and if the user would like to include troposphere delay.

The troposphere zenith delay is modeled using the Saastamoinen model, whose input parameters are hardcoded. The zenith delay is mapped using a function found on [*http://www.navipedia.net/index.php/Tropospheric\_Delay*](http://www.navipedia.net/index.php/Tropospheric_Delay) *.* It was decided to use dual-frequency ionosphere free pseudorange data for position estimation. Using ionosphere free data simplifies the filter (less functions and inputs), but it was also chosen due to prior experience with a LLS estimator, where it provided more accurate position estimation than the Klobuchar model.

Appendix B features the Carrier Phase Smoothing Function, that is called in the KF. It uses ionosphere free pseudorange, and phase data, to generate a smoothed pseudorange The coefficient M is hardcoded to be 100 for this filter.

## State Vector (X)

The state vector is initialized as a zero vector, as it is assumed there is no prior knowledge of the initial position. The length of the state vector is dependent on if the user decided to estimate velocity or not.

Without velocity estimation, the state vector only has 5 states; x, y, z, clock bias, clock drift.

With velocity estimation, the state vector has 8 states; x, y, z, clock bias, Vx, Vy, Vz, clock drift.

## Process Model (F)

The receiver position is modeled assuming the receiver is static (dataset 3 and 4 are known to be static), unless the user wishes to estimate velocity, in which case the receiver is modeled as some position plus some change in position. This can be seen in the initialization of the F matrix in the code featured in Appendix A. The clock bias is modeled to be dynamic (bias + some drift), in the F matrix with and without velocity estimation.

## Process Noise (Q)

The process noise is assumed to be 0 for our position estimation, as well as velocity estimation. This was determined through trial and error. Having any noise greater than 0 rapidly increases RMS errors, this suggests that the system is pretty well modeled. A small process noise of 100 meters was given to clock bias and drift, also found through trial and error.

## Measurement Noise (R)

The R matrix uses a weighting function of 2 times the inverse of the sine of the elevation angle. The weighting function is then multiplied against the identity matrix to form the R matrix. The coefficient 2 was found through trial and error and comparing the RMS error.

## Comparison of LLS and KF

The tables and plots below compare the difference in ENU errors between the KF and a LLS estimator for data sets 3 and 4. Both the LLS and the KF use ionosphere free pseudorange data, as well as include the troposphere delay. From looking at the plots, the KF starts with a large initial condition error that eventually settle, whereas the LLS estimator doesn’t seem to have the initial condition error. However comparing RMS errors, it is clear to see that the KF is more accurate and precise than the LLS estimator.

Table : Comparison of RMSE for Data Set 3

|  |  |  |
| --- | --- | --- |
|  | **LLS** | **KF** |
| **RMSE East** | 0.9904 | 0.4906 |
| **RMSE North** | 1.5274 | 0.1908 |
| **RMSE Up** | 2.5815 | 0.6997 |

Table 2: Comparison of RMSE for Data Set 4

|  |  |  |
| --- | --- | --- |
|  | **LLS** | **KF** |
| **RMSE East** | 1.0512 | 0.2024 |
| **RMSE North** | 1.2525 | 0.3252 |
| **RMSE Up** | 2.1913 | 0.7048 |

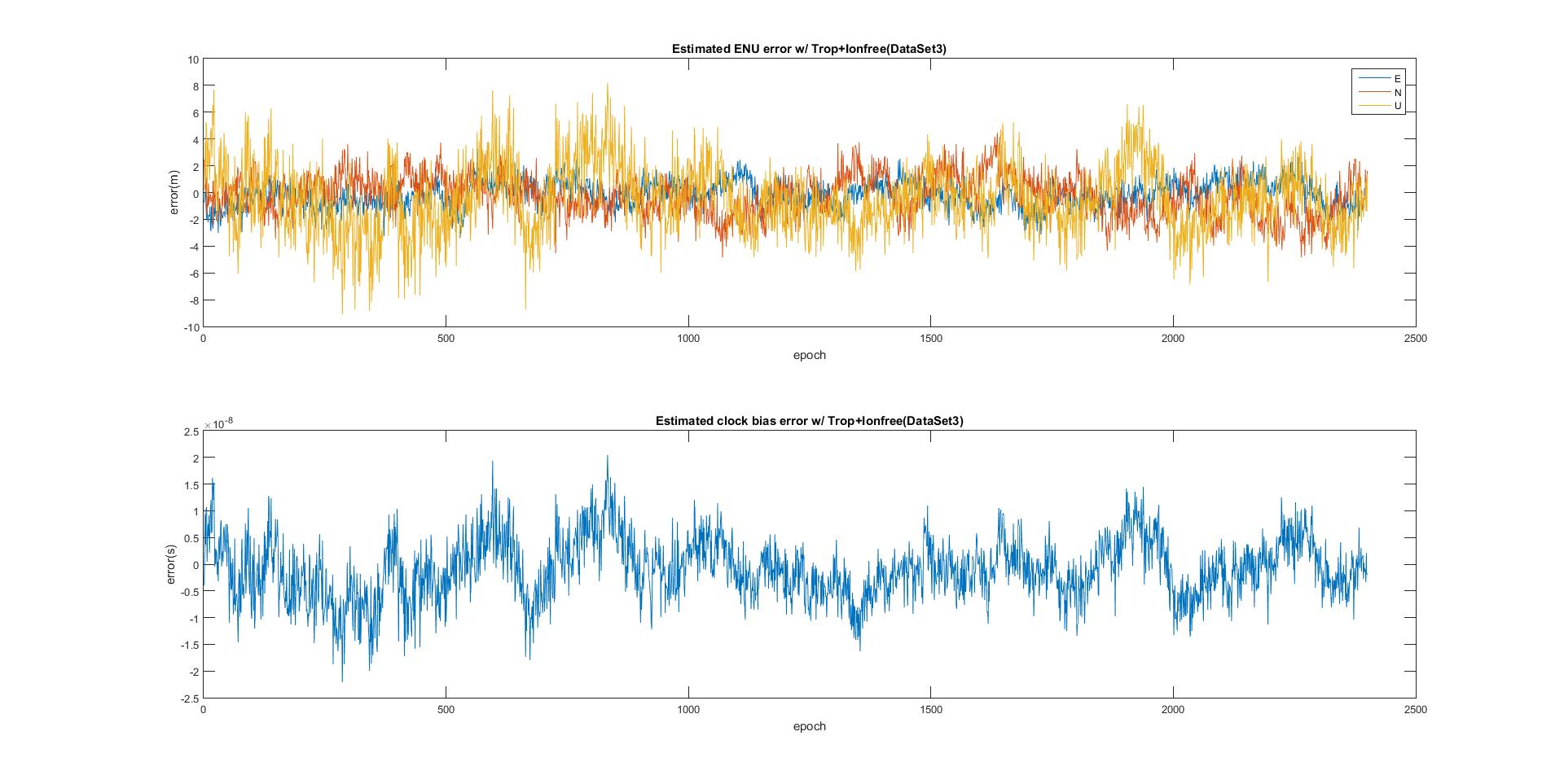


Figure : LLS estimator ENU error for Data Set 3

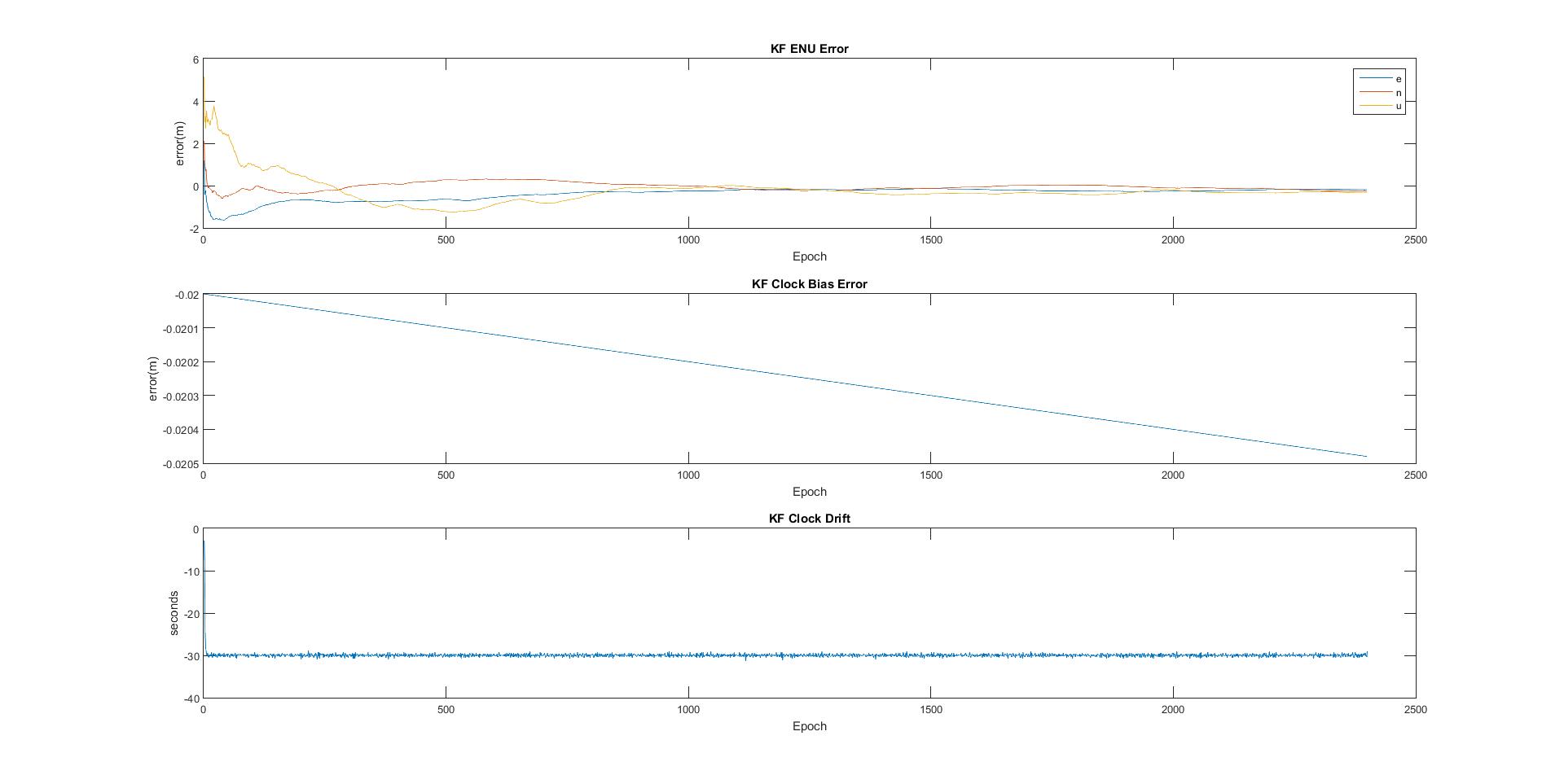


Figure :KF ENU error for Data Set 3

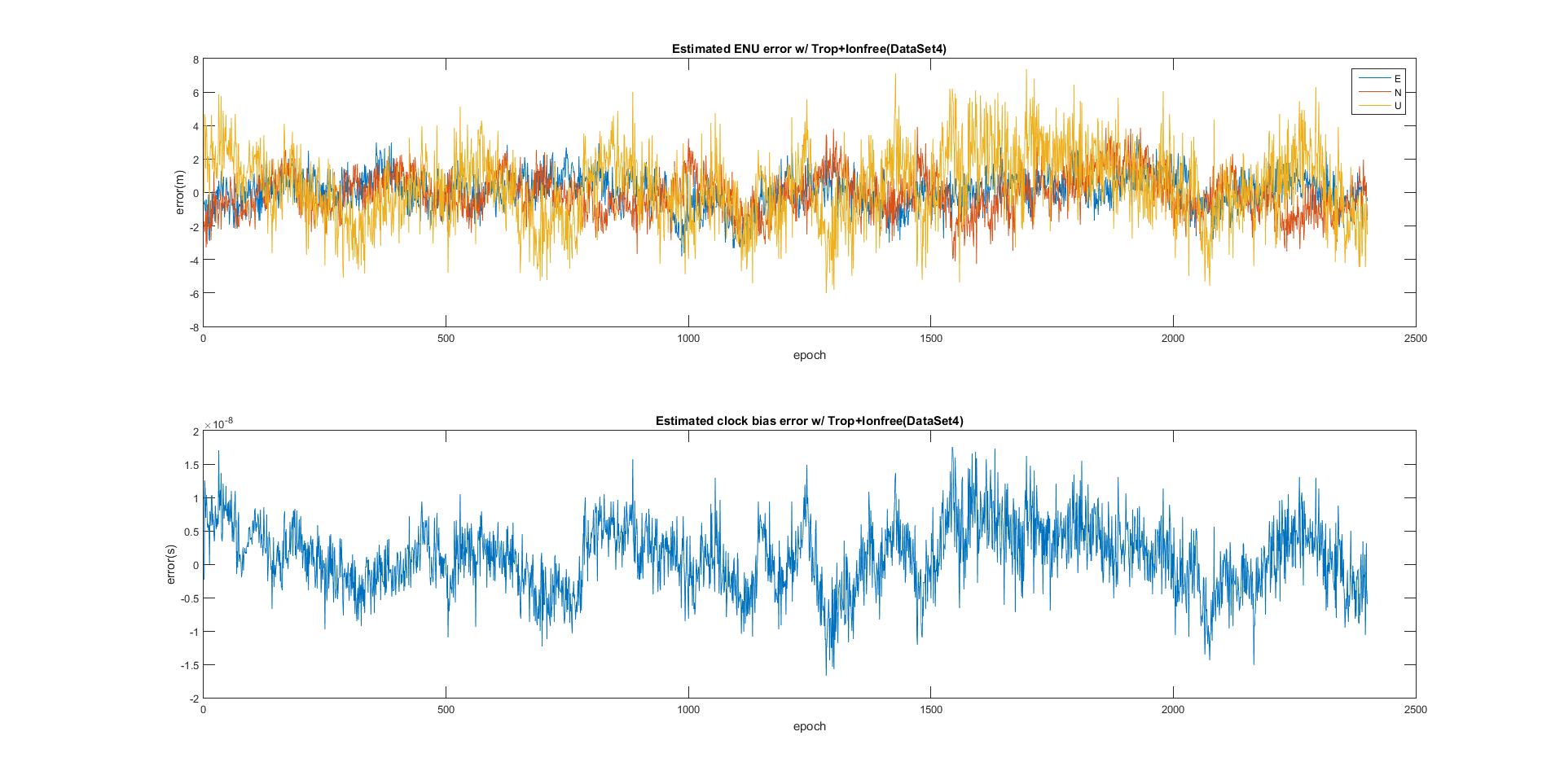


Figure :LLS Estimator ENU error for Data Set 4

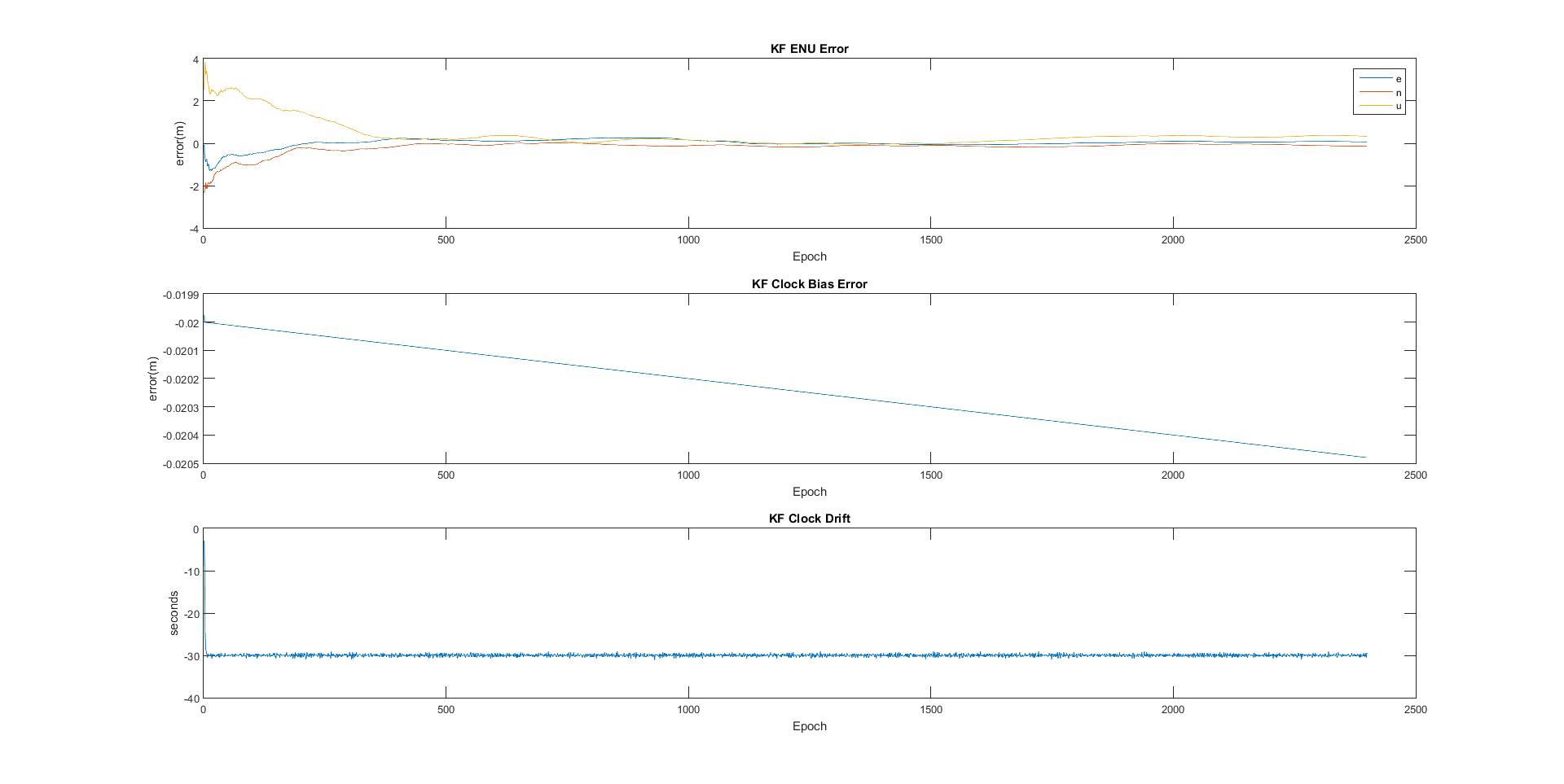


Figure :KF ENU error for Data Set 4

# Carrier Phase Smoothing

Figures 5 and 6 show the difference between non-smoothed, and smoothed pseudorange data used in position estimation for Data Set 5. It should be noted that for this data set, the process noise for position was changed to be 100m^2, as the receiver is moving. Smoothing the pseudorange makes an apparent difference in the plots, and significantly improves the RMS errors, as shown in table 3. The difference between smoothed and unsmoothed errors, comes from the mitigation of the unmodeled errors. This can be reasoned from the fact that the troposphere model was included in the filter, as well as the fact that the filter works with ionosphere free data, leaving only the unmodeled errors. These could include multi-path and/or thermal noise. Figure 7 shows the effect of not modeling troposphere on the smoothed pseudorange data, where it is clear to see, that smoothing does not impact the troposphere delay.

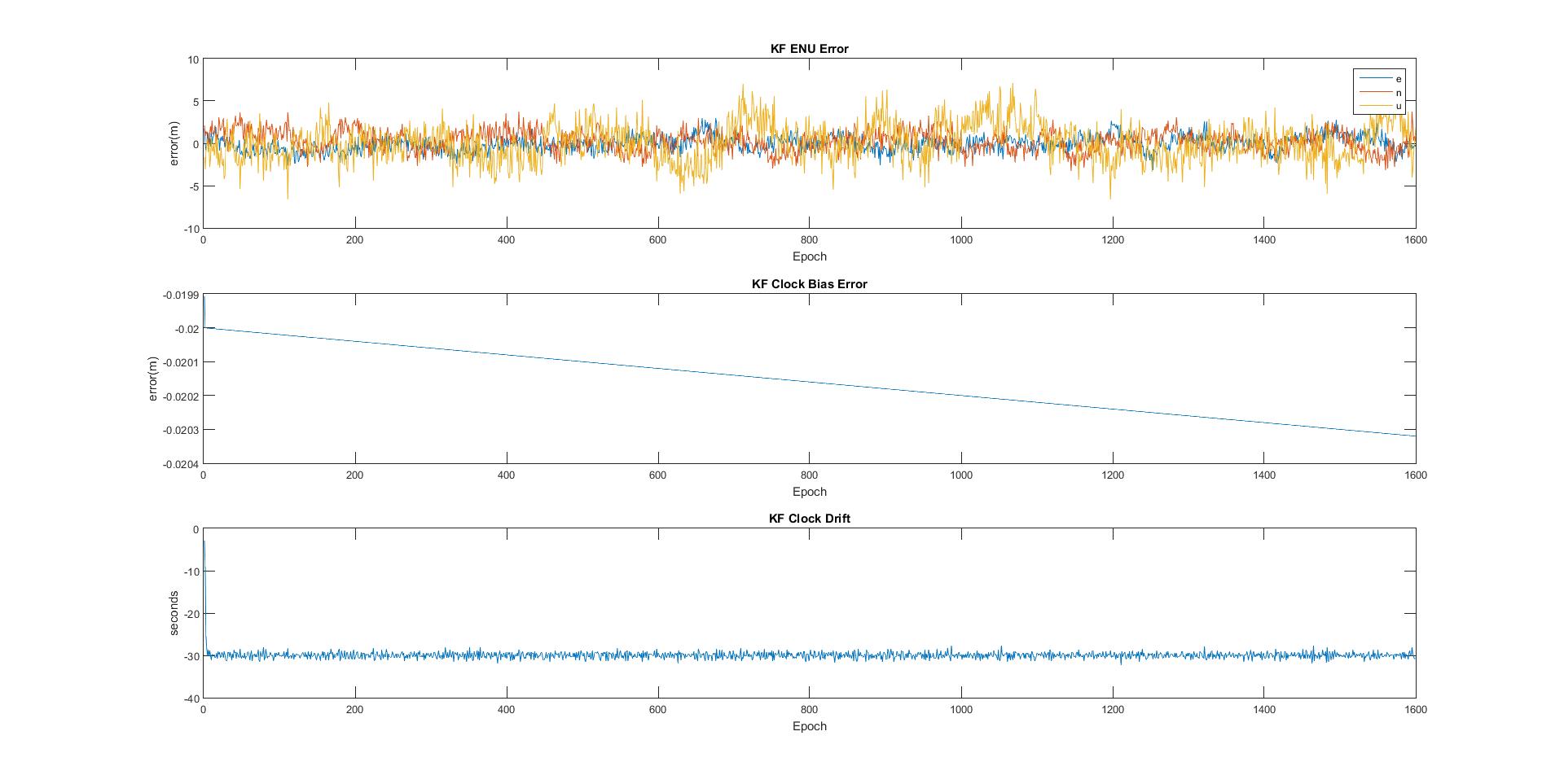


Figure :ENU error for non-smoothed Data Set 5

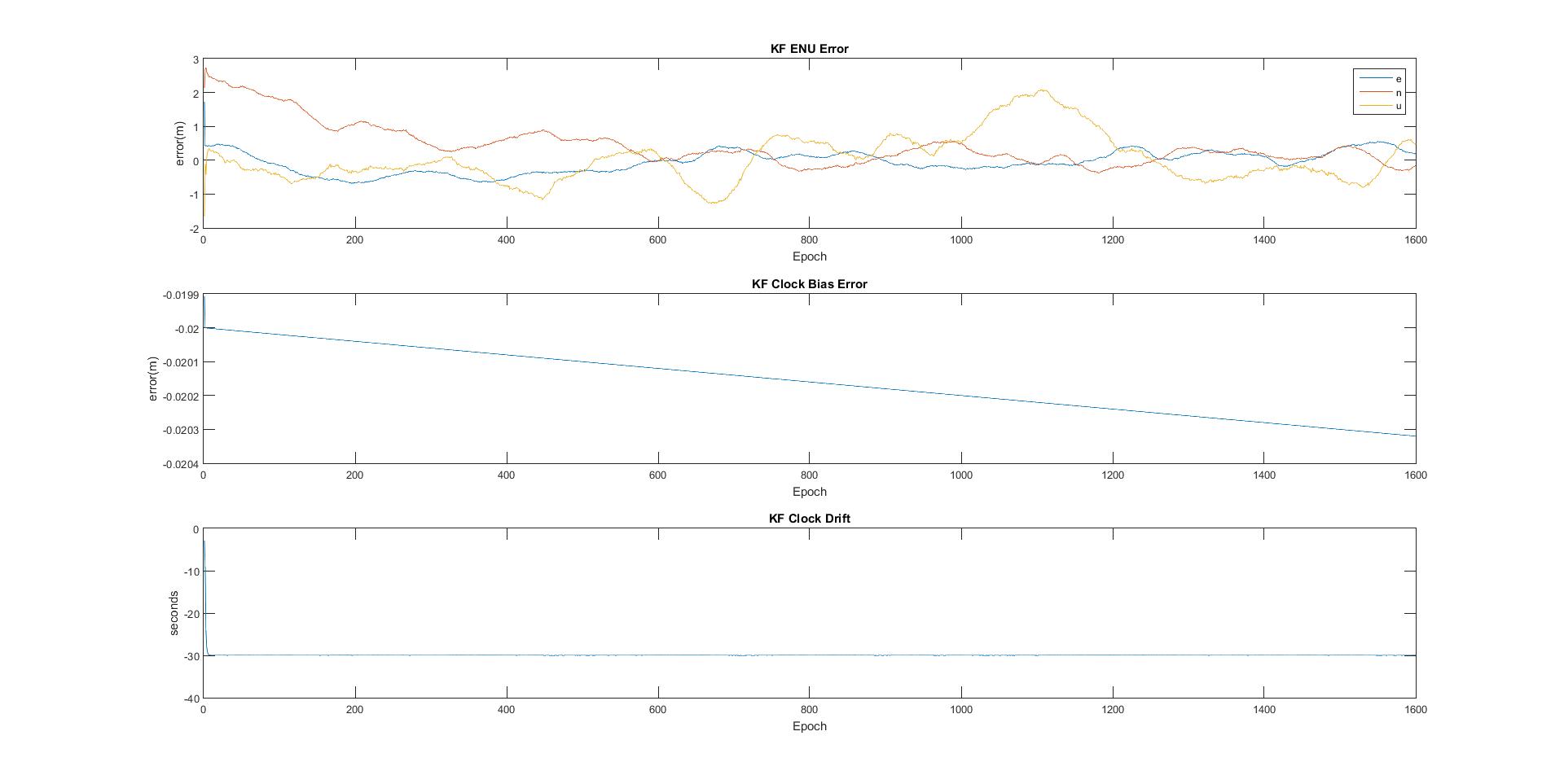


Figure :ENU error for smoothed Data Set 5

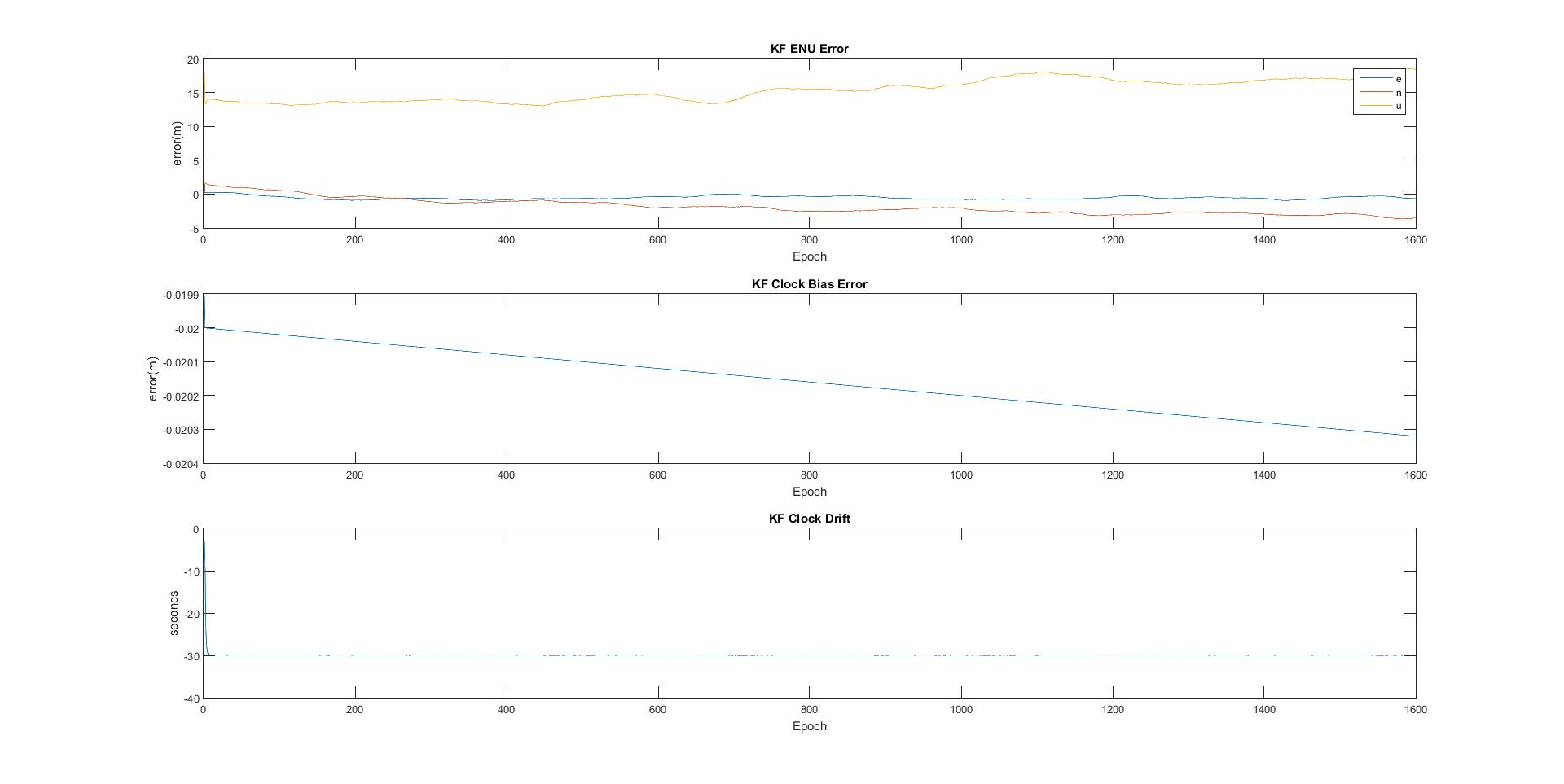


Figure :Smoothed pseudorange without troposphere modeling on Data Set 5

Table 3: Comparison of non-smoothed and smoothed RMS errors for Data Set 5

|  |  |  |
| --- | --- | --- |
|  | **Non-Smoothed** | **Smoothed** |
| **RMSE East** | 0.9044 | 0.3187 |
| **RMSE North** | 1.2112 | 0.7345 |
| **RMSE Up** | 2.2175 | 0.7051 |

# Velocity Estimation

The KF can be used to estimate receiver velocity using the pseudorange rate in Data Set 5. Using this method, the estimated velocities and the RMS ENU errors can be found in table 4. RMS errors using this method are comparable to the smoothed data from the previous section, with the exception of RMS in Up. From Figure 8 below, it also appears that Up does not converge like it does using carrier phase smoothing.

Table 4: RMS erros and Velocity estimations for Data Set 5 using pseudorange rate

|  |  |
| --- | --- |
|  | **Non-smoothed** |
| **RMSE East** | 0.5811 |
| **RMSE North** | 0.4951 |
| **RMSE Up** | 1.6430 |
| **Vx(m/s)** | 0.0516 |
| **Vy(m/s)** | 0.0166 |
| **Vz(m/s)** | 0.0115 |

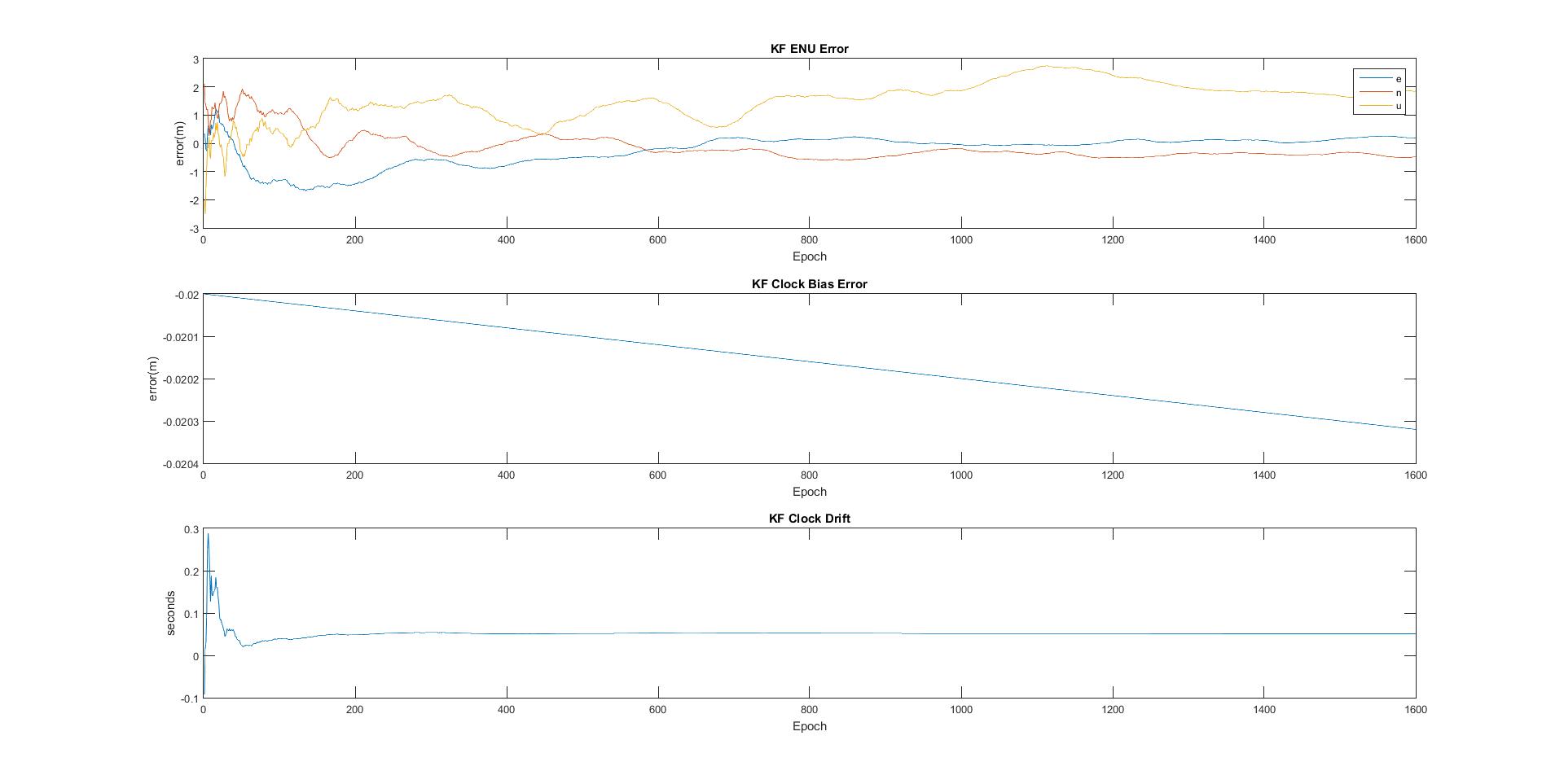


Figure : ENU errors for Data Set 5 using PVT filter

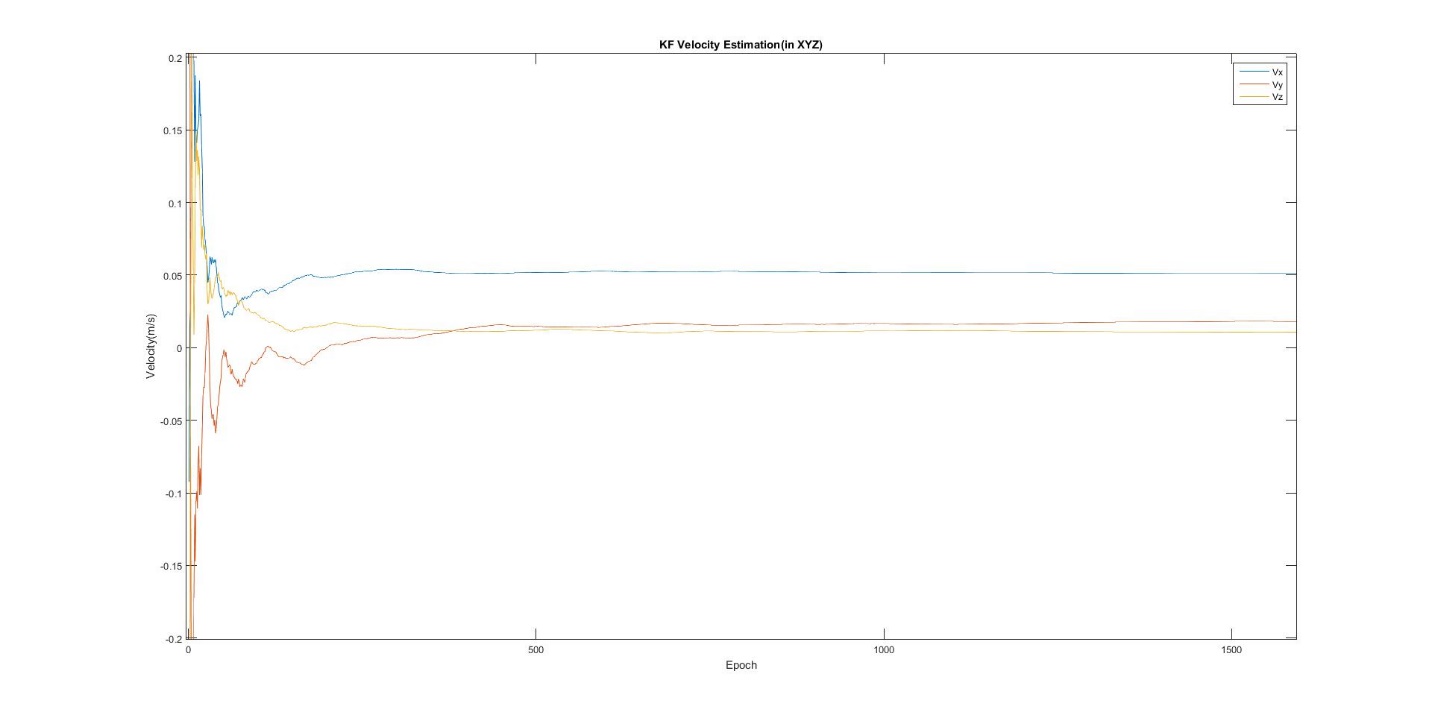


Figure :Velocity estimate in X, Y, Z

# Supplemental Questions

1. Answered in Carrier Smoothing Section
2. The assumption is that the delays do not change, for the reasons that it is over a short period of time, and that we are in the same general location. This assumption is valid for Data Set 5 for both smoothing and pseudorange rate. If the velocity estimation is right, then the receiver isn’t moving very far, satisfying the same general location assumption, and the short period of time is satisfied by the fact that the measurements are taken only 1 second apart. The notes suggest that the troposphere delays cancel out, but as shown in figure 6 and 7, we can’t ignore the delay from troposphere without increasing our error.

# Appendix A

%Kalman Filter HW3

%Sean Lantto

clear all

load('dataSet5.mat')

%Setting Constants

N=length(nSat); %number of epochs

c=299792458; %speed of light, m/s

smooth=input('Carrier Phase Smoothing?(1 for yes, 0 for no)');

velocity=input('given prRate?(1 for yes, 0 for no)');

trop=input('Include Troposphere delay(1 for yes 0 for no)');

if trop==1

Po=1013;

To=288.15;

eo=12.8;

% Po=input('what is your total pressure?(mbar)');

% To=input('what is your temperature?(kelvin)');

% eo=input('what is your partial pressure do to water?(mbar)');

else

Po=0;

To=0;

eo=0;

end

if velocity==1

%state-vector x=[delta-x, delta-y, delta-z, clock bias, Vx, Vy, Vz,clock drift]

%assume delta is zero intially

x=[0 0 0 0 0 0 0 0]';

% initiallize uncertainty, 100m at xyz

P=[100^2 0 0 0 0 0 0 0 ;

0 100^2 0 0 0 0 0 0 ;

0 0 100^2 0 0 0 0 0 ;

0 0 0 10000^2 0 0 0 0 ;

0 0 0 0 100^2 0 0 0;

0 0 0 0 0 100^2 0 0;

0 0 0 0 0 0 100^2 0;

0 0 0 0 0 0 0 100^2];

% State transition matrix

%user is stationary

F=[1 0 0 0 1 0 0 0;

0 1 0 0 0 1 0 0;

0 0 1 0 0 0 1 0;

0 0 0 1 0 0 0 1;

0 0 0 0 1 0 0 0;

0 0 0 0 0 1 0 0;

0 0 0 0 0 0 1 0;

0 0 0 0 0 0 0 1];

% process noise

Qpos=00

Qvel=00

Q=[ Qpos^2 0 0 0 0 0 0 0;

0 Qpos^2 0 0 0 0 0 0;

0 0 Qpos^2 0 0 0 0 0;

0 0 0 100 0 0 0 0;

0 0 0 0 Qvel^2 0 0 0;

0 0 0 0 0 Qvel^2 0 0;

0 0 0 0 0 0 Qvel^2 0;

0 0 0 0 0 0 0 100];

elseif velocity==0

%state-vector x=[delta-x, delta-y, delta-z, clock bias, clock drift]

%assume delta is zero intially

x=[0 0 0 0 0]';

% initiallize uncertainty, 100m at xyz

P=[100^2 0 0 0 0;

0 100^2 0 0 0;

0 0 100^2 0 0;

0 0 0 10000^2 0

0 0 0 0 10^2];

% State transition matrix

%user is stationary

F=[1 0 0 0 0;

0 1 0 0 0;

0 0 1 0 0;

0 0 0 1 1;

0 0 0 0 1];

% process noise

Qpos=100

Q=[ Qpos^2 0 0 0 0;

0 Qpos^2 0 0 0;

0 0 Qpos^2 0 0;

0 0 0 100 0;

0 0 0 0 100];

end

%use Ionosphereic free data

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

if smooth==1

M=100;

prSmooth=carrierSmooth(phaseL1,phaseL2,M,prDataIF);

prDataIF=prSmooth;

end

if velocity==1

prRateL2(:,1600)=prRateL1(:,1600);

prRateIF=(2.546\*prRateL1)-(1.546\*prRateL2);

end

llh=xyz2llh(nomXYZ);

for i=1:N

% step #1 predict state

x=F\*x;

% step #2 predict error-covarince

P=F\*P\*F'+Q;

% form observation matrix

n=length(x);

if velocity==1

m=2\*nSat(i);

elseif velocity==0

m=nSat(i);

end

H=zeros(m,n);

prComputed=zeros(nSat(i),1);

for j=1:nSat(i)

prComputed(j)=norm(satsXYZ(j,:,i)-nomXYZ)+clockBiasNom\*c;

if velocity==1

prSquigDot(j)=prRateIF(j)-(satsVxVyVz(j,:,i)\*((satsXYZ(j,:,i)-nomXYZ)/norm(satsXYZ(j,:,i)-nomXYZ))');

end

H(j,1:4)=[(satsXYZ(j,:,i)-nomXYZ)/norm(satsXYZ(j,:,i)-nomXYZ), 1];

end

if velocity==1

H(((m/2)+1):m,5:8)=-1\*H(1:(m/2),1:4);

end

% form meaurement error covariance ( could do elevation dependent weighting here)

R=(2^2)\*eye(m);

for j=1:nSat(i)

Satenu(j,:)=xyz2enu(satsXYZ(j,:,i),nomXYZ);

sinel(j,:)=(Satenu(j,3))/(norm(Satenu(j,:)));

R(j,j)=R(j,j)\*(1/sinel(j));

el(j,i)=asind(sinel(j,:));

mel(j)=1.001/(sqrt(0.002001+((sind(el(j,i)))^2)));%found on navipedia

if trop==1

prComputed(j)=prComputed(j)+mel(j)\*TropSaastamoinen(llh,Po,To,eo);

end

end

% step #3 compute Kalman Gain

K=P\*H'\*inv(H\*P\*H'+R);

% step #4 update state

% measurement vector

if velocity==0

z=prComputed-prDataIF(1:nSat(i),i); %delta-rho

y=H\*x;

x=x+K\*(z-y);

elseif velocity==1

z(1:nSat(i))=prComputed-prDataIF(1:nSat(i),i); %delta-rho

z((nSat(i)+1):(2\*nSat(i)))=prSquigDot;

y=H\*x;

x=x+K\*(z'-y);

end

% step # 5 update measurment error covariance

P=(eye(length(x))-K\*H)\*P;

% step 6 save estimate and move to next step

xyzKF(i,1:3)=nomXYZ'+x(1:3);

clockBiasKF(i)=clockBiasNom'+(x(4));

KFclkDrift(i)=x(5);

if velocity==1

VxyzKF(i,1:3)=x(5:7);

end

enuTruth(i,:)=xyz2enu(truthXYZ(:,i),nomXYZ);

enuKF(i,:)=xyz2enu(xyzKF(i,1:3),nomXYZ);

KF\_3DErr(i)=norm(enuKF(i,:)-enuTruth(i,:));

KF\_clkBiasErr(i)=(clockBiasKF(i)-truthClockBias(i))/c;

llh=xyz2llh(xyzKF(i,:));

end

% figure

% plot(KF\_3DErr)

% figure

% plot(KF\_clkBiasErr/1000)

figure

subplot(311)

plot(enuKF-enuTruth)

title('KF ENU Error')

xlabel('Epoch')

ylabel('error(m)')

legend('e','n','u')

subplot(312)

plot(KF\_clkBiasErr)

title('KF Clock Bias Error')

xlabel('Epoch')

ylabel('error(m)')

subplot(313)

plot(KFclkDrift)

title('KF Clock Drift')

xlabel('Epoch')

ylabel('seconds')

RMSEenu(1)=sqrt(mean((enuKF(:,1)-enuTruth(:,1)).^2));

RMSEenu(2)=sqrt(mean((enuKF(:,2)-enuTruth(:,2)).^2));

RMSEenu(3)=sqrt(mean((enuKF(:,3)-enuTruth(:,3)).^2));

figure

plot(KF\_3DErr)

title('KF 3D Error')

xlabel('Epoch')

ylabel('error(m)')

if velocity==1

figure

plot(VxyzKF)

title('KF Velocity Estimation(in XYZ)')

xlabel('Epoch')

ylabel('Velocity(m/s)')

legend('Vx','Vy','Vz')

end

# Appendix B

function prSmooth=carrierSmooth(phaseL1,phaseL2,M,prDataIF)

phaseIF=(2.546\*phaseL1)-(1.546\*phaseL2);

prSmooth(:,1)=prDataIF(:,1);

for j=2:length(prDataIF)

prSmooth(:,j)=((1/M)\*prDataIF(:,j))+(((M-1)/M)\*(phaseIF(:,j)-phaseIF(:,j-1)+prSmooth(:,j-1)));

end

end