Detection and Estimation Theory

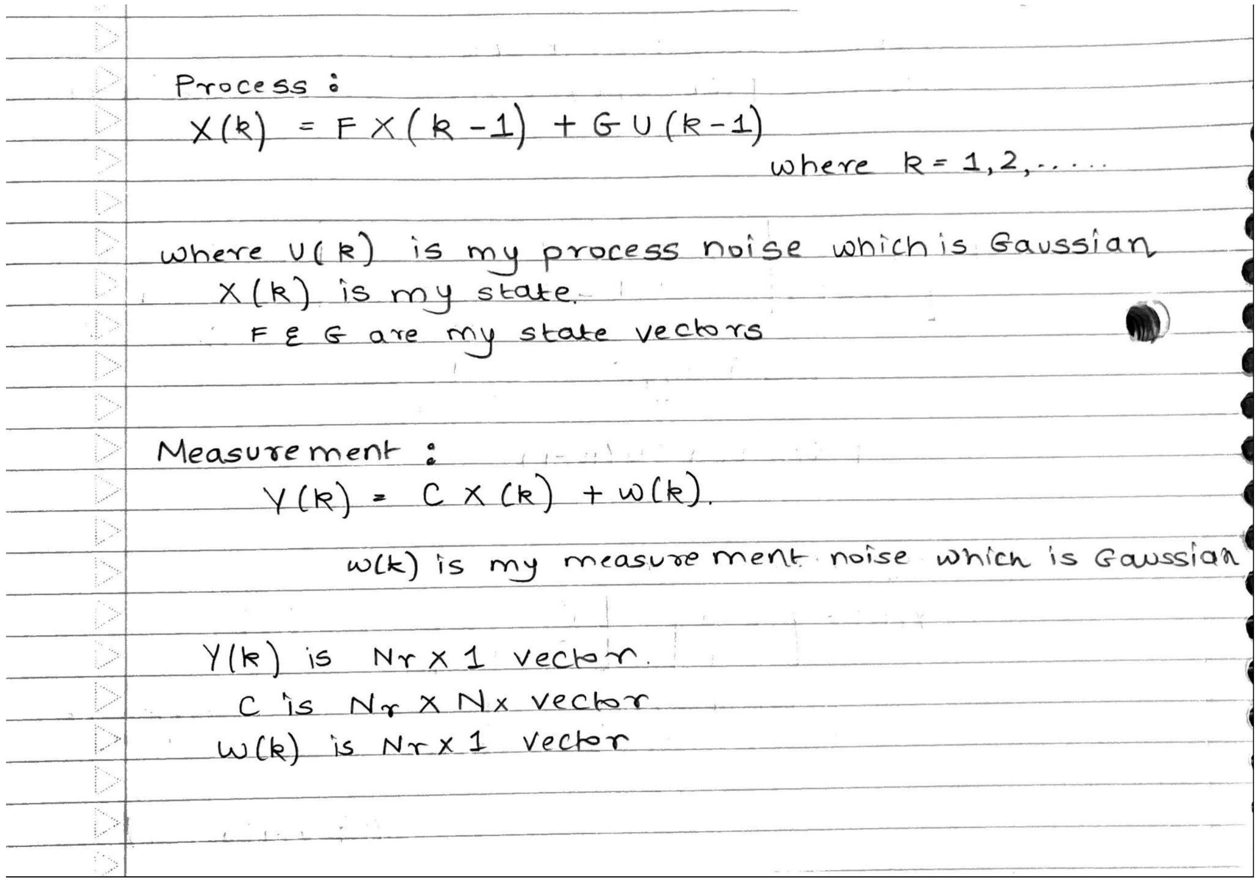
Software Project 1

-Tanisha Khurana

<u>Background:</u>

A Kalman filter state model is a mathematical model that describes the evolution of a system over time. It consists of a set of equations that describe the state of a system as a function of time, as well as the relationship between the system's state and the measurements obtained from it. The state model is used to predict the state of the system at each time step, while the measurements are used to update the predicted state. The state model can be used to estimate any system that can be modeled as a linear dynamic system with Gaussian noise.

This is the Kalman signal model where I have both process and measurement signal equations for estimation:



Process :

$X(k) = F X(k-1) + G U(k-1)$

where $k = 1, 2, \ldots$

where $U(k)$ is my process noise which is Gaussian

$X(k)$ is my state.

F & G are my state vectors

Measurement :

$Y(k) = C X(k) + w(k)$.

$w(k)$ is my measurement noise which is Gaussian

$Y(k)$ is $N_r \times 1$ vector.

C is $N_r \times N_x$ vector

$w(k)$ is $N_r \times 1$ vector

As an example we have used a motion model which is used to estimate the position P(0) and velocity S(0) of a moving body. And U(x) and w(x) are my process and measurement noise respectively.

Example of a roughly constant velocity motion model.

at $t = 0$.

$$s(t) = s_0 + at$$
$$p(t) = p_0 + s_0\tau + \frac{1}{2}a\tau^2$$

in a discrete space:

$$s(kT) = s((k-1)T) + a((k-1)T)T$$
$$p(kT) = p((k-1)T) + s((k-1)T)T + \frac{1}{2}a((k-1)T)T^2$$

$$X(k) = \begin{bmatrix} p(kT) \\ s(kT) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}}_{\text{this is my } F} \underbrace{\begin{bmatrix} p((k-1)T \\ s((k-1)T \end{bmatrix}}_{\substack{\text{feedback} \\ X(k-1)}} + \underbrace{\begin{bmatrix} 1/2\,T^2 \\ T \end{bmatrix}}_{G} \underbrace{a((k-1)T}_{U(k-1)}$$

$$= F\,X(k-1) + G\,U(k-1)$$

Here my $Y(k)$ for this example is

$$Y(k) = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\substack{C \text{ for observing} \\ \text{both position \& } \\ \text{velocity}}} \underbrace{\begin{bmatrix} P(kT) \\ S(kT) \end{bmatrix}}_{X(k)} + \underbrace{w(k)}_{\substack{\text{measurement} \\ \text{noise}}}$$

$$w(k) \sim N(0, \sigma_w^2)$$

Interface design:

In this code I have made a function `function [X,Y] = kalman_filter(N,F,G,C,sigma_w,sigma_a, X_0)` which takes in arbitrary F, G and C matrices as well as state vector initial conditions. The inputs to this function are N, F, G, C, sigma_w, sigma_a and X_0.

Here:

1. N is my total time period
2. F, G and C are my state vectors
3. sigma_a is the variance of my u(k) gaussian process noise
4. sigma_w is the variance of my w(k) gaussian measurement noise

T is my time interval.

My outputs are:
X and Y which are my state vectors for estimation results.

Benchmark example:

For the benchmark example of the velocity motion model, I have a set of 100 measurements ranging from 1 to 10 with an interval of 0.1.

Thus my T= 0.1, N =100 and my time ranges from `t = (1:N)*T;`

My model is initialized as:

p(0) = 1, 000 m,
s(0) = −50 m/s,
Where p(0) is the position and s(0) is the velocity and my x(k) is defined as [p(k), s(k)]
Sigma_a = 40
sigma_w = 100;

My state vectors for this motion model are:

```
C = [1 0 ; 0 1];
F = [1 T ;0 1];
G = [1/2 * (T^2); T];
```

The main code is as follows:
With k ranging from 1 to N-1 steps, I use the past values of X to generate the next value with the following equation:

```
for k = 1:N-1
% initialise equations
   X(:, k+1) = F * X(:,k) + G * a(k);
end
```

Conclusion:

Below are the results from the textbook (Detection, estimation, and modulation theory by Harry L. Van Trees) Example 9.10.

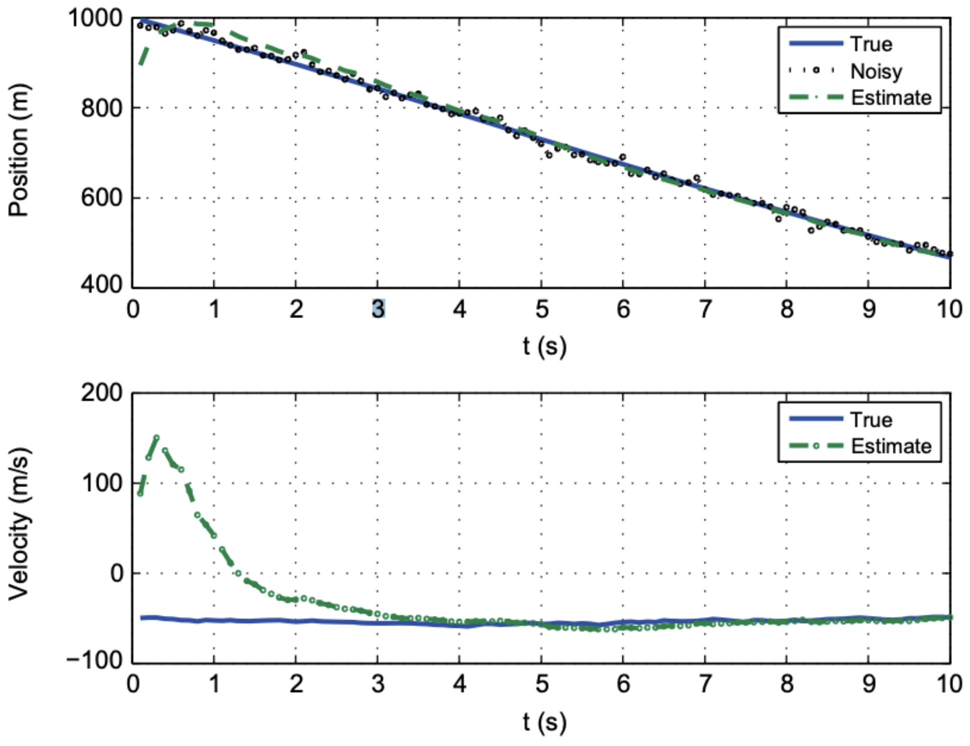
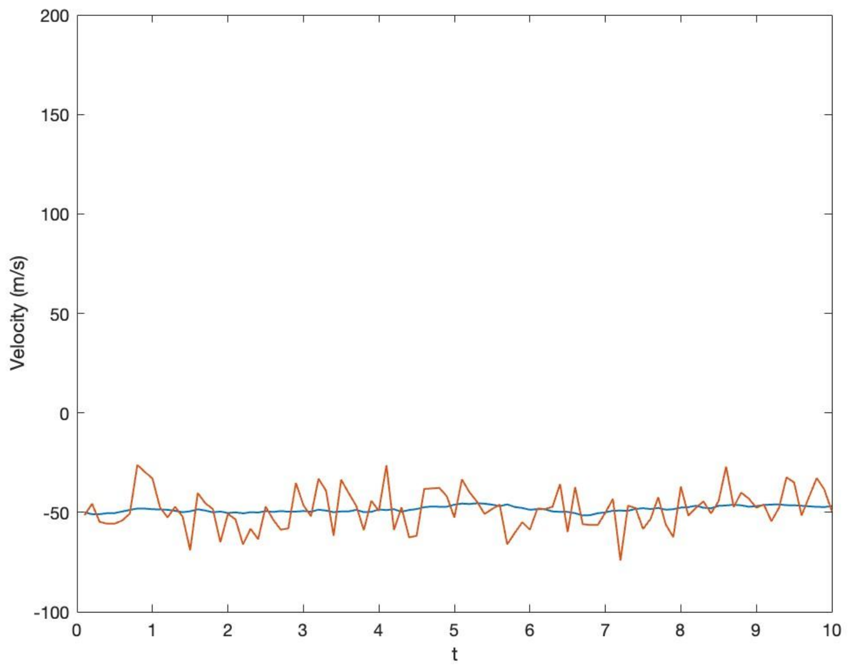
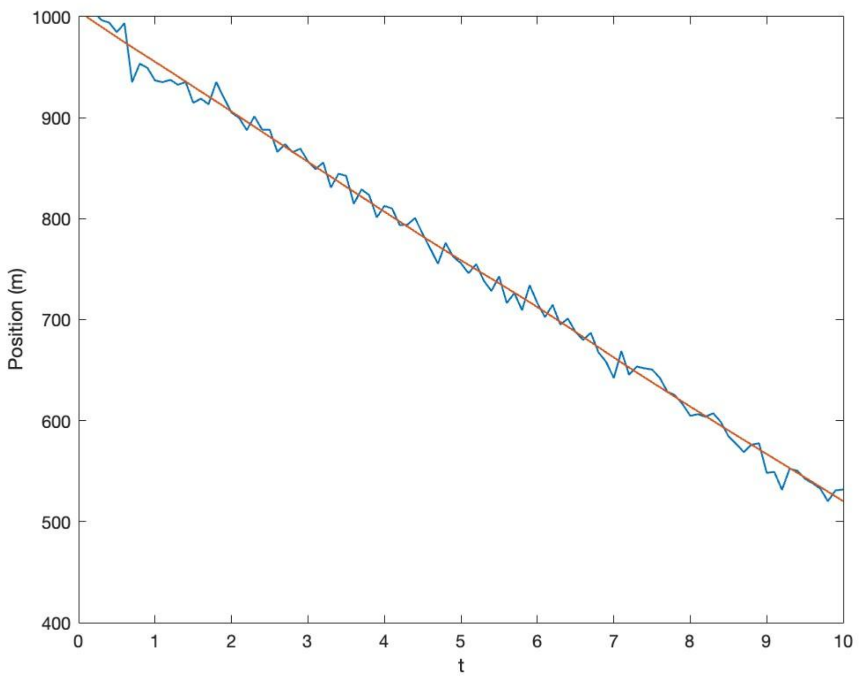


Figure 9.38: A realization of Kalman tracker with range-only observation.

Using the same parameters and initial conditions, below are my results from the function:





Thus, we can see that my results and the one from the textbook are almost similar.

Appendix:

Here is the attached code:

```
seed = 1344;
rng(seed,'twister');
clc;
clear;
close all;
% at t=0
% P = position
% S = speed
% T = sample time
% T = no. of samples
T = 0.1;
N = 100;
t = (1:N)*T;
% a = acceleration a ~ N(0,sigma)
sigma_a = 40;
%measurement variables
C = [1 0 ; 0 1];
sigma_w = 100;
% P_0 = 1000;
% S_0 = -50;
X_0 = [1000, -50];
%initialize state vectors
F = [1 T ;0 1];
G = [1/2 * (T^2); T];
% X = zeros(2,T);
[X, Y] = kalman_filter(N,F,G,C,sigma_w,sigma_a, X_0);
figure;
plot(t,Y(1,:),'LineWidth',1);
hold on;
plot(t,X(1,:),'LineWidth',1);
ylim([400 1000])
xlabel('t');
ylabel('Position (m)');
figure;
plot(t,X(2,:),'LineWidth',1);
hold on;
plot(t,Y(2,:),'LineWidth',1);
ylim([-100 200])
xlabel('t');
ylabel('Velocity (m/s)');
function [X,Y] = kalman_filter(N,F,G,C,sigma_w,sigma_a, X_0)
Nx = size(F,1);
Nr = size(C,1);
X = zeros(Nx, N);
```

```
X(:,1) =X_0;
Y = zeros(Nr,N);
w = normrnd(0,sigma_w ^0.5,[Nr,N]);
a = normrnd(0,sigma_a ^0.5,[Nx,N]);
for k = 1:N-1
% initialise equations
   X(:, k+1) = F * X(:,k) + G * a(k);
end
Y = C * X + w;
end
```