בס"ד

HW1-KF

Ori Aharon (ID: 200274504)

Nadav Lehrer (ID: 302170378)

1) for linear model system:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \qquad with\ \ \varepsilon_t \sim N(0,R_t)$$

with observations:

$$z_k = C_t x_t + \delta_t \qquad with\ \ \delta_t \sim N(0,Q_t)$$

Kalman Filter algorithm:

$$\begin{aligned}
&prediction:\\
&\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t\\
&\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t\\
&correction\,/\,filtering:\\
&S_t = C_t \bar{\Sigma}_t C_t^T + Q_t\\
&K_t = \bar{\Sigma}_t C_t^T S_t^{-1}\\
&\bar{z}_t = C_t \bar{\mu}_t\\
&\mu_t = \bar{\mu}_t + K_t (z_t - \bar{z}_t)\\
&\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t = (I - K_t C_t)\bar{\Sigma}_t (I - K_t C_t)^T + K_t Q_t K_t^T
\end{aligned}$$

for the following system:

$$x_k = x_{k-1} + \varepsilon_k \qquad with\ \ \varepsilon_k \sim N(0,1)$$

with observations:

$$z_k = x_k + \delta_k \qquad with\ \ \delta_k \sim N(0,2)$$

$$A_t = 1,\ B_t = 0,\ C_t = 1,\ R_t = 1,\ Q_t = 2$$

KF will produce the following recursion algorithm:

$$\begin{aligned}
&\textit{prediction}:\\
&\bar{\mu}_t = \mu_{t-1}\\
&\bar{\Sigma}_t = \Sigma_{t-1} + 1\\
&\textit{correction / filtering}:\\
&K_t = \frac{\bar{\Sigma}_t}{\bar{\Sigma}_t + 2}\\
&\bar{z}_t = \bar{\mu}_t\\
&\mu_t = \bar{\mu}_t + K_t(z_t - \bar{z}_t)\\
&\Sigma_t = (1 - K_t)\bar{\Sigma}_t
\end{aligned}$$

1a)

for the first step:

$$\begin{aligned}
&\textit{prediction}:\\
&\bar{\mu}_1 = \mu_0 = 1\\
&\bar{\Sigma}_1 = \Sigma_0 + 1 = 11\\
&\textit{correction / filtering}:\\
&K_1 = \frac{\bar{\Sigma}_1}{\bar{\Sigma}_1 + 2} = \frac{11}{13}\\
&\bar{z}_1 = \bar{\mu}_1 = 1\\
&\mu_1 = \bar{\mu}_1 + K_1(z_1 - \bar{z}_1) = 1 + \frac{11}{13}(2-1) = 1\frac{11}{13}\\
&\Sigma_1 = (1 - K_1)\bar{\Sigma}_1 = \left(1 - \frac{11}{13}\right)11 = 1\frac{9}{13}
\end{aligned}$$

for the second step:

$$\begin{aligned}
&\textit{prediction}:\\
&\bar{\mu}_2 = \mu_1 = 1\frac{11}{13}\\
&\bar{\Sigma}_2 = \Sigma_1 + 1 = 2\frac{9}{13}\\
&\textit{correction / filtering}:\\
&K_2 = \frac{\bar{\Sigma}_1}{\bar{\Sigma}_1 + 2} = \frac{35}{61}\\
&\bar{z}_2 = \bar{\mu}_2 = 1\frac{11}{13}\\
&\mu_2 = \bar{\mu}_2 + K_2(z_2 - \bar{z}_2) = 1\frac{11}{13} + \frac{35}{61}\left(3 - 1\frac{11}{13}\right) = 2\frac{31}{61}\\
&\Sigma_2 = (1 - K_2)\bar{\Sigma}_2 = \left(1 - \frac{35}{61}\right)2\frac{9}{13} = 1\frac{9}{61}
\end{aligned}$$

Table 1.1: The requested values for two time steps increment as seen in the calculations above.

k	$\bar{\mu}_k$	$\bar{\Sigma}_k$	K_k	μ_k	Σ_k
1	1	11	$\frac{11}{13}$	$1\frac{11}{13}$	$1\frac{9}{13}$
2	$1\frac{11}{13}$	$2\frac{9}{13}$	$\frac{35}{61}$	$2\frac{31}{61}$	$1\frac{9}{61}$

1b)

$\bar{\mu}_k$ - Estimated state without the integration of observation i.e. prediction step based on the linear model only.

$\bar{\Sigma}_k$ - Estimated (Co)variance without the integration of observation. This is simply the Covariance matrix of a random vector which undergoes a linear transformation.

K_k - Kalman Gain (weights the observation error).

μ_k - Estimated state using the observation, this is the maximum posterior estimation.

Σ_k - estimated (Co)variance using the observation.

1c)

$$\Sigma_t = (1-K_t)\bar{\Sigma}_t$$

$$\Sigma_t = \left(1-\frac{\bar{\Sigma}_t}{\bar{\Sigma}_t+2}\right)\bar{\Sigma}_t$$

$$\Sigma_t = \left(\frac{2}{\bar{\Sigma}_t+2}\right)\bar{\Sigma}_t$$

$$\Sigma_t = \frac{2\bar{\Sigma}_t}{\bar{\Sigma}_t+2}$$

$$\Sigma_t = \frac{2\left(\Sigma_{t-1}+1\right)}{\Sigma_{t-1}+1+2} = \frac{2\Sigma_{t-1}+2}{\Sigma_{t-1}+3}$$

$$\textit{for } k \to \infty:$$

$$\Sigma_\infty = \frac{2\Sigma_\infty + 2}{\Sigma_\infty + 3}$$

$$\Sigma_\infty{}^2 + 3\Sigma_\infty = 2\Sigma_\infty + 2$$

$$\Sigma_\infty{}^2 + \Sigma_\infty - 2 = 0$$

$$\Sigma_\infty = 1, \cancel{-2} \quad \left(\Sigma_\infty > 0\right)$$

$$\Sigma_\infty = 1$$

2a)

state sransition :

$$x_t = x_{t-1} + \dot{x}_{t-1}\Delta t + \frac{1}{2}a_t\Delta t^2$$
$$\dot{x}_t = \quad \dot{x}_{t-1} \quad + a_t\Delta t \qquad \text{with } a_t \sim N(a,\sigma_a{}^2)$$

$$a_t = a + \eta\cdot\sigma_a \quad \text{with } \eta \sim N(0,1)$$

$$x_t = x_{t-1} + \dot{x}_{t-1}\Delta t + \frac{1}{2}\left(a+\eta\cdot\sigma_a\right)\Delta t^2$$
$$\dot{x}_t = \quad \dot{x}_{t-1} \quad + \left(a+\eta\cdot\sigma_a\right)\Delta t$$

$$x_t = x_{t-1} + \dot{x}_{t-1}\Delta t + \frac{1}{2}a\Delta t^2 + \frac{1}{2}\sigma_a\Delta t^2\cdot\eta$$
$$\dot{x}_t = \quad \dot{x}_{t-1} \quad + a\Delta t \quad + \sigma_a\Delta t\cdot\eta$$

in state space form :

$$\begin{pmatrix} x_t \\ \dot{x}_t \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{pmatrix} + \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix} a + \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix}\sigma_a\cdot\eta$$

linear system:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$
$$z_t = C_t x_t + \delta_t$$

while :

the state : $x_t = \begin{pmatrix} x_t \\ \dot{x}_t \end{pmatrix}$

$$A_t = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}, \quad B_t = \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix}, \quad u_t = a\ , \quad \varepsilon_t = \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix}\sigma_a\cdot\eta$$

$$C_t = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad \text{with } \delta_t \sim N\left(0,\sigma^2\right)$$

$$R_t = E\left[\varepsilon_t \cdot \varepsilon_t^T\right] = \begin{pmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{pmatrix} \sigma_a^{\ 2} \cdot E\left[\eta^2\right] = \begin{pmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{pmatrix} \sigma_a^{\ 2}$$

$$Q_t = E\left[\delta_t \cdot \delta_t^T\right] = \sigma^2$$

2.b,c)

position and velocity without Kalman filter:

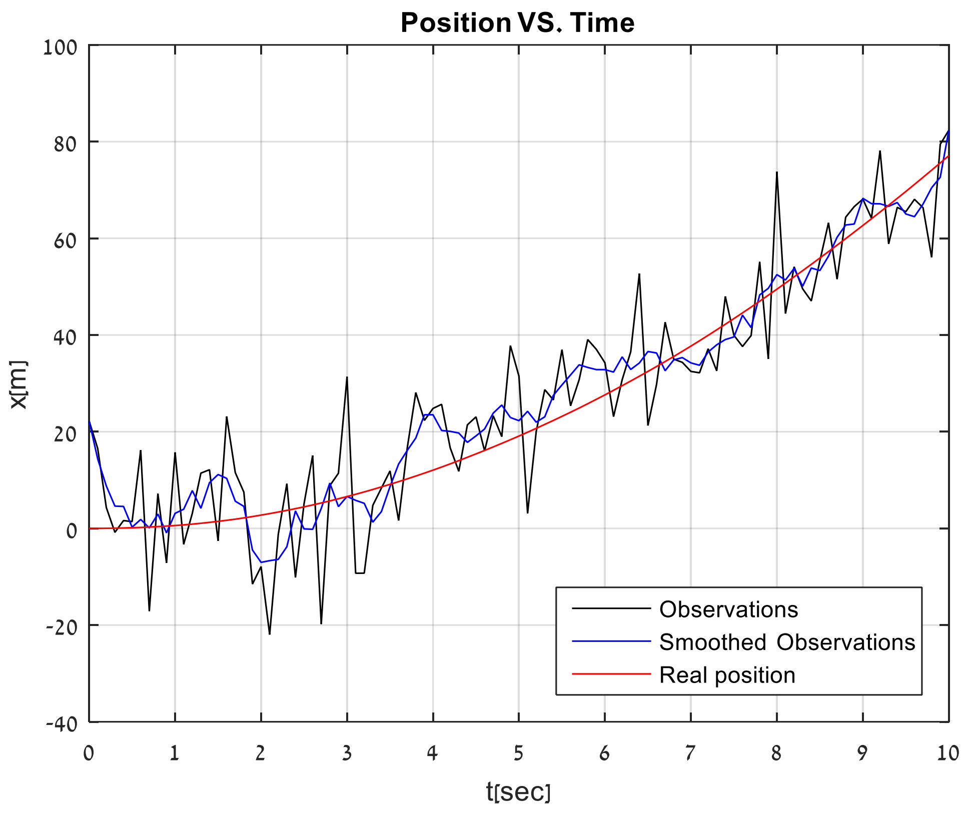

Figure 2.1: Simulation of the linear model presented in the previous section with observed noisy data, real state position and a moving average filter to estimate the state’s position (denoted as “smoothed observations”)

In this section, a simulation of the process was held and an estimation of the state’s position was yielded using MATLAB’s “smooth” function.

This MATLAB function practically preforms a convolution of two series, one of which is the observed data and the other is an average “mask” filter (in image processing this type of filtering technique is referred to as “mask” filters). It can be shown that this filter is equivalent to a Low Pass filter in the frequency domain.

We should note that this estimation will be very poor compared to the Kalman Filter estimation for the reason that this technique doesn’t take to consideration the linear model and the uncertainties at play, and thus attributes significance only to the observations as this is the only data that is used.

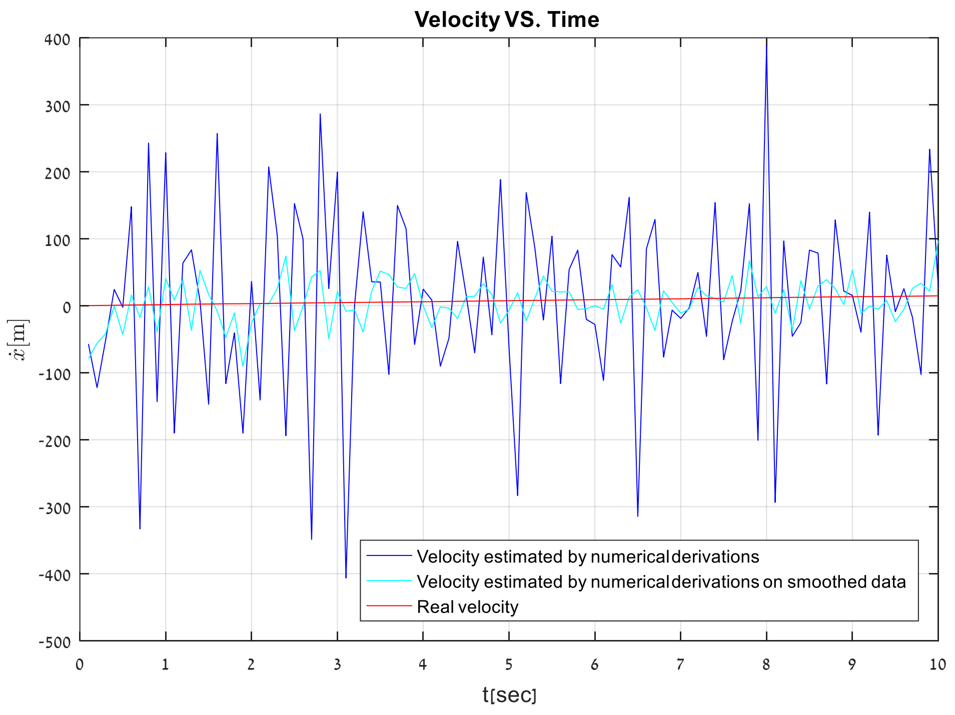

Figure 2.2: The state vectors velocity entry, as achieved from first order numerical derivative. Because we differentiate noisy position data, it is expected that the result will be noisy as well.

After obtaining the position of the object on which we modeled the system, we can perform numerical differentiation on the sensor data as well as on the smoothed data via the forward discrete derivative:

$$\dot{x}_t = \frac{x_{t+1} - x_t}{\Delta t}$$

Differentiating noisy data yields poor results, though the differentiating the smoothed results, has shown better results the those of the pure data differentiation.

2d)

position and velocity with Kalman filter, $\left(\sigma^2 = 100\right)$:

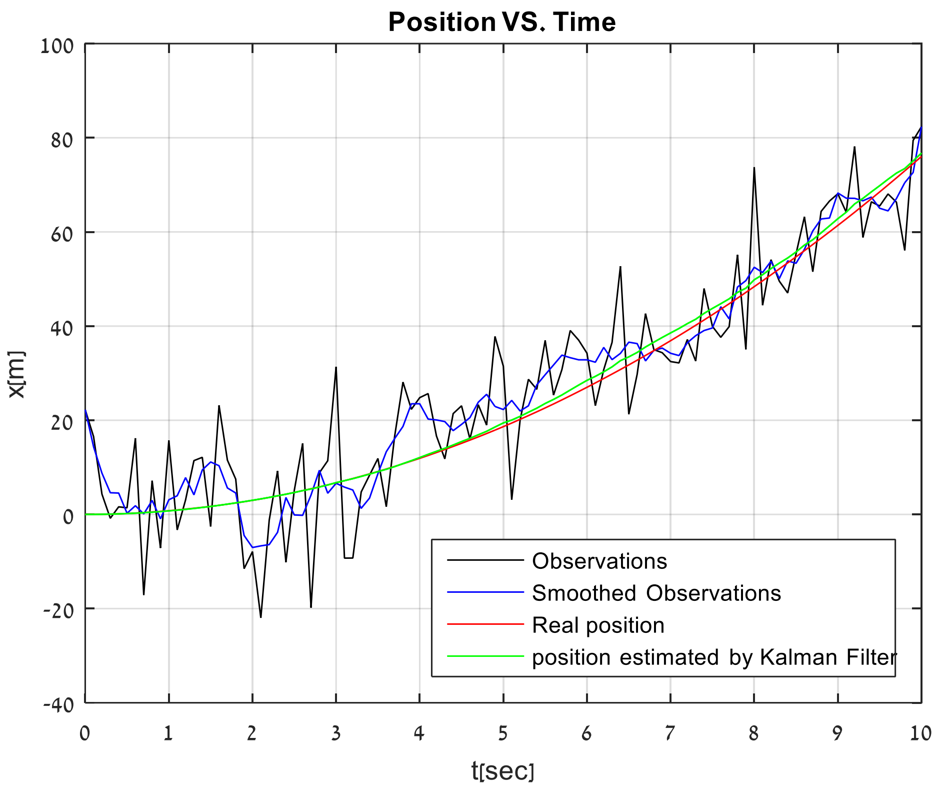

Figure 2.3: The position as retrieved from the sensor reading in black, smoothed sensor readings as seen in the previous section in blue, Kalman Filter position estimation in green and finally the real position in red. Clearly, the Kalman Filter achieves far better results then the moving average filter.

In this section the Kalman Filter algorithm was implemented on the process, and a variance of $\sigma_{obs}^2 = 100$, was attributed to the sensor reading. In Fig. (2.3) we clearly see that the observations are characterized by a large amount of noise (i.e. large variance), and therefor by assigning the value of $\sigma_{obs}^2 = 100$ we expect to a get credible estimation just as the results show us.

This result is due to the low process noise and high sensor noise attributed to the system which effect the estimation through the Kalman gain. As the relative noise of the sensor to the noise of system rises, the Kalman gain decreases and assigns less “weight” in the correction step, to the difference between the actual observation and observation prediction. For this reason, the process’s estimation relay’s mostly on the dynamic model’s prediction.

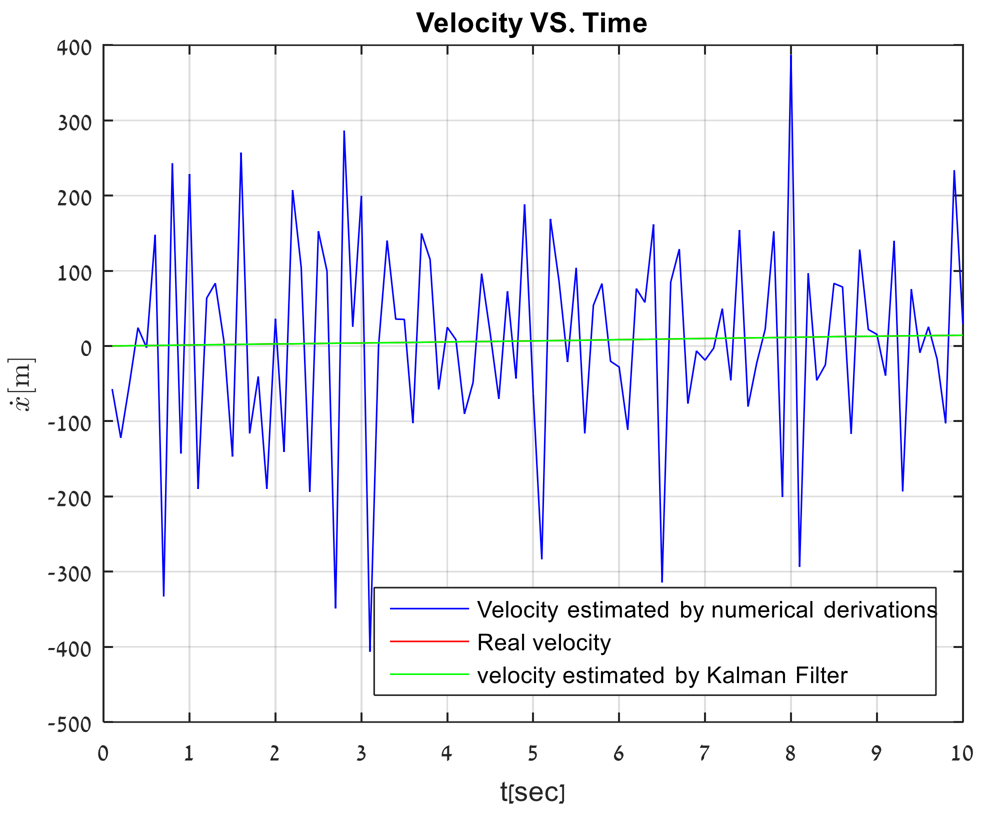

Figure 2.4: The estimated velocity using numerical differentiation of the position measurements, real velocity and the velocity estimated by the Kalman filter algorithm. Once more it is quite clear that the Kalman filter yielded far more accurate results.

The Kalman filter is also recognized as an optimal conditional expectation estimator, that is:

$$\hat{x}_t = E[x_t|Z_{1:t}]$$

In this report we will occasionally denote an estimated value with a "hat" above the symbol of the property to be estimated (As in some literature). Here $\hat{x}_t$ will be estimated state vector at time t.

The Kalman filter is optimal in the sense of the minimal mean square error i.e. the Kalman filter estimation, minimizes the expectation of the squared estimation error

$$E[(x_t - \hat{x}_t)^2]$$

Under the conditions of a linear model and Gaussian white noise.

Thus, the Euclidean distance between the state of the system to it's estimation is minimal. Being so, we would expect to get a good estimation of all the state vector entries – position and velocity. In our case, as seen in Fig. (2.4) the velocity estimation is almost spot on, due to this property of the Kalman filter.

position and velocity with Kalman filter, $\left(\sigma^2 = 1\right)$:

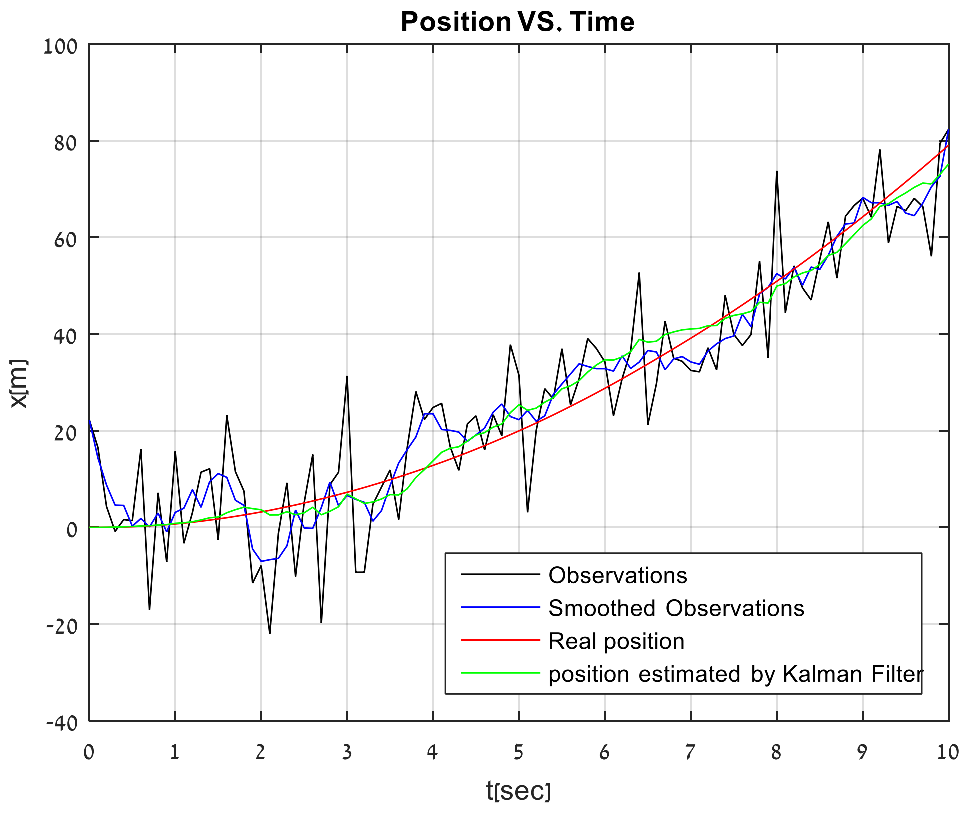

Figure 2.5: The position as retrieved from the sensor reading in black, smoothed sensor readings as seen in the previous section in blue, Kalman Filter position estimation in green and finally the real position in red. Clearly, the Kalman Filter achieves far better results then the moving average filter.

In this section we performed the same simulation as in the previous sections, only this time attributing the small variance of $\sigma_{obs}^2 = 1$, to the observation. From the simulation results we can assume that this variance of the sensor noise is rather small compared to the real variance, due to the inaccurate results that arise from Fig. (2.5). Let us try to asses the true variance of the sensor using some pooled statistics:

Given a random variable (r.v.) that represents the system's measurements

$$z = x_{realPosition} + \delta, \qquad \delta \sim N(0, \sigma_{obs}^2)$$

Therefor we can see that variance of the measurement is:

$$\mathrm{E}[(z - x_{realPosition})^2] = \mathrm{E}[\delta^2] = \sigma_{obs}^2$$

Which can be estimated by:

$$\hat{\sigma}_{obs}^2 = \frac{1}{T}\sum_{t=0}^{T}(z_t - x_t)^2$$

By performing 1000 Monte-Carlo runs the true observation variance was estimated to be

$$\hat{\sigma}_{obs}^2 \approx 110$$

which clearly justifies the claim that the preferable results were those of the simulation in the previous section ($\sigma_{obs}^2 = 100$), due to better assessment of σ_{obs}^2.

Being smaller, σ_{obs}^2 forces the algorithm to trust the measurement more (because there is less uncertainty to them) and thus results in a larger Kalman gain which gives more weight to the measurement in the final correction step.

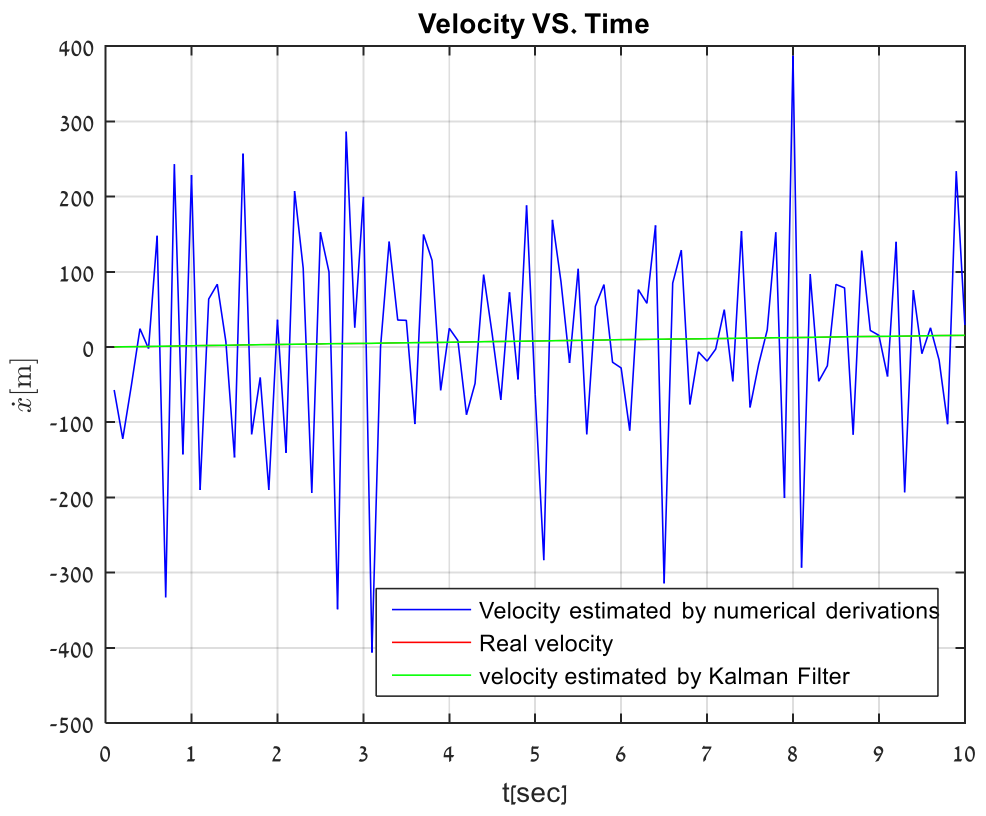

Figure 2.6.1: The estimated velocity using numerical differentiation of the position measurements, real velocity and the velocity estimated by the Kalman filter algorithm. Once more it is quite clear that the Kalman filter yielded far more accurate results than those of the numerical scheme.

Fig. (2.6.1) shows us the velocity estimate generated by the Kalman filter against the true velocity, and the velocity estimate generated by numerical differentiation of the sensor readings. It is quite clear that once again, even when we underestimated the sensors noise variance, better results were yielded via the Kalman filter that those of numerical differentiation. The reason for this is that in the Kalman filter, the process model is also taken into consideration in the prediction step, and although the algorithm gives more weight to the observations (compared to the previous section in which

$\sigma^2_{obs} = 100$), the prediction step based on the linear model, is still taken to consideration.

Once we noticed that both Kalman filters implemented, yielded far better results than those of the smoothed estimate, we are able compare the two Kalman filters. In the following figure we will notice that the velocity generated by the Kalman filter which underestimates the sensor variance, will be worse than that of the Kalman filter with $\sigma^2_{obs} = 100$.

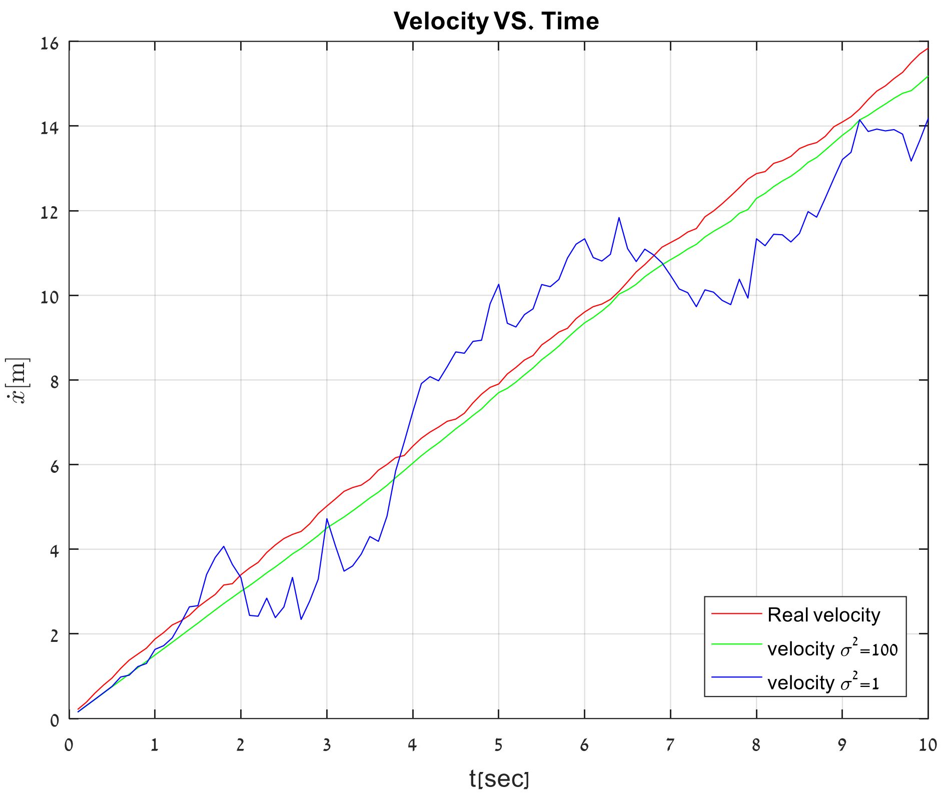

Figure 2.6.2: The velocity generated by the two Kalman filter simulations, $\sigma^2_{obs} = 100$ in green and $\sigma^2_{obs} = 1$ in blue, and the true velocity. For the reasons discussed above, we see that underestimating the sensor noise, results in decreasing accuracy.

As can be seen in Fig. (2.6.2), the velocity estimation of the underestimated $\sigma^2_{obs} = 1$, yields far less accurate results that those of the other Kalman simulation of $\sigma^2_{obs} = 100$. These results are not surprising as the position estimates has shown the same results (for the reasons discussed before).

2e)

PDF's plots for three different time steps:

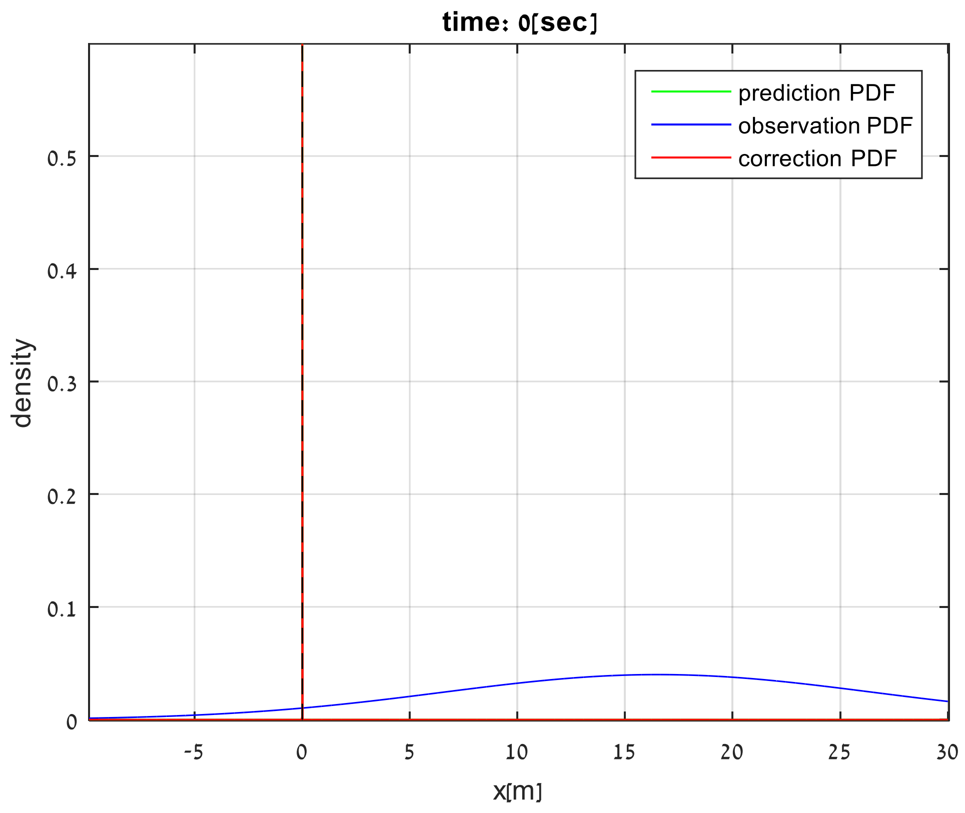

Figure 2.7: The pdf's (partial density functions) of the prediction step estimate, observation, and correction step estimate at time t=0. The straight dashed line indicates the true position of the system (in this figure the prediction and correction densities are superimposed on the true position and thus it is seen as a straight line).

In this section we plotted three functions that represent the prediction, correction and observation partial density at three different time steps $t = 0, 5, 10\ [sec]$. These pdf's are the result of the simulation that attributed a variance of $\sigma_{obs}^2 = 100$ to the measuring device.

As the Covariance in the beginning is $\Sigma_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ we can conclude that the first position of the car is deterministic. For this reason, the first position estimate is the exact position with zero uncertainty. The observation though, is highly uncertain as its variance throughout the simulation is very high.

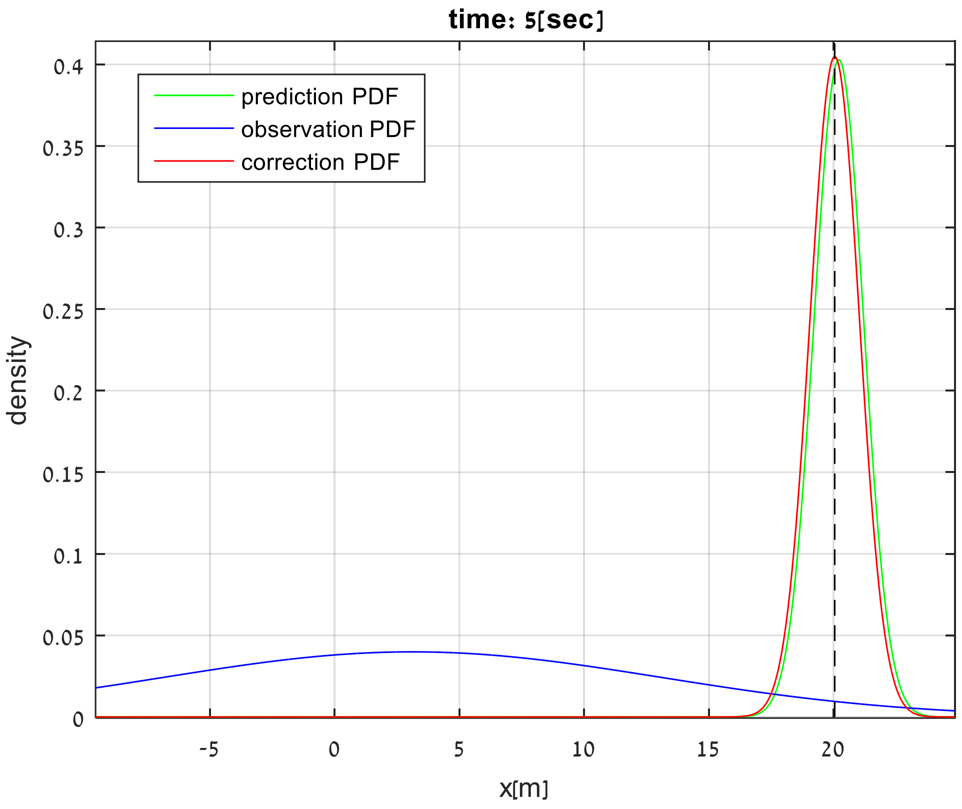

Figure 2.8: The pdf's of the prediction step estimate, observation, and correction step estimate at time t=5. The straight dashed line indicates the true position of the system.

As can be seen in Fig. (2.8) the variance of the prediction step is far smaller then this of the observation. Therefor we would expect that the correction step's density would be closer (i.e. the means of the Gaussians) to the prediction step (as a small Kalman gain would indicate). This result is reasonable as we would obviously trust the more certain data. Furthermore, it is notable that the correction step's mean would always be bounded between the means of the measurement and the prediction step's densities.

As the Kalman gain gets smaller due to large observation uncertainty, the mean of the correction step would tend more to the mean of the prediction step as:

$$\mu_t = \bar{\mu}_t + K_t(z_t - C\bar{\mu}_t)$$

Would suggest. In addition, we would expect to gain certainty as we integrate the measurements in the process, as we utilize more data, just as seen in class (predict – lose certainty, measure – gain certainty). This property is not so apparent in this simulation because of the ratio between the process uncertainty and the measurement's uncertainty. Once more, this is manifested in the Kalman gain which gets smaller and effect the process's Covariance matrix, by:

$$\Sigma_t = (I - K_t C)\bar{\Sigma}_t$$

As K_t gets smaller, the correction's covariance will be more similar to the prediction's covariance.

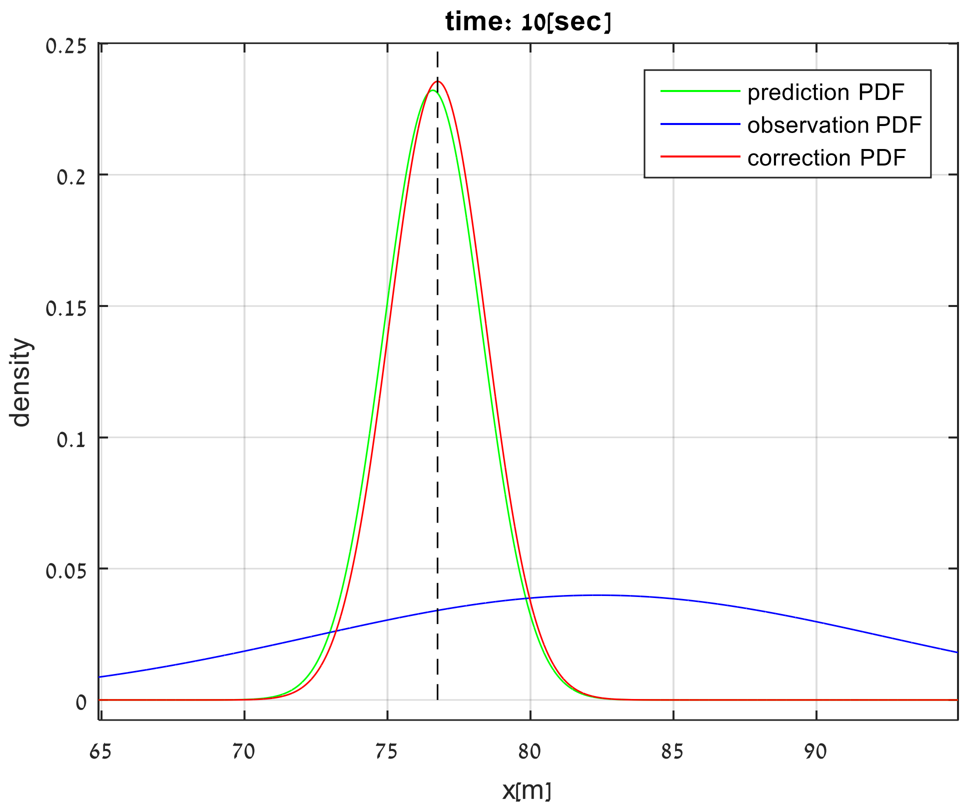

Figure 2.9: The pdf's of the prediction step estimate, observation, and correction step estimate at time t=10. The straight dashed line indicates the true position of the system.

In Fig. (2.9) we see that the correction step's mean coincides with the dashed line which represent the true position, and thus yielding a good estimate. As seen and discussed in the previous figure, the correction's pdf is far closer to that of the prediction than to the observation's pdf, and the variance of the correction step's pdf, is not noticeably smaller than that of the prediction step. Again, this is due to the ratio between the measurement uncertainty and the process's noise uncertainty which is manifested in a small Kalman gain.

Appendix

MATLAB code

Work Assignment 1 - Kalman Filter Implementation

```
clc; clear all; close all;
% read the given measurment file and assign the appropriate values to a
% time array 'Time' and observations array 'z'.
data=dlmread('data.txt');
Time=data(:,1);
z=data(:,2);

% Initializing Parameters
T=10;                % total simulation time in seconds
dt=0.1;              % time interval
a=1.5;               % process acceleration action
sigma_a2=.25;        % variance of process interferance i.e. process noise
sigma2=100;            % the sensor readings varance (100 for first simulation and 1
for the second)
I=eye(2);            % identity matrix of appropriate dimensions for later use in the
kalman algorithm.

x_real(:,1)=[0;0];
x(:,1)=[0;0];      % x0 - initial state
P=zeros(2);        % P0 - initial covariance

% Dinamic model deffinition according to Newtonian physics.
A=[1 dt; 0 1];
B=[dt^2/2; dt];
u=a;
C=[1 0];
R=[dt^4/4 dt^3/2; dt^3/2 dt^2]*sigma_a2;
Q=sigma2;
```

Section (b): moving average filter

```
%estimated position plot
figure(1);
plot(Time,z,'k',Time,smooth(z),'b'); grid on; hold on;
title('Position VS. Time');
xlabel('t[sec]'); ylabel('x[m]');
legend('Observations','Smoothed Observations','Location','SouthEast');

% estimated velocity plot
figure(2);
```

```
plot(Time(2:end),diff(z)/dt,'b',Time(2:end),diff(smooth(z))/dt,'c'); grid on; hold on;
title('Velocity VS. Time');
xlabel('t[sec]'); ylabel('$\dot{x}$[m]','Interpreter','latex');
legend('Velocity estimated by numerical derivations','Velocity estimated by numerical
derivations on smoothed data','Location','SouthEast');
```

Section (c): the true state of the model

```
k=1;
for t=dt:dt:T
    x_real(:,k+1)=A*x_real(:,k) + B*u + mvnrnd([0;0],R)';
    k=k+1;
end

% superimposing the true state's position on the moving average estimation
figure(1);
plot(Time,x_real(1,:),'r'); grid on; hold on;
legend('Observations','Smoothed Observations','Real position','Location','SouthEast');

% superimposing the true state's velocity on the moving average estimation
figure(2);
plot(Time(2:end),x_real(2,2:end),'r'); grid on; hold on;
legend('Velocity estimated by numerical derivations','Velocity estimated by numerical
derivations on smoothed data','Real velocity','Location','SouthEast');
```

section (d): Kalman filter simulation

```
k=1;
for t=dt:dt:T

% Predition
x(:,k+1)=A*x(:,k) + B*u ;       % state prediction (mean) x(k|k-1)
P = A*P*A' + R;                 % covarinace prediction P(k|k-1)

x_1_0(:,k)=x(:,k+1);% only for pdf's plot in section (e)
P_1_0(:,:,k)=P;     % only for pdf's plot in section (e)

% Correction/Filtering
S = C*P*C' + Q;                 % innovation covarince
K = P*C' / S;                   % Kalman Gain
y = z(k+1) - C*x(:,k+1);        % innovation measurement
x(:,k+1) = x(:,k+1) + K*(y) ;   % state filtered (mean) x(k|k)
P = (I-K*C)*P*(I-K*C)' + K*Q*K';%P = (I- K*C)*P  % covariance filtered P(k|k)

x_1_1(:,k)=x(:,k+1);% only for pdf's plot in section (e)
P_1_1(:,:,k)=P;     % only for pdf's plot in section (e)

k=k+1;
```

```
end
figure(1);
plot(Time,x(1,:),'g'); grid on; hold on;
legend('Observations','Smoothed Observations','Real position','position estimated by
Kalman Filter','Location','SouthEast');
```

section (e): pdf's plotting (apply only if sigma2=100)

```
for i=[0 5 10-dt]
Ind=find(Time==i);
prediction=gmdistribution(x_1_0(1,Ind),P_1_0(1,1,Ind));
posterior=gmdistribution(x_1_1(1,Ind),P_1_1(1,1,Ind));
observation=gmdistribution(z(Ind+1),sigma2);

xx=x(1,Ind)-30:.0001:x(1,Ind)+30;
figure;
plot(xx,pdf(prediction,xx'),'g',xx,pdf(observation,xx'),'b',xx,pdf(posterior,xx'),'r',
[x(1,Ind+1) x(1,Ind+1)],[0 1],'--k');
grid on; title(['time: ' num2str(i) '[sec]']); xlabel('x[m]'); ylabel('density');
legend('prediction PDF','observation PDF','correction PDF');
end
```

Velocity estimation between the two Kalman filter simulation

the '.mat' files in the zip fie should be in the same directory as this matlab file ('HW1_KF.m'). these files are the results of our previous simulations of sigma2=100 and sigma2=1.

```
x1=load('x1.mat');
x100=load('x100.mat');
x_r=load('x_real.mat');

x1=x1.x;
x100=x100.x;
x_r=x_r.x_real;

figure;
plot(Time(2:end),x_r(2,2:end),'r',Time(2:end),x100(2,2:end),'g',Time(2:end),x1(2,2:end
),'b'); grid on; hold on;
legend('Real velocity','velocity \sigma^2=100','velocity
\sigma^2=1','Location','SouthEast');
title('Velocity VS. Time');
xlabel('t[sec]'); ylabel('$\dot{x}$[m]','Interpreter','latex');
```