Kalman Filter Construction

전기공학전공 홍종혁

전체 코드(해답)

```
clear;clc;close all

%% Modeling
F=[1 0 0.5 0;0 1 0 0.5;0 0 1 0;0 0 0 1];
FT=F.';
H=[1 0 0 0;0 1 0 0];
HT=H.';
P=eye(4);
R=100;
%% Original Data Set
x=zeros(4,100);
x(:,1)=[3;0;1;0];
%% Measurement Data Set
tri1=[20 20;-20 0];
noisy=zeros(2,100);
%% Filtered Data Set
xhat=zeros(4,100);
xhat(:,1)=[3;0;1;0];

%% Original Data
for i=1:1:99
    x(:,i+1)=F*x(:,i)+randn(4,1);
end
plot(x(1,:),x(2,:),"-o","Color",[0 0 0])
hold on

%% Measurement data
for i=1:1:100
    r1=normrnd(0,0.3)+sqrt(x(1,i)^2+x(2,i)^2);
    r2=normrnd(0,0.3)+sqrt((x(1,i)-10)^2+(x(2,i)-10)^2);
    r3=normrnd(0,0.3)+sqrt(x(1,i)^2+(x(2,i)-10)^2);
    tri2=[(r1^2)-(r2^2)+200;(r2^2)-(r3^2)-100];
    input=tri1\tri2;
    noisy(:,i)=input;
end
plot(noisy(1,:),noisy(2,:),"-x")
hold on

%% Kalman Filtering
for i=1:1:99
    xhat(:,i+1)=F*xhat(:,i);
    P=F*P*FT + eye(4);
    K=(P*HT)/((H*P*HT+R*eye(2)));
    xhat(:,i+1)=xhat(:,i+1)+K*(noisy(:,i+1)-H*xhat(:,i+1));
    P=P-K*H*P;
end
plot(xhat(1,:),xhat(2,:),"-x","Color","b")
```

알고리즘 설명 1 – Measurement Data의 생성

```
%% Modeling
F=[1 0 0.5 0;0 1 0 0.5;0 0 1 0;0 0 0 1];
FT=F.';
H=[1 0 0 0;0 1 0 0];
HT=H.';
P=eye(4);
r=input('R? : ');
```

Kalman Filtering에 필요한 Model들을 설정하였다. 변수 'r'은 Covariance Matrix R의 값 변화에 따른 Filter 성능 변화를 살펴보기 위해서 입력 받는 것이고, 보통은 최적의 값으로 설정된다.

```
%% Original Data Set
x=zeros(4,100);
x(:,1)=[3;0;1;0];
%% Measurement Data Set
tri1=[20 20;-20 0];
noisy=zeros(2,100);
%% Filtered Data Set
xhat=zeros(4,100);
xhat(:,1)=[3;0;1;0];
```

원본 Data와 Noised Data, Filtered Data의 틀을 생성해준다.

```
%% Original Data
for i=1:1:99
    x(:,i+1)=F*x(:,i)+randn(4,1);
end
plot(x(1,:),x(2,:),"-o","Color",[0 0 0])
hold on
```

원본 Data와 Measurement Data, Filtered Data의 틀을 생성해준다.

```
%% Original Data
for i=1:1:99
    x(:,i+1)=F*x(:,i)+randn(4,1);
end
plot(x(1,:),x(2,:),"-o","Color",[0 0 0])
hold on
```

원본 Data를 plot 해준다. randn(4,1)는 N(0,I)를 따르는 랜덤 난수이다.

```
%% Measurement data
for i=1:1:100
    r1=normrnd(0,0.3)+sqrt(x(1,i)^2+x(2,i)^2);
    r2=normrnd(0,0.3)+sqrt((x(1,i)-10)^2+(x(2,i)-10)^2);
    r3=normrnd(0,0.3)+sqrt(x(1,i)^2+(x(2,i)-10)^2);
    tri2=[(r1^2)-(r2^2)+200;(r2^2)-(r3^2)-100];
    input=tri1\tri2;
    noisy(:,i)=input;
end
plot(noisy(1,:),noisy(2,:),"-x")
hold on
```

Trilateration을 이용하여 Data를 생성한다. 이 때 $N(0, 0.3^2)$를 따르는 Noise가 측정시에 더해진다.

```
%% Kalman Filtering
for i=1:1:99
    xhat(:,i+1)=F*xhat(:,i);
    P=F*P*FT + eye(4);
    K=(P*HT)/((H*P*HT+R*eye(2)));
    xhat(:,i+1)=xhat(:,i+1)+K*(noisy(:,i+1)-H*xhat(:,i+1));
    P=P-K*H*P;
end
plot(xhat(1,:),xhat(2,:),"-x","Color","b")
```

for문을 이용하여, Kalman Filtering 과정을 수행하고 그 값을 xhat에 저장하도록 하였다. 수식으로 설명하자면,

$$\hat{x}_{k+1}^{-} = \mathbf{F}\,\hat{x}_k$$

$$\mathbf{P}_{k+1}^{-} = \mathbf{F}\mathbf{P}_k^{-}\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$$

$$\mathbf{K}_{k+1} = (\mathbf{P}_{k+1}^{-}\mathbf{H}^{\mathrm{T}})(\mathbf{H}\mathbf{P}_{k+1}^{-}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + \mathbf{K}_{k+1}(x_{k+1} - \mathbf{H}\hat{x}_{k+1}^{-})$$

$$\mathbf{P}_{k+1} = \mathbf{P}_{k+1}^{-} - \mathbf{K}_{k+1}\boldsymbol{H}\mathbf{P}_{k+1}^{-}$$

를 for문을 활용해서 반복하는 것이다.

알고리즘 설명 2 – Kalman Filtering 최적의 R 값 찾기

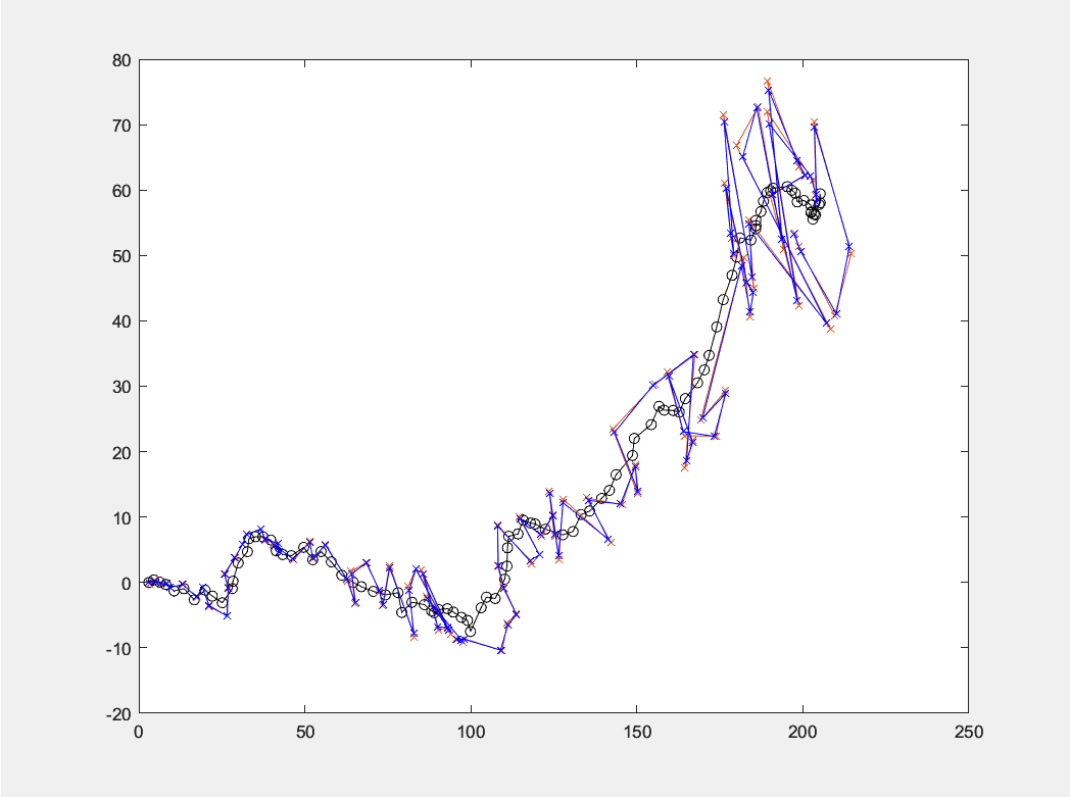
코드를 약간 수정하여, 최적의 R 값에 가장 가까운 값을 찾아보고자 했다.

```
%% Kalman Filtering
r=[0.1 1 10 100 1000 10000 100000 1000000];
for i=1:1:7
    figure('Position', [500 100 840 630])
    plot(x(1,:),x(2,:),"-o","Color",[0 0 0])
    hold on
    plot(noisy(1,:),noisy(2,:),"-x")
    hold on
    kalman(F,FT,H,HT,P,noisy,r(i))
    waitfor(gcf, 'CloseRequestFcn');
end
function kalman(F,FT,H,HT,P,noisy,r)
xhat=zeros(4,100);
xhat(:,1)=[3;0;1;0];
for i=1:1:99
    xhat(:,i+1)=F*xhat(:,i);
    P=F*P*FT + eye(4);
    K=(P*HT)/((H*P*HT+r*eye(2)));
    xhat(:,i+1)=xhat(:,i+1)+K*(noisy(:,i+1)-H*xhat(:,i+1));
    P=P-K*H*P;
end
plot(xhat(1,:),xhat(2,:),"-x","Color","b")
end
```

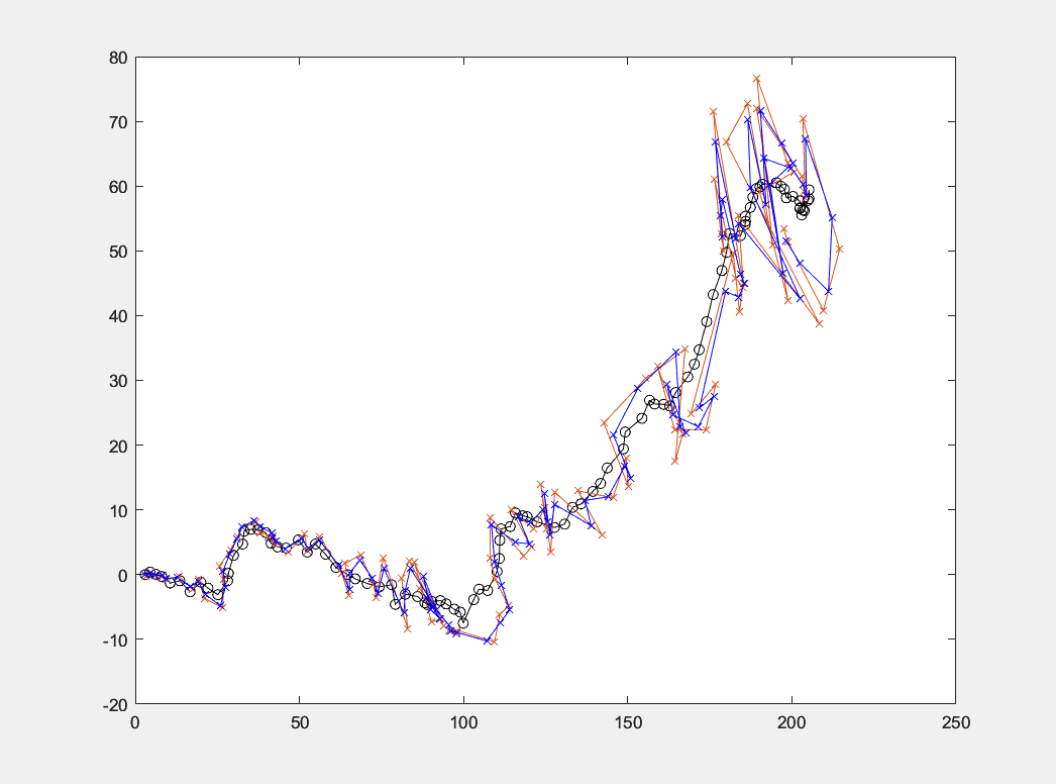
10의 거듭제곱 값들을 활용하여 반복시행 하였다. 원본 Data와 Measurement Data는 고정해 놓고, Kalman Filtered Data만 변화하도록 하여 비교할 수 있도록 하였다. 결과는 다음과 같았다.

결과

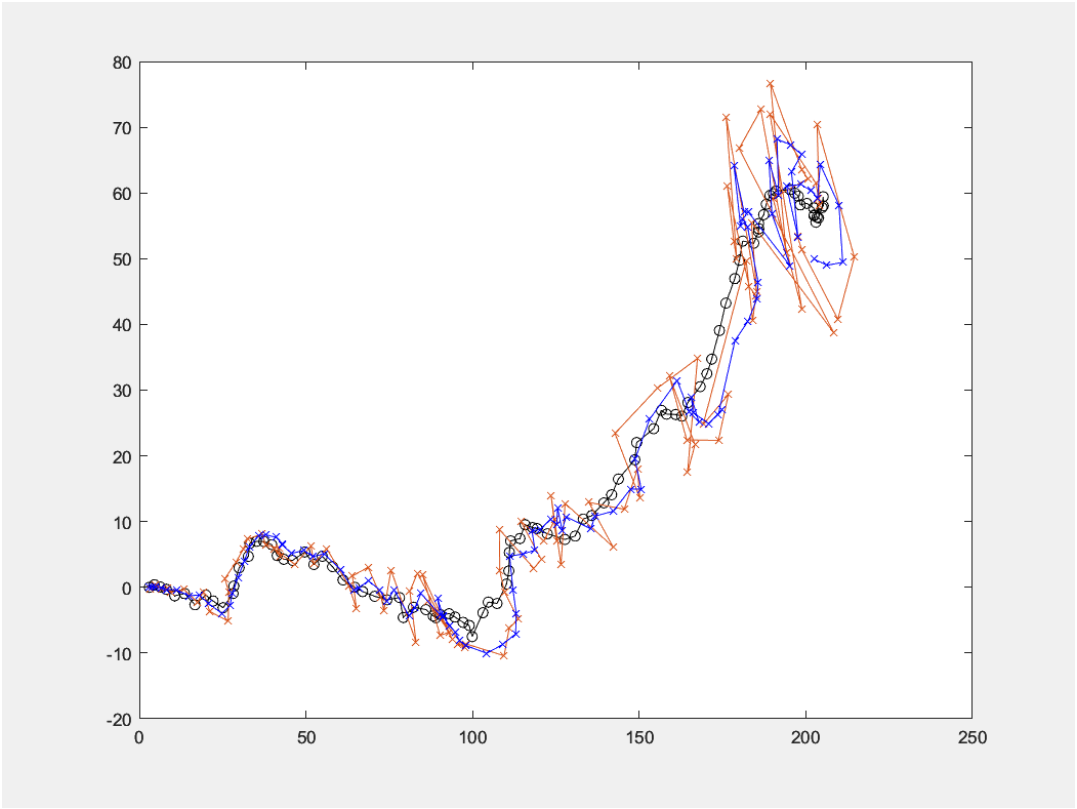
(1) R = 0.1


80
70
60
50
40
30
20
10
0
-10
-20
0
50
100
150
200
250

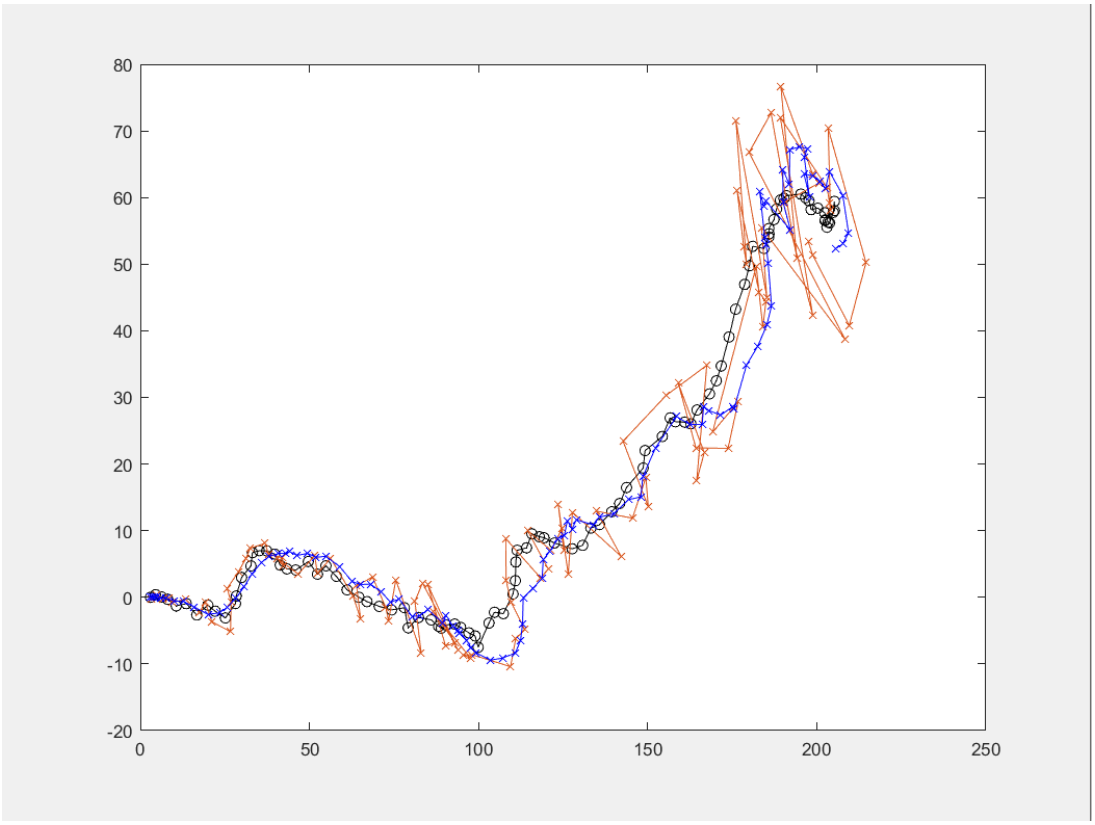
(2) R = 1


80
70
60
50
40
30
20
10
0
-10
-20
0
50
100
150
200
250

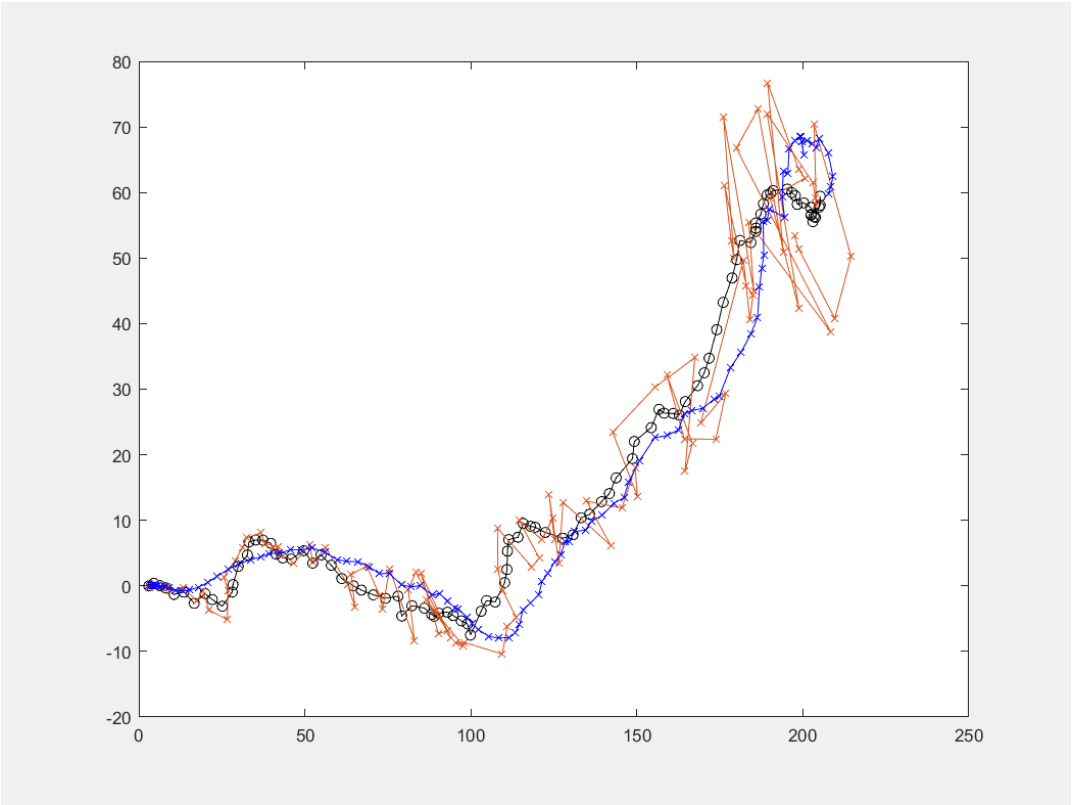
(3) R = 10


80
70
60
50
40
30
20
10
0
-10
-20
0
50
100
150
200
250

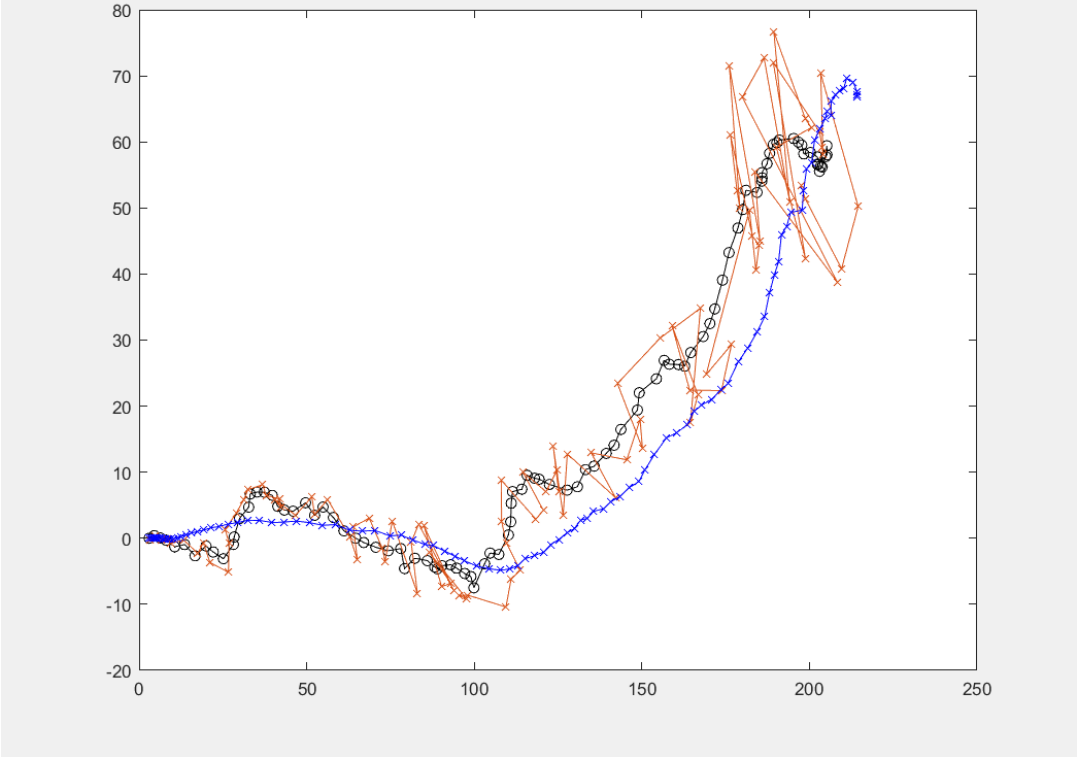
(4) R = 100


80
70
60
50
40
30
20
10
0
-10
-20
0
50
100
150
200
250

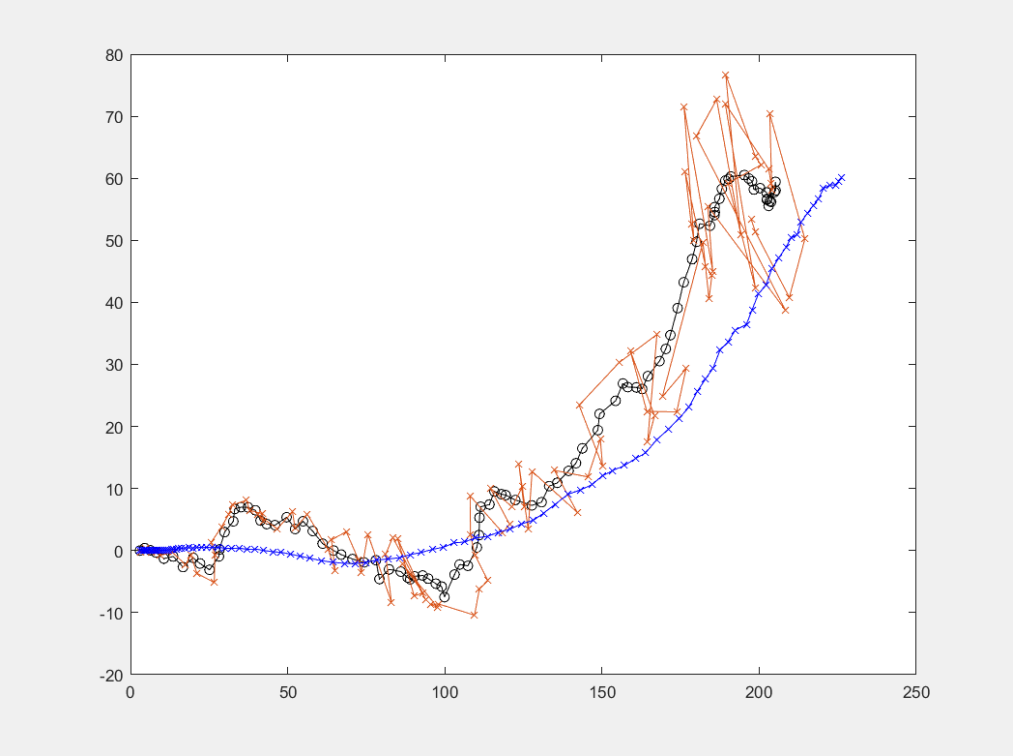
(5) R = 1000



(6) R = 10000



(7) R = 100000



그래프의 개형을 보았을 때, R의 값이 너무 작거나 너무 크면, Filter가 제 기능을 하지 못하는 것을 확인할 수 있었다. 이외에도 여러 번의 시행을 거친 결과, Measurement Data의 개형이 복잡할 경우, 분산이 클 때 Filter가 제대로 작동하지 않는 경향이 더 크다는 것을 알 수 있었다. 물론 개형과 관계없이 대체적으로 너무 크거나 작다면 Filter가 제대로 작동하지 않았다.

따라서 10~1000사이의 R 값을 선택해야 Filter가 제대로 작동할 확률이 크다는 것을 알 수 있었다. 제출한 코드의 R 값은 log_{10}scale에서 1과 3 사이의 값인 2, 즉 100으로 선택하였지만, 적합한 R값은 매 시행마다 달라질 것이다.

참고문헌

https://ko.wikipedia.org/wiki/%EC%B9%BC%EB%A7%8C_%ED%95%84%ED%84%B0