Assignment 5 Tracking of a moving object which trajectory is disturbed by random acceleration

Group 6:

- **Andrei Shumeiko**
- **Ayush Gupta**
- **Olga Klyagina**
- **Simona Nitti**

```
clc;
clear;
close all;
```

Task 1: Generating a true trajectory

```
n = 200;
x = 5*ones(1,n);
V = ones(1,n);
T = 1;
var_a = 0.2^2;
a = sqrt(var_a)*randn(1,n);

for i = 2:n
    x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
    V(i) = V(i-1)+a(i-1)*T;
end
```

A true trajectory for a moving object is created with random (noise) acceleration and given initial conditions.

Task 2: Generating measurements of the process

```
var_eta = 20^2;
eta = sqrt(var_eta)*randn(1,n);
z = eta+x; % measurements of coordinate x
```

The measurements with true trajectory and normally distributed random noise is created.

Task 3: Present the system at state space

```
phi = [1 T; 0 1]; % transition matrix that relates X(i) and X(i-1);
G = [T^2/2 T].'; % input matrix, that determines how random acceleration affects state vector;
H = [1 0]; % observation matrix
```

State space representation of the transition matrix, input matrix and observation matrix are initialized.

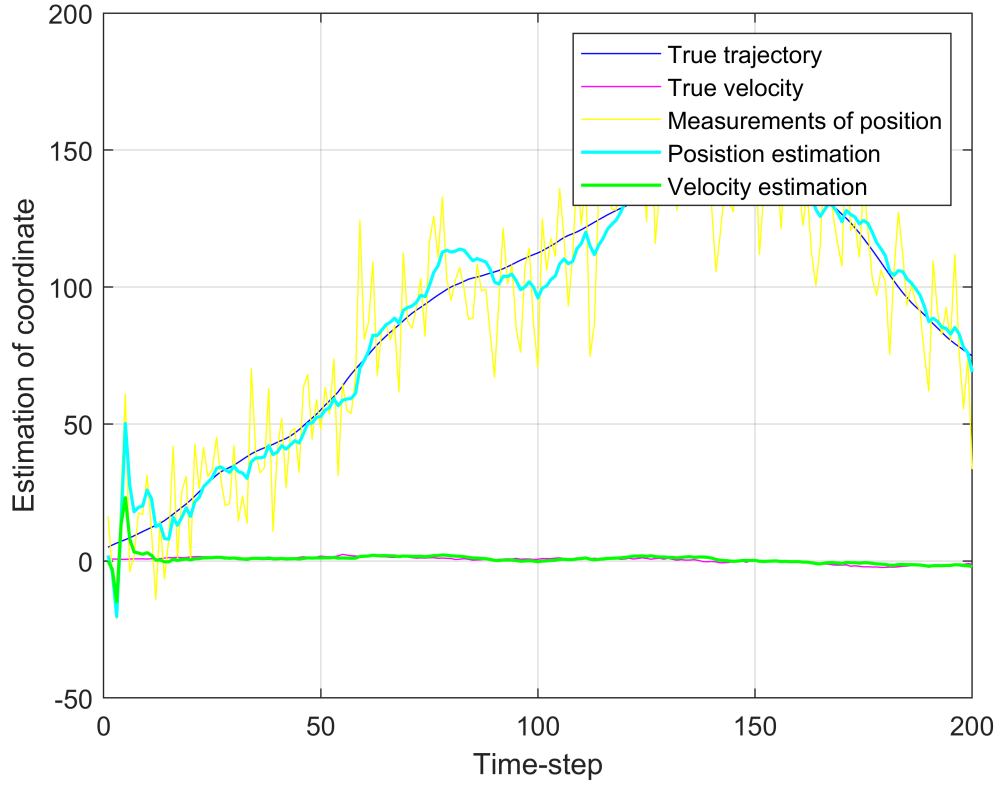
Task 4: Kalman filter Algorithm

```
X(:,1) = [2 0].';
P = [1e4 0; 0 1e4];
sigma_x = sqrt(P(1,1))*ones(1,n);
R = var_eta;
Q = G*G'*var_a;

K = zeros(2,n);
for i = 2:n
    X(:,i) = phi*X(:,i-1);
    P_pred = phi*P*phi.'+Q;

    K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
    X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
    P = (eye(2)-K(:,i)*H)*P_pred;
    sigma_x(i) = sqrt(P(1,1));
end
K(:,1)= [NaN,NaN];
```

Kalman filter algorithm consits of extrapolation of data and then filtration of the data. In the extrapolation stage the prediction of the state vector is done at the time (t) by the use of the (i-1) measurements. For instance, the new predicted value has a prevoious measurement's value as input. The prediction error covariance matrix is also calculated. In the filtration stage the improved estimate is calculated by taking new measurement into account. The filter gain and filtration error covariance matrix are also calculated.

Task 5: Plot results (true trajectory, measurements and filter estimates)

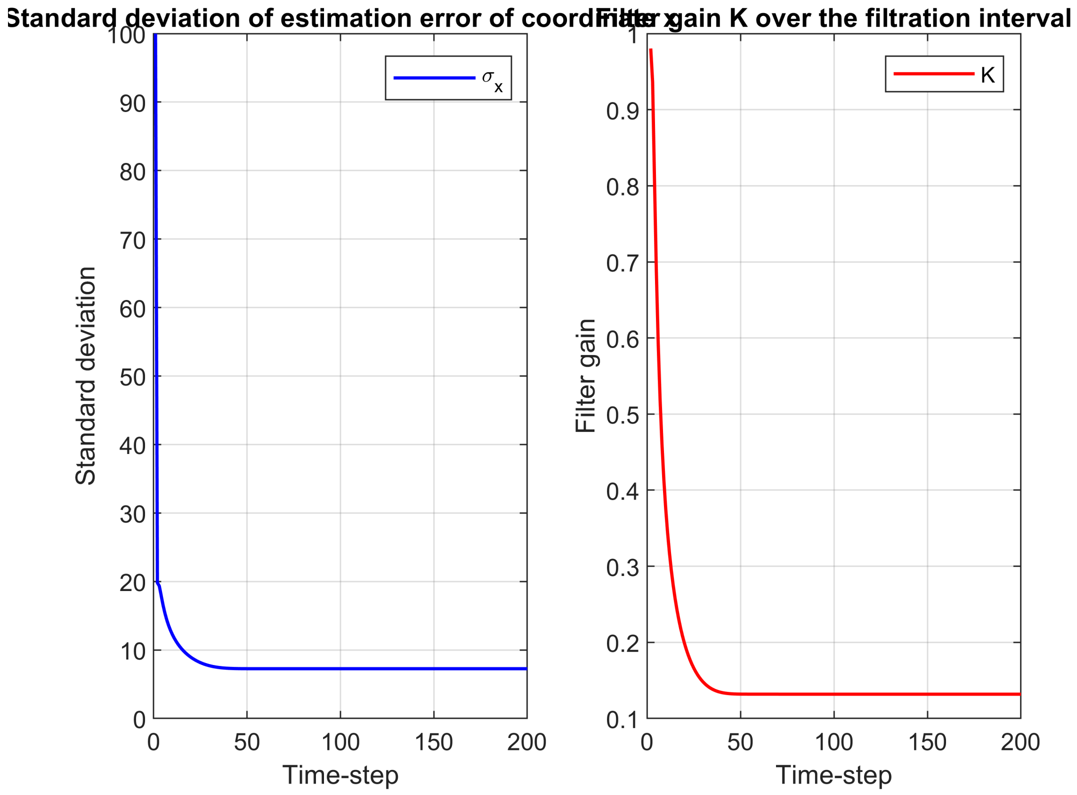
```
figure
plot(x,'b')
hold on
plot(V,'m')
plot(z,'y')
plot(X(1,:),'c','linewidth',1.2)
plot(X(2,:),'g','linewidth',1.2)
xlabel('Time-step')
ylabel('Estimation of coordinate')
legend('True trajectory','True velocity','Measurements of position',...
    'Posistion estimation','Velocity estimation')
grid on
```



True trajectory, true velocity, measurements and estimation of both position and velocity are plotted. As the filter is ran serveral times the estimation results changes. We can observe from the graph that the estimation of positions at the initilal conditions tends to fluctuate a lot as the filter needs some initial data to provide better estimations.

Task 6: Plot filter gain K and standard deviation of estimation error of coordinate x(i)

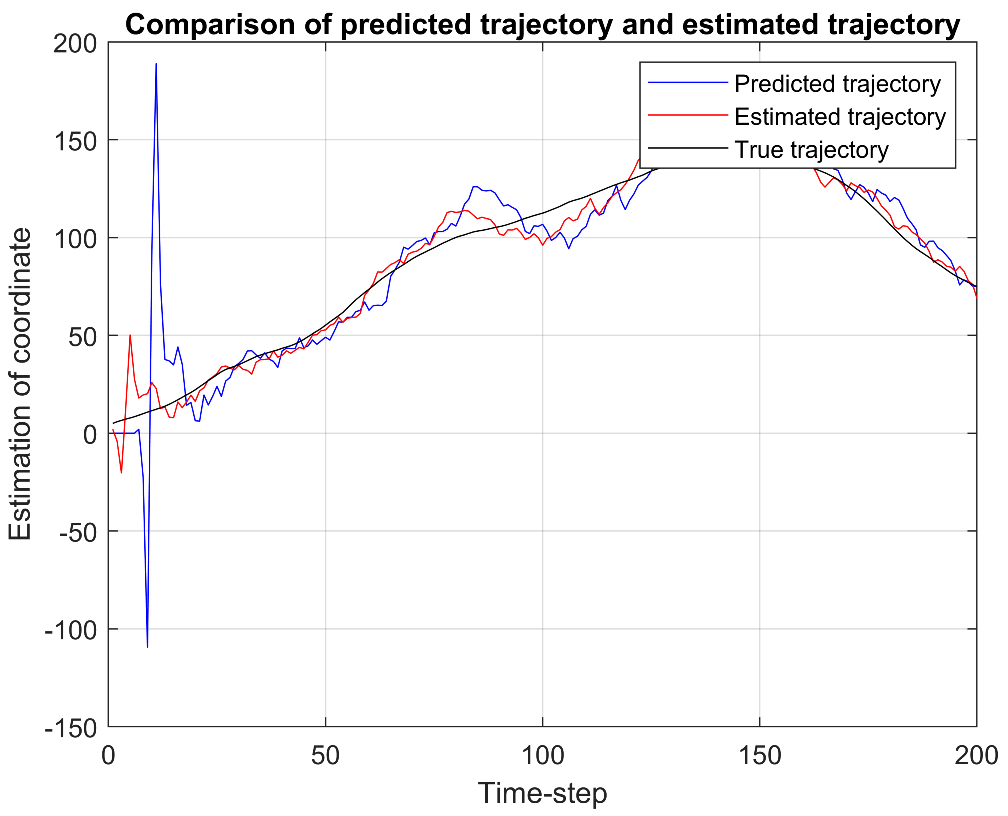
```
figure
subplot(1,2,1)
plot(sigma_x,'b','linewidth',1.2)
xlabel('Time-step')
ylabel('Standard deviation')
legend('\sigma_x')
title('Standard deviation of estimation error of coordinate x')
grid on

subplot(1,2,2)
plot(K(1,:),'r','linewidth',1.2)
xlabel('Time-step')
ylabel('Filter gain')
grid on
legend('K')
title('Filter gain K over the filtration interval')
```



The standard deviation of estimation of x(i) becomes constant at 7.2626 and filtration gain K at 0.1319. We can not estimate more than estimated limit of accuracy due to uncertainity.

Task 7: Extrapolation on m = 7 steps ahead

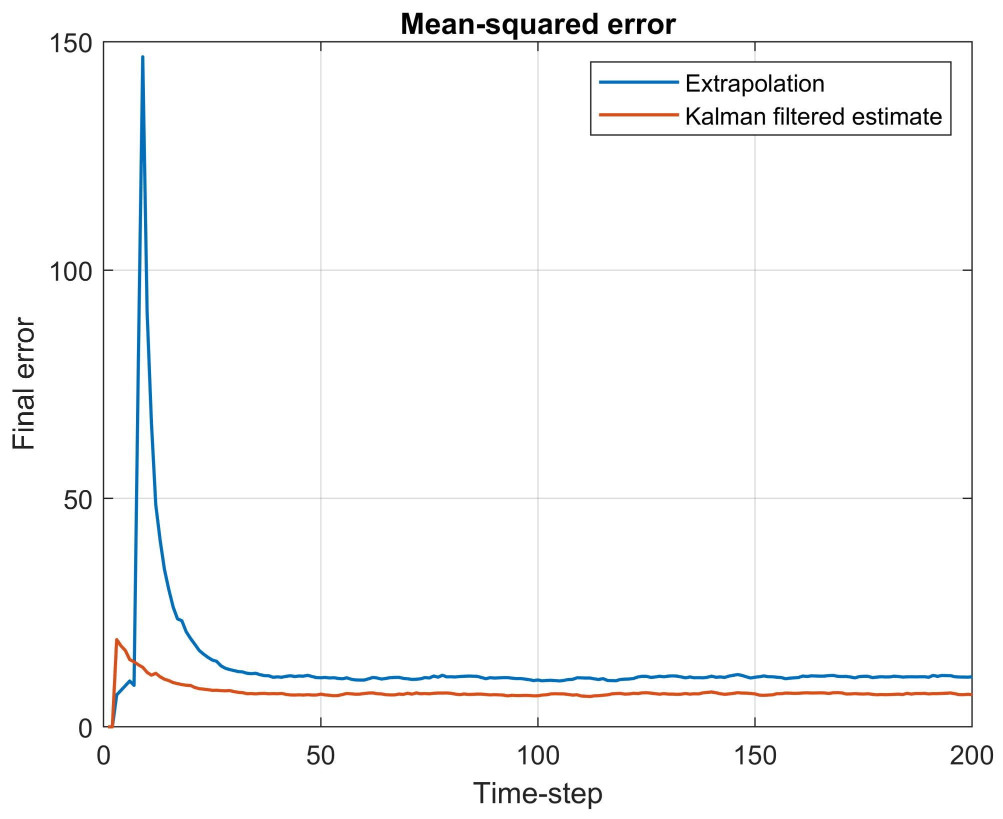
```
m = 7;
X_pred = zeros(2,n);
for i = 1:n-m+1
    X_pred(:,i+m-1) = phi^(m-1)*X(:,i);
end

figure
plot(X_pred(1,:),'b')
hold on
plot(X(1,:),'r')
plot(x,'k')
xlabel('Time-step')
ylabel('Estimation of coordinate')
title('Comparison of predicted trajectory and estimated trajectory')
legend('Predicted trajectory','Estimated trajectory','True trajectory')
grid on
```



In this case extrapolation of the estimated trajectory is done with a step size of 7. The predicted trajectory fluctuates at the initial time steps but then converges with the estimated values when more data is fed to it.

Task 8:Estimation of dynamics of mean-squared error of estimation over observation interval

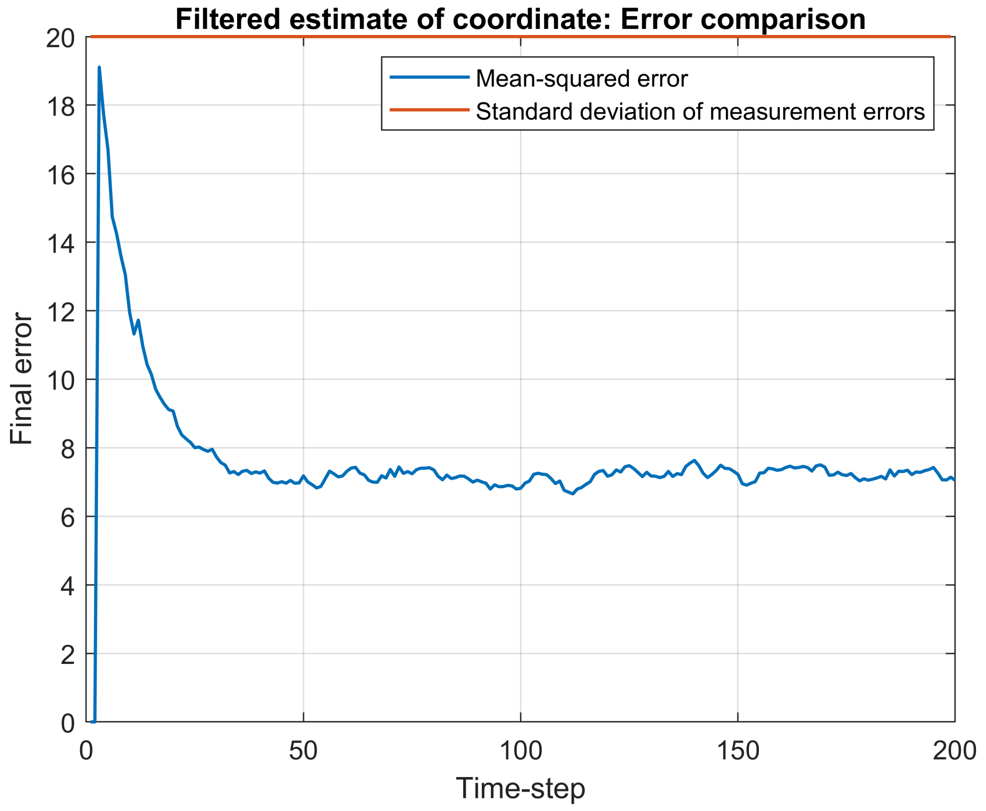
```
M = 500;

Final_Error_estr = zeros(1,n);
error_estr = zeros(1,n);
Final_Error_filt = zeros(1,n);
error_filt = zeros(1,n);
for run = 1:M
    x = 5*ones(1,n);
    V = ones(1,n);
    a = sqrt(var_a)*randn(1,n);

    for i = 2:n
        x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
        V(i) = V(i-1)+a(i-1)*T;
    end

    var_eta = 20^2;
    eta = sqrt(var_eta)*randn(1,n);
    z = eta+x;

    X(:,1) = [2 0].';
    P = [1e4 0; 0 1e4];
    sigma_x = sqrt(P(1,1)*ones(1,n));
    R = var_eta;
    Q = G*G.'*var_a;

    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi.'+Q;

        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x(i) = sqrt(P(1,1));
    end

    m = 7;
    for i = 1:n-m+1
        X_pred(:,i+m-1) = phi^(m-1)*X(:,i);
    end

    for i = 3:n
        error_estr(i) = (x(1,i)-X_pred(1,i))^2;
        Final_Error_estr(i)= Final_Error_estr(i)+error_estr(i);
    end

    for i = 3:n
        error_filt(i) = (x(1,i)-X(1,i))^2;
        Final_Error_filt(i)= Final_Error_filt(i)+error_filt(i);
    end
end
Final_Error_estr = sqrt(Final_Error_estr/(M-1));
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

figure
plot(Final_Error_estr, 'linewidth',1.2)
hold on
plot(Final_Error_filt,'linewidth',1.2)
grid on
title(' Mean-squared error ')
legend('Extrapolation','Kalman filtered estimate')
ylabel('Final error')
xlabel('Time-step')
```



Making the filter run for 500 iterations, the graphs for averaged final error and and filter estimates are plotted. The final error becomes constant after the time step of 40 at 10.8858. The obervation steps are taken from the 3rd step as for 1st and 2nd the errors can be huge, for better convergence of the model we ignore the 1st and 2nd iterates.

Task 9: Compare mean-squared error with standard deviation of measurement errors

```
figure
plot(Final_Error_filt,'linewidth',1.2)
hold on
plot(sqrt(var_eta)*ones(1,199),'linewidth',1.2)
grid on
title('Filtered estimate of coordinate: Error comparison')
legend('Mean-squared error','Standard deviation of measurement errors')
ylabel('Final error')
xlabel('Time-step')
```

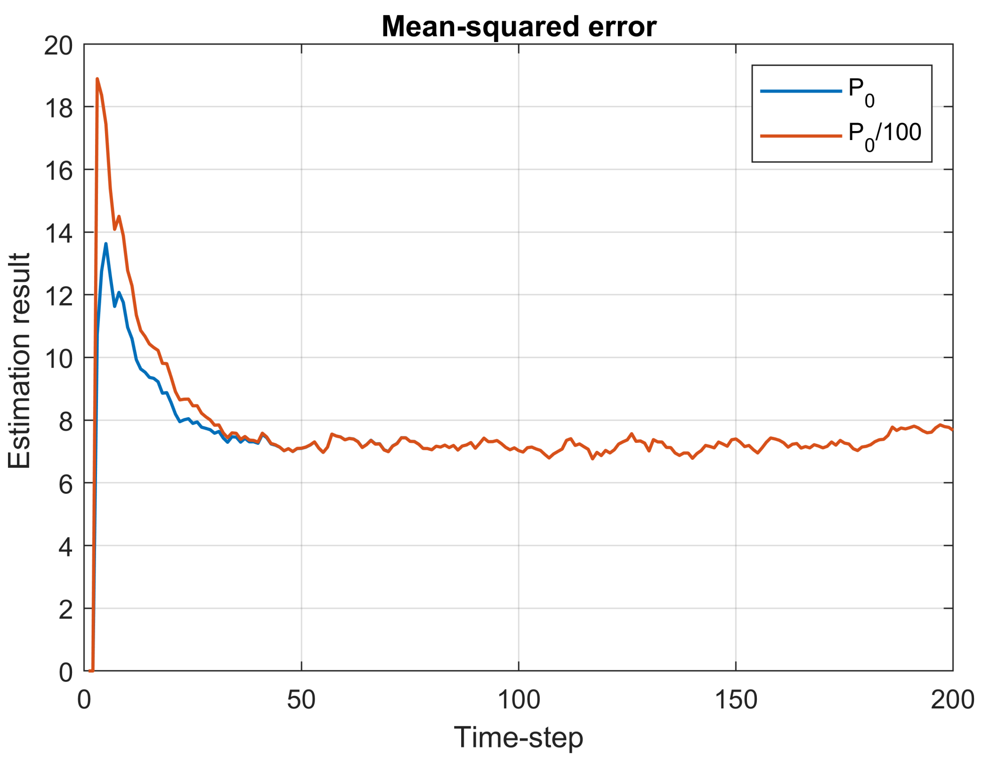


The graph between the final error estimate and the standard deviation of measurement errors is plotted which shows that the initial error of measurements are on a level of 20 and the filtered results at a level of 7. So, the final error is roughly 3 times smaller than the error of initial measurements.

Task 10: Change in the initial values of error covariance matrix P

```
M = 500;
Final_Error_filt_2 = zeros(1,n);
error_filt_2 = zeros(1,n);
Final_Error_filt = zeros(1,n);
error_filt = zeros(1,n);
for run = 1:M
    x = 5*ones(1,n);
    V = ones(1,n);
    a = sqrt(var_a)*randn(1,n);

    for i = 2:n
        x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
        V(i) = V(i-1)+a(i-1)*T;
    end

    var_eta = 20^2;
    eta = sqrt(var_eta)*randn(1,n);
    z = eta+x;

    X_10(:,1) = [2 0].';
    X_(:,1) = [2 0].';
    P = [1e2 0; 0 1e2];
    sigma_x = sqrt(P(1,1)*ones(1,n));
    R = var_eta;
    Q = G*G.'*var_a;

    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi.'+Q;

        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x(i) = sqrt(P(1,1));
    end

    for i = 3:n
        error_filt(i) = (x(1,i)-X(1,i))^2;
        Final_Error_filt(i)= Final_Error_filt(i)+error_filt(i);
    end

    P = [1e4 0; 0 1e4];
    sigma_x_2 = sqrt(P(1,1)*ones(1,n));

    for i = 2:n
        X_10(:,i) = phi*X_10(:,i-1);
        P_pred = phi*P*phi.'+Q;

        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
        X_10(:,i) = X_10(:,i) + K(:,i)*(z(i)-H*X_10(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x_2(i) = sqrt(P(1,1));

    end

    for i = 3:n
        error_filt_2(i) = (x(1,i)-X_10(1,i))^2;
        Final_Error_filt_2(i)= Final_Error_filt_2(i)+error_filt_2(i);
    end
end
Final_Error_filt_2 = sqrt(Final_Error_filt_2/(M-1));
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

figure
subplot(1,1,1)
sgtitle('Data comparison for different initial filtration error covariance matrix P_0')
plot(Final_Error_filt, 'linewidth',1.2)
hold on
plot(Final_Error_filt_2,'linewidth',1.2)
grid on
title('Mean-squared error')
legend('P_0','P_0/100')
ylabel('Estimation result ')
xlabel('Time-step')
```

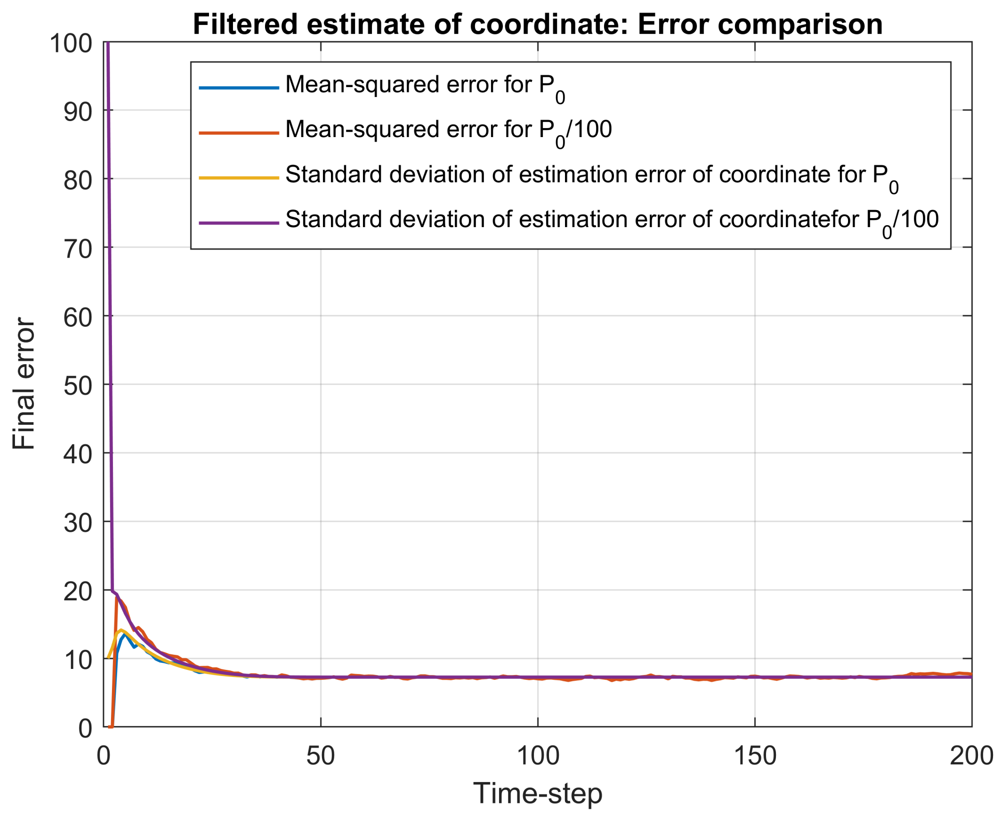
a comparison for different initial filtration error covariance matrix



For the change in the initial conditions of the error covariance matrix P_0 is more accuarte than P_0/100. The mean squared error stabilizes faster in this case. The accuracy of filtration directly depends upon the error covariance state space. For the change in the initial conditions of the P_0 the error of the P_0 is smaller than the P_0/100.

Task 11: Compare calculation errors of estimation Pi,i provided Kalman filter algorithm with true estimation errors

```
figure
plot(Final_Error_filt,'linewidth',1.2)  %a
hold on
plot(Final_Error_filt_2,'linewidth',1.2)
plot(sigma_x,'linewidth',1.2) %b
plot(sigma_x_2,'linewidth',1.2)
grid on

title('Filtered estimate of coordinate: Error comparison')
legend('Mean-squared error for P_0','Mean-squared error for P_0/100',...
    'Standard deviation of estimation error of coordinate for P_0',...
    'Standard deviation of estimation error of coordinatefor P_0/100')
ylabel('Final error')
xlabel('Time-step')
```

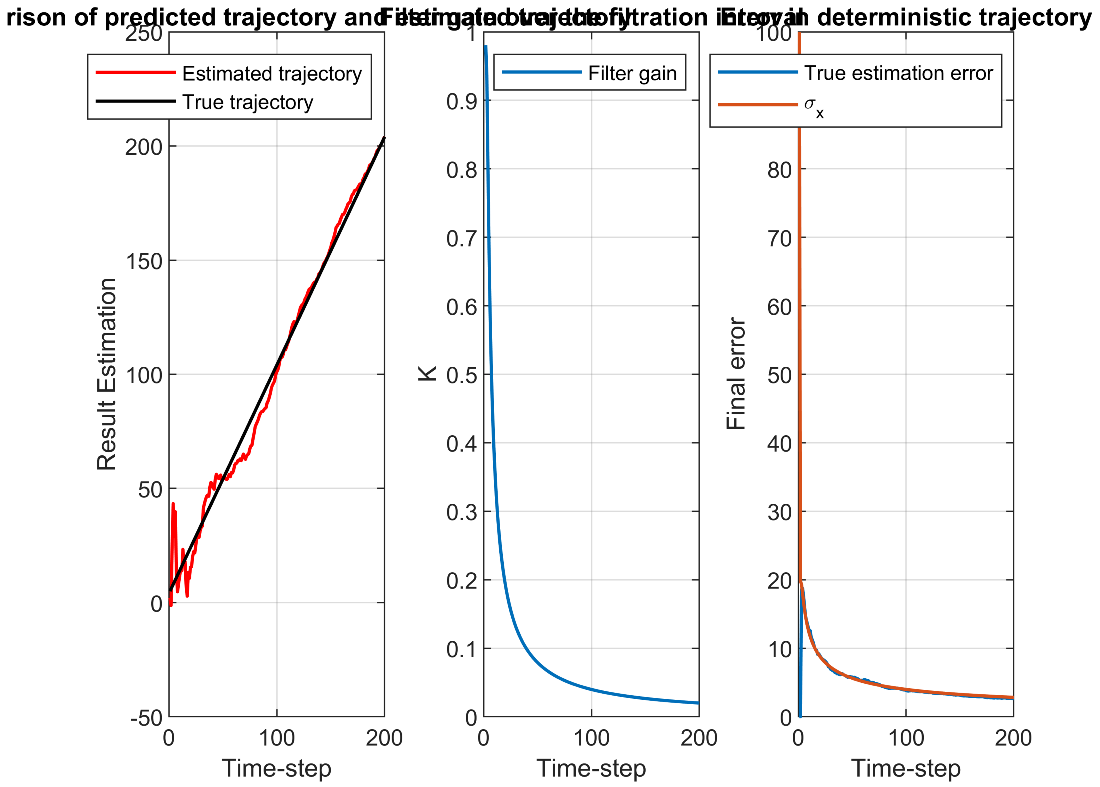


The true estimation error (mean-squared error), which is obtained over M=500 runs is plotted for both P_0 and P_0/100. The system stabilizes as the number of time steps increases. The maximum error is reached in the same time step in both cases.

The plot for the standard deviation of estimation error of the coordinate for P_0 and P_0/100 shows that there is a abrupt increase in the standard deviation for P_0/100 which then stabilizes with the standard deviation for P_0. This abrupt increase is due to the initial conditions choice.

Task 12: Run filter for deterministic trajectory

```
M = 500;
Final_Error_filt = zeros(1,n);
error_filt = Final_Error_estr;
for run = 1:M
    x = 5*ones(1,n);
    V = ones(1,n);
    var_a = 0;
    a = sqrt(var_a)*randn(1,n);

    for i = 2:n
        x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
        V(i) = V(i-1)+a(i-1)*T;
    end

    var_eta = 20^2;
    eta = sqrt(var_eta)*randn(1,n);
    z = eta+x;

    X(:,1) = [2 0].';
    P = [1e4 0; 0 1e4];
    sigma_x = sqrt(P(1,1)*ones(1,n));
    R = var_eta;
    Q = G*G.'*var_a;

    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi.'+Q;

        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x(i) = sqrt(P(1,1));
    end

    for i = 3:n
        error_filt(i) = (x(1,i)-X(1,i))^2;
        Final_Error_filt(i)= Final_Error_filt(i)+error_filt(i);
    end
end
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

figure
subplot(1,3,1)
plot(X(1,:),'r','linewidth',1.2)
hold on
plot(x,'k','linewidth',1.2)
title('Comparison of predicted trajectory and estimated trajectory')
legend('Estimated trajectory','True trajectory')
ylabel('Result Estimation')
xlabel('Time-step')
grid on

% 1
subplot(1,3,2)
plot(K(1,:),'linewidth',1.2)
grid on
title('Filter gain over the filtration interval')
legend('Filter gain')
ylabel('K')
xlabel('Time-step')
% 2
subplot(1,3,3)
plot(Final_Error_filt,'linewidth',1.2)
hold on
plot(sigma_x,'linewidth',1.2)
grid on
title('Error in deterministic trajectory')
legend('True estimation error','\sigma_x')
ylabel('Final error')
xlabel('Time-step')
```

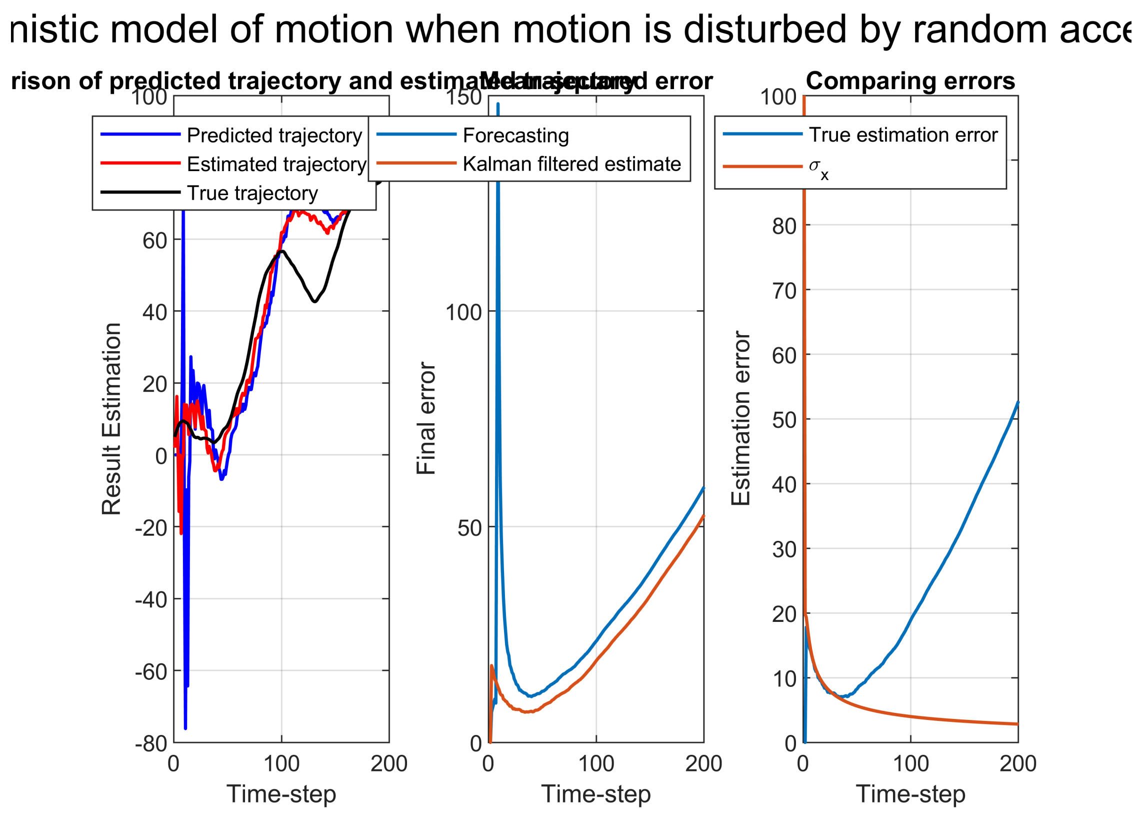


The filter gain, true estimation error and standard deviation of estimation error approch zero with the increasing number of time steps (if one enlarges the time step number, for instance, to 1000, one can see the graph almost touches the x-axis). The conditions of motion without any random disturbances, estimation error approaches to zero and filter switches off from measurements.

Task 13: what happens if you use deterministic model of motion, but in fact motion is disturbed by random acceleration

```
M = 500;
Final_Error_estr = zeros(1,n);
error_estr = Final_Error_estr;
Final_Error_filt = zeros(1,n);
error_filt = Final_Error_estr;
for run = 1:M
    x = 5*ones(1,n);
    V = ones(1,n);
    var_a = 0.2^2;
    a = sqrt(var_a)*randn(1,n);

    for i = 2:n
        x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
        V(i) = V(i-1)+a(i-1)*T;
    end

    var_eta = 20^2;
    eta = sqrt(var_eta)*randn(1,n);
    z = eta+x;

    X(:,1) = [2 0].';
    P = [1e4 0; 0 1e4];
    sigma_x = sqrt(P(1,1)*ones(1,n));
    R = var_eta;
    Q = 0;

    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi.'+Q;

        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x(i) = sqrt(P(1,1));
    end

    m = 7;
    for i = 1:n-m+1
        X_pred(:,i+m-1) = phi^(m-1)*X(:,i);
    end

    for i = 3:n
        error_estr(i) = (x(1,i)-X_pred(1,i))^2;
        Final_Error_estr(i)= Final_Error_estr(i)+error_estr(i);
    end
    for i = 3:n
        error_filt(i) = (x(1,i)-X(1,i))^2;
        Final_Error_filt(i)= Final_Error_filt(i)+error_filt(i);
    end
end

Final_Error_estr = sqrt(Final_Error_estr/(M-1));
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

figure
subplot(1,3,1)
sgtitle('Deterministic model of motion when motion is disturbed by random acceleration')
plot(X_pred(1,:),'b','linewidth',1.2)
hold on
plot(X(1,:),'r','linewidth',1.2)
plot(x,'k','linewidth',1.2)
title('Comparison of predicted trajectory and estimated trajectory')
legend('Predicted trajectory','Estimated trajectory','True trajectory')
ylabel('Result Estimation')
xlabel('Time-step')
grid on

%a
subplot(1,3,2)
plot(Final_Error_estr, 'linewidth',1.2)
hold on
plot(Final_Error_filt,'linewidth',1.2)
```

```
grid on
title(' Mean-squared error ')
legend('Forecasting','Kalman filtered estimate')
ylabel('Final error')
xlabel('Time-step')

%b Compare calculation errors of estimation P provided...
% Kalman filter algorithm with true estimation errors
subplot(1,3,3)
plot(Final_Error_filt,'linewidth',1.2)
hold on
plot(sigma_x,'linewidth',1.2)
grid on
title('Comparing errors')
legend('True estimation error','\sigma_x')
ylabel('Estimation error')
xlabel('Time-step')
```



Covariance matrix Q of state noise is zero which helps in determining the prediction error covariance matrix. It also takes into account the noise in the random acceleration. We observe some divergence in the prediction of the model away from the true trajectory. The final filter estimation and the extrapolation tend to diverge as the number of time steps increases.

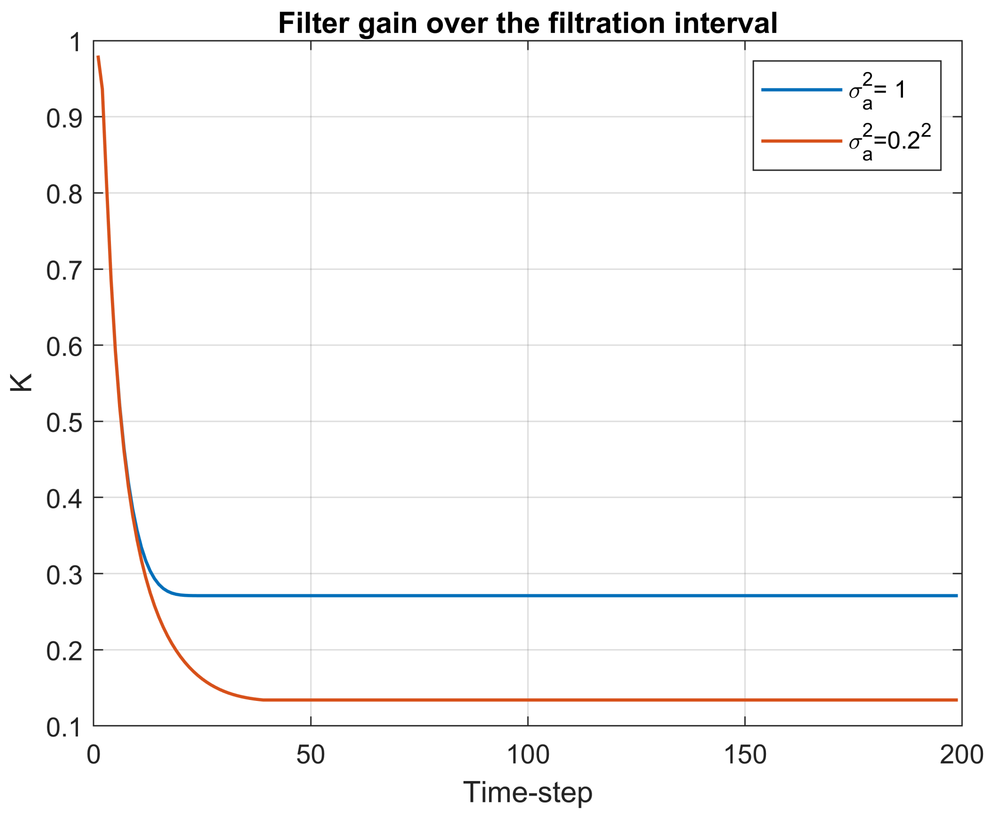
In case of motion disturbed by random acceleration and described by the deterministic model, the curve of final error has fluctuation around mean-squared error of filtered estimate of coordinate. In this case the gap between the lines starts to diverge after almost 40 iterations. This is a significant sign about how wrong the choise of the model is.

Task 14: Analyze how the relationship between state and measurement noise affect time

```
clc
clear
%close all

%  var_a = 1;
n = 200;
x = 5*ones(1,n);
V = ones(1,n);
T = 1;
var_a = 1;
a = sqrt(var_a)*randn(1,n);

for i = 2:n
    x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
    V(i) = V(i-1)+a(i-1)*T;
end

var_eta = 20^2;
eta = sqrt(var_eta)*randn(1,n);
z = eta+x; % measurements of coordinate x

% System at state space
phi = [1 T; 0 1]; % transition matrix that relates X(i) and X(i-1);
G = [T^2/2 T].'; % input matrix, that determines how random acceleration affects state vector;
H = [1 0];     % observation matrix

X(:,1) = [2 0].';
P = [1e4 0; 0 1e4];
sigma_x(1) = sqrt(P(1,1));
R = var_eta;
Q = G*G.'*var_a;

for i = 2:n
    X(:,i) = phi*X(:,i-1);
    P_pred = phi*P*phi.'+Q;

    if i > 25
        K(:,i) = K(:,25);
    else
        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
    end

    X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
    P = (eye(2)-K(:,i)*H)*P_pred;
    sigma_x(i) = sqrt(P(1,1));
end
K(:,1)= [];
```

```matlab
figure
plot(K(1,:),'linewidth',1.2)
grid on
hold on

% var_a = 0.2^2
n = 200;
x = 5*ones(1,n);
V = ones(1,n);
T = 1;
var_a = 0.2^2;
a = sqrt(var_a)*randn(1,n);

for i = 2:n
    x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
    V(i) = V(i-1)+a(i-1)*T;
end

var_eta = 20^2;
eta = sqrt(var_eta)*randn(1,n);
z = eta+x;        % measurements of coordinate x

% System at state space
phi = [1 T; 0 1]; % transition matrix that relates X(i) and X(i-1);
G = [T^2/2 T].'; % input matrix, that determines how random acceleration affects state vector;
H = [1 0];     % observation matrix

X(:,1) = [2 0].';
P = [1e4 0; 0 1e4];
sigma_x(1) = sqrt(P(1,1));
R = var_eta;
Q = G*G.'*var_a;

for i = 2:n
    X(:,i) = phi*X(:,i-1);
    P_pred = phi*P*phi.'+Q;

    if i > 40
        K(:,i) = K(:,40);
    else
        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
    end
    X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
    P = (eye(2)-K(:,i)*H)*P_pred;
    sigma_x(i) = sqrt(P(1,1));
end
K(:,1)= [];

plot(K(1,:),'linewidth',1.2)
grid on
title('Filter gain over the filtration interval')
ylabel('K')
xlabel('Time-step')
```

```
legend('\sigma^2_a= 1','\sigma^2_a=0.2^2')
```

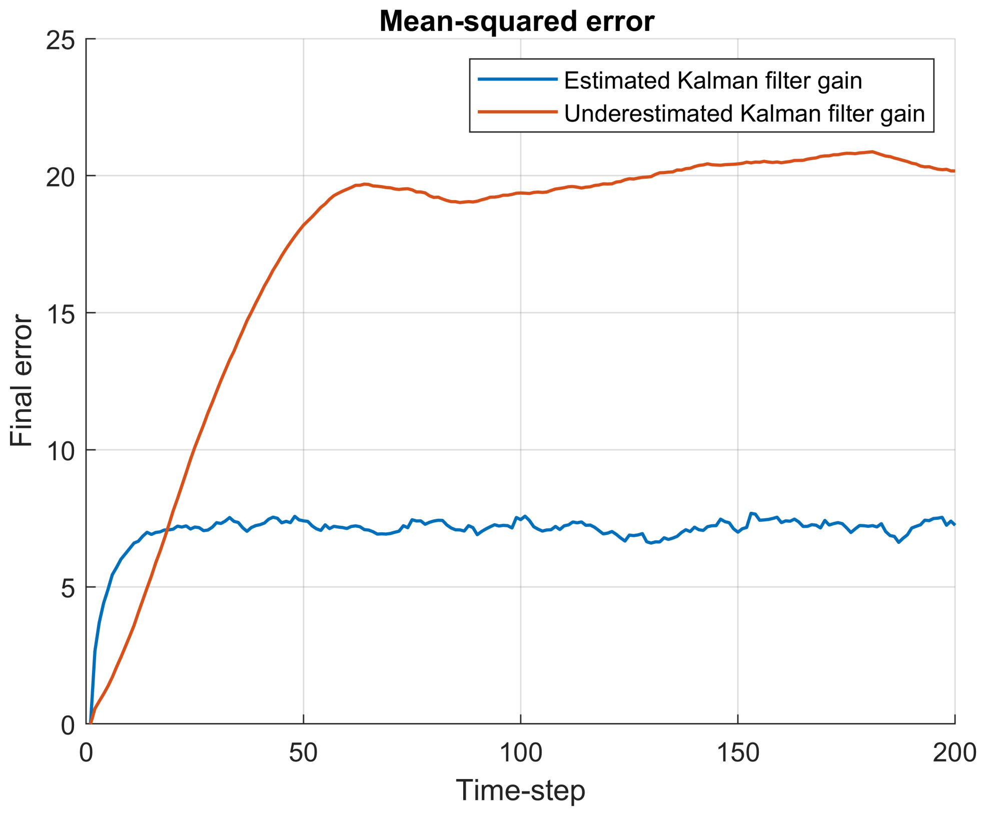


The lower the ratio of the variance to state noise to variance of measurement noise, the slower the filter gain becomes constant.

Task 15: Analyze sensitivity of filter

```
close all
% Trajectory generation
n = 200;
x = 5*ones(1,n);
V = ones(1,n);
T = 1;

var_a = 0.2^2;
a = sqrt(var_a)*randn(1,n);

for i = 2:n
    x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
    V(i) = V(i-1)+a(i-1)*T;
end

% Optimal filter gain (Kalman)
phi = [1 T; 0 1]; % transition matrix that relates X(i) and X(i-1);
G = [T^2/2 T].'; % input matrix, that determines how random acceleration affects state vector;
H = [1 0];     % observation matrix

X(:,1) = [100 5].';
```

```
P = [1e4 0; 0 1e4];
sigma_x = sqrt(P(1,1)*ones(1,n));
R = var_eta;
Q = G*G.'*var_a;

M = 500;
Final_Error_filt = zeros(1,n);
error_filt = Final_Error_filt;
for run = 1:M
    x = 100*ones(1,n);
    V = 5*ones(1,n);
    var_a = 0.2^2;
    a = sqrt(var_a)*randn(1,n);

    for i = 2:n
        x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
        V(i) = V(i-1)+a(i-1)*T;
    end

    var_eta = 20^2;
    eta = sqrt(var_eta)*randn(1,n);
    z = eta+x;

    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi.'+Q;

        K(:,i) = P_pred*H.'/(H*P_pred*H.'+R);
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x(i) = sqrt(P(1,1));
    end

    for i = 1:n
        error_filt(i) = (x(1,i)-X(1,i))^2;
        Final_Error_filt(i)= Final_Error_filt(i)+error_filt(i);
    end
end
Final_Error_filt = sqrt(Final_Error_filt/(M-1));


figure
hold on
plot(Final_Error_filt,'linewidth',1.2)

% Underestimated filter gain
K = K(:,40)/5.*ones(2,n); %underestimated kalman filter

Final_Error_filt = zeros(1,n);
error_filt = Final_Error_filt;
for run = 1:M
    x = 100*ones(1,n);
    V = 5*ones(1,n);
    var_a = 0.2^2;
    a = sqrt(var_a)*randn(1,n);

    for i = 2:n
        x(i) = x(i-1)+V(i-1)*T+a(i-1)*T^2/2;
        V(i) = V(i-1)+a(i-1)*T;
    end

    var_eta = 20^2;
    eta = sqrt(var_eta)*randn(1,n);
    z = eta+x;

    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi.'+Q;

        %K(:,i) = (P_pred*H.'/(H*P_pred*H.'+R))/5;%underestimated kalman filter
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P = (eye(2)-K(:,i)*H)*P_pred;
        sigma_x(i) = sqrt(P(1,1));
    end

    for i = 1:n
        error_filt(i) = (x(1,i)-X(1,i))^2;
        Final_Error_filt(i)= Final_Error_filt(i)+error_filt(i);
    end
end
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

plot(Final_Error_filt,'linewidth',1.2)
grid on
title('Mean-squared error ')
legend('Estimated Kalman filter gain','Underestimated Kalman filter gain')
ylabel('Final error')
xlabel('Time-step')
```



The estimation results for the optimal filter gain and underestimated filter gain are compared. The optimal filter gain becomes constant with fluctuations after a particular time step but the underestimated filter gain increases sharply and diverges with large differences. This is due to the initial conditions of the filter estimation.