Assignment 8 Development of optimal smoothing to increase the estimation accuracy

Group 6:

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


```
clc
clear
close all
```

Task 1: Reminder

Smoothing procedure is performed in backward in time and is applied to forward Kalman filter estimates. Smoothing takes into account both current and future measurements and therefore provides improved estimation compared to Kalman filter.

Task 2: Developing forward Kalman filter algorithm for a motion disturbed by normally distributed unbiased random acceleration

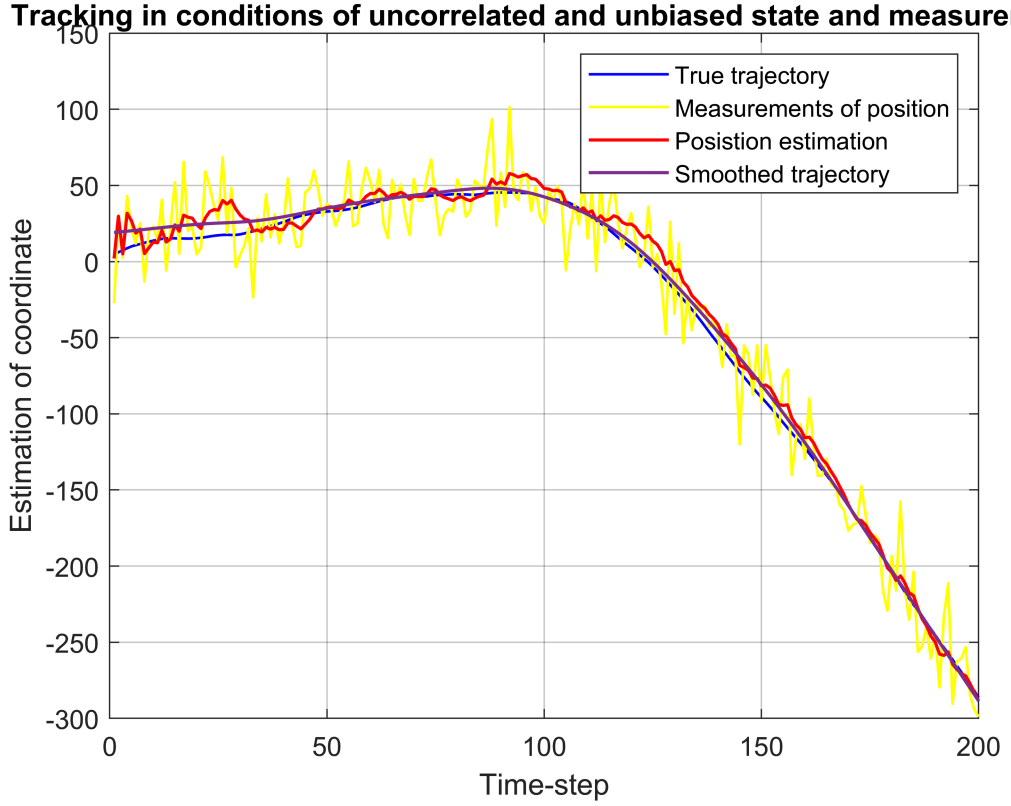
```
n = 200;
x = 5*ones(1,n);
V = ones(1,n);
T = 1;
var_a = 0.2^2;
a = sqrt(var_a)*randn(1,n);

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

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

```
P = 1e4*eye(2);

K = zeros(2,n);
j = 1;
for i = 2:n
    X(:,i) = phi*X(:,i-1);
    P_pred(:,j+2:j+3) = phi*P(:,j:j+1)*phi'+ Q;

    K(:,i) = P_pred(:,j+2:j+3)*H'/(H*P_pred(:,j+2:j+3)*H'+R);
    X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
    P(:,j+2:j+3) = (eye(2)-K(:,i)*H)*P_pred(:,j+2:j+3);
    j = j + 2;
end
K(:,1)= [NaN,NaN]';
```

Task 3: Develop backward smoothing algorithm

```
X_s = X;
P_s = P;
sigma_x = zeros(1,n-1);
sigma_V = zeros(1,n-1);
for i = n-1:-1:1
    j = -1 + 2*i;
    A = P(:,j:j+1)*phi'*inv(P_pred(:,j+2:j+3));
    X_s(:,i) = X(:,i) + A*(X_s(:,i+1)-phi*X(:,i));
    P_s(:,j:j+1) = P(:,j:j+1) + A*(P_s(:,j+2:j+3)-P_pred(:,j+2:j+3))*A';
    sigma_x(i) = sqrt(P_s(1,j));
    sigma_V(i) = sqrt(P_s(2,j+1));
end

figure
plot(x,'b','linewidth',1)
hold on
plot(z,'y','linewidth',1)
plot(X(1,:),'r','linewidth',1.2)
xlabel('Time-step')
title('Fig.1: Tracking in conditions of uncorrelated and unbiased state and measurement noise')
ylabel('Estimation of coordinate')
grid on
plot(X_s(1,:),'linewidth',1.2)
legend('True trajectory','Measurements of position',...
    'Posistion estimation','Smoothed trajectory')
```

.1: Tracking in conditions of uncorrelated and unbiased state and measurement



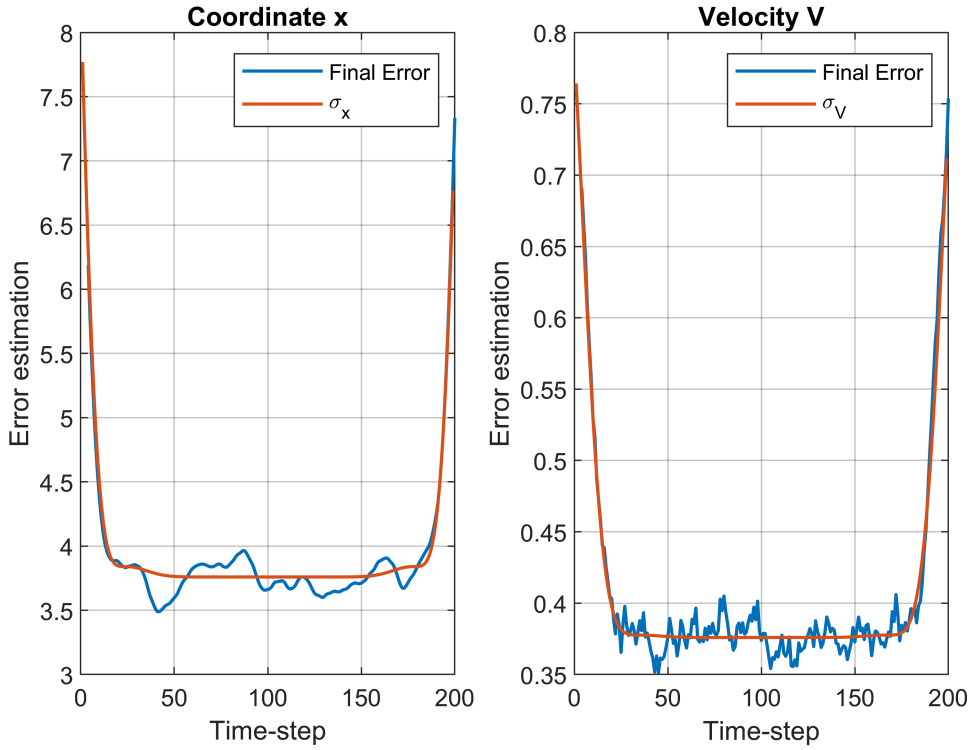
Task 4: Compare true estimation error with errors of smoothing

```
M = 500;

Final_Error_Vs = zeros(1,n);
error_Vs = zeros(1,n);
Final_Error_xs = zeros(1,n);
error_xs = zeros(1,n);
Final_Error_xf = zeros(1,n);
error_xf = zeros(1,n);
Final_Error_vf = zeros(1,n);
error_vf = 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*eye(2);
    R = var_eta;
    Q = G*G.'*var_a;

    j = 1;
    for i = 2:n
        X(:,i) = phi*X(:,i-1);
        P_pred(:,j+2:j+3) = phi*P(:,j:j+1)*phi.'+ Q;

        K(:,i) = P_pred(:,j+2:j+3)*H.'/(H*P_pred(:,j+2:j+3)*H.'+R);
        X(:,i) = X(:,i) + K(:,i)*(z(i)-H*X(:,i));
        P(:,j+2:j+3) = (eye(2)-K(:,i)*H)*P_pred(:,j+2:j+3);
        j = j + 2;
    end

    X_s = X;
    P_s = P;
    for i = n-1:-1:1
        j = -1 + 2*i;
        A = P(:,j:j+1)*phi'*inv(P_pred(:,j+2:j+3));
        X_s(:,i) = X(:,i) + A*(X_s(:,i+1)-phi*X(:,i));
        P_s(:,j:j+1) = P(:,j:j+1) + A*(P_s(:,j+2:j+3)-P_pred(:,j+2:j+3))*A';
    end

    for i = 3:n
        error_xs(i) = (x(1,i)-X_s(1,i))^2;
        Final_Error_xs(i)= Final_Error_xs(i)+error_xs(i);

        error_Vs(i) = (V(1,i)-X_s(2,i))^2;
        Final_Error_Vs(i)= Final_Error_Vs(i)+error_Vs(i);

        error_xf(i) = (x(1,i)-X(1,i))^2;
        Final_Error_xf(i)= Final_Error_xf(i)+error_xf(i);

        error_vf(i) = (V(1,i)-X(2,i))^2;
        Final_Error_vf(i)= Final_Error_vf(i)+error_vf(i);
    end
end
Final_Error_Vs = sqrt(Final_Error_Vs/(M-1));
Final_Error_xs = sqrt(Final_Error_xs/(M-1));
Final_Error_xf = sqrt(Final_Error_xf/(M-1));
Final_Error_vf = sqrt(Final_Error_vf/(M-1));
Final_Error_Vs(1:3)= [NaN,NaN,NaN];
Final_Error_xs(1:3)= [NaN,NaN,NaN];
Final_Error_xf(1:3)= [NaN,NaN,NaN];
Final_Error_vf(1:3)= [NaN,NaN,NaN];


figure
subplot(1,2,1)
sgtitle('Fig.2: Error of smoothed estimates')
plot(Final_Error_xs,'linewidth',1.2)
hold on
```

```
plot(sigma_x,'linewidth',1.2)
legend('Final Error','\sigma_x')
ylabel('Error estimation')
xlabel('Time-step')
title('Coordinate x')
grid on

subplot(1,2,2)
plot(Final_Error_Vs, 'linewidth',1.2)
hold on
plot(sigma_V,'linewidth',1.2)
grid on
title('Velocity V')
legend('Final Error','\sigma_V')
ylabel('Error estimation')
xlabel('Time-step')
```

Fig.2: Error of smoothed estimates

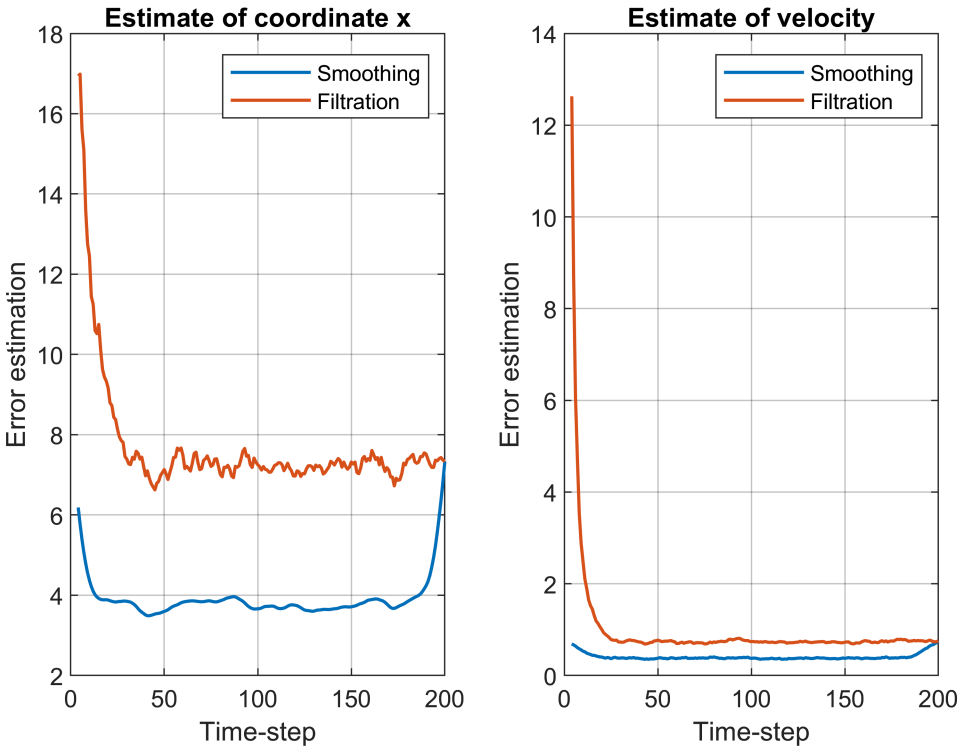


Task 5: Error comparison

```
figure
subplot(1,2,1)
sgtitle('Fig.3: Error comparison for:')
plot(Final_Error_xs,'linewidth',1.2)
hold on
plot(Final_Error_xf,'linewidth',1.2)
grid on
title('Estimate of coordinate x')
ylabel('Error estimation')
```

```
xlabel('Time-step')
legend('Smoothing','Filtration')

subplot(1,2,2)
plot(Final_Error_Vs,'linewidth',1.2)
hold on
plot(Final_Error_vf,'linewidth',1.2)
grid on
title('Estimate of velocity')
ylabel('Error estimation')
xlabel('Time-step')
legend('Smoothing','Filtration')
```

Fig.3: Error comparison for:



Conclusions:

- The trajectory and measurements of moving object, whose motion is disturbed by normally distributed unbiased random acceleration with given variance of 0.2^2, is plotted along with the forward Kalman filter estimates. The forward Kalman filter follows the true trajectory of the moving object with minimal fluctuations.
- To minimize the sharp fluctuations of forward Kalman filter estimates, backward smoothing algorithm is applied to the state vector to obtain more improved estimates. The resulting graph shows better and almost no fluctuations for the backward smoothed trajectory.
- On comparing the true estimation error and errors of smoothing (standard deviation) provided by smoothing algorithm, the true error estimation of coordinate x fluctuates around the standard deviation

of position ~3.76 and the true estimation errors for the velocity of the object also fluctuates, but this time more closely, around the standard deviation of velocity ~0.37.

- Comparing the errors of filtered estimates with the errors of smoothed estimation of position, it can be observed that there is a large and almost steady gap between the curves. As it could be expected, smoothed trajectory provides less errors of estimation than just filtration of measurements. The same trend is shown in the graph comparing error of estimates of velocity: error of filter estimates is costantly bigger, within few tenths, than error of smoothed estimates.
- The error of filtered estimates for position is approximatively 1.8 times greater than the standard deviation for position. Also for velocity, the error of filtered estimates is ~2 times the error of smoothed estimates, which is also greater than the standard deviation of velocity.
- Hence, it can be concluded that the backward smoothed trajectory improved the filter estimates, providing more accurated results.