Lab7: Development of forward-backward Kalman filter in conditions of correlated state noise

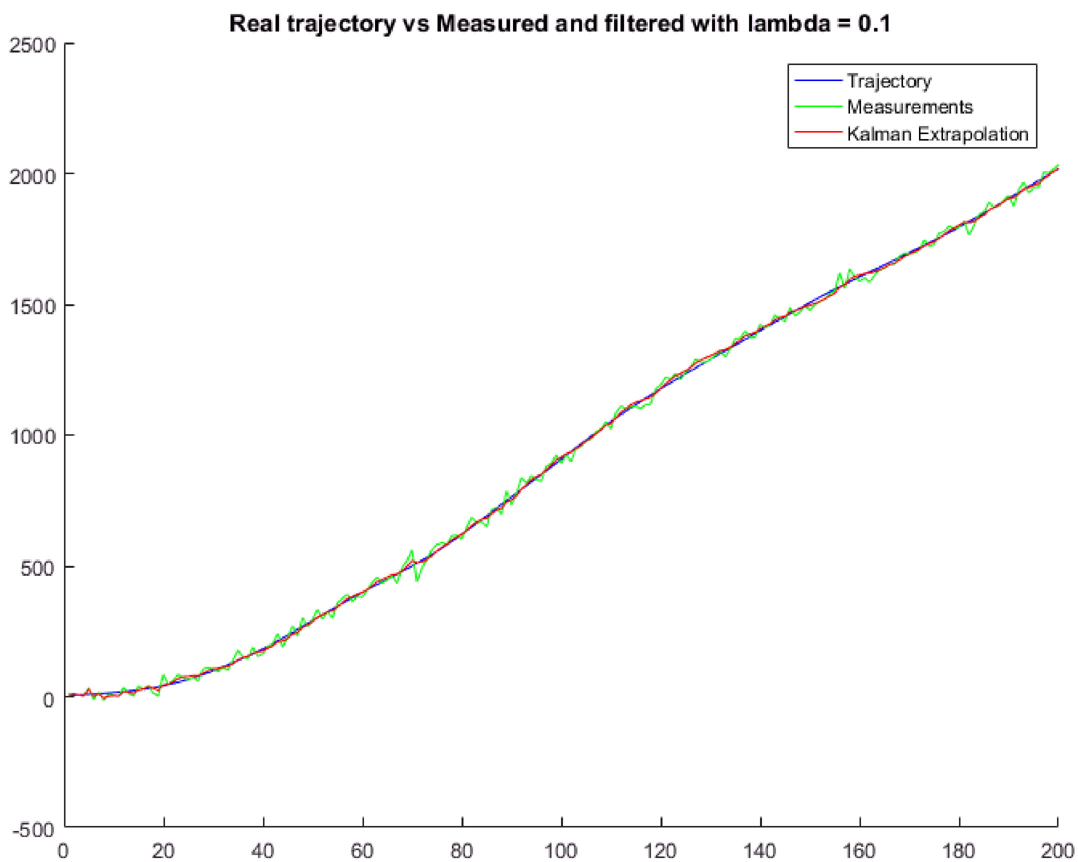
Team1: Dmitry Shadrin and Eugenii Israelit, Skoltech

```
clc; clear; close all;
addpath('functions/');

n = 200;
sigmaA = 0.2; sigmaN = 20; sigmaXi = 1;
x1 = 5; v1 = 1; t = 1;

G = [0; 0; 1];
H = [1, 0, 0];
P = [ 10000, 0, 0;     0, 10000, 0;    0, 0, 10000; ];

lambda = 0.1;
M = 500;
ErrSum = zeros(3,n);

F = [ 1, t, (t^2)/2;
  0, 1, t;
  0, 0, exp(-lambda*t) ];

sigmaZeta = (sigmaA^2)*(1-exp(-2*lambda*t));
```

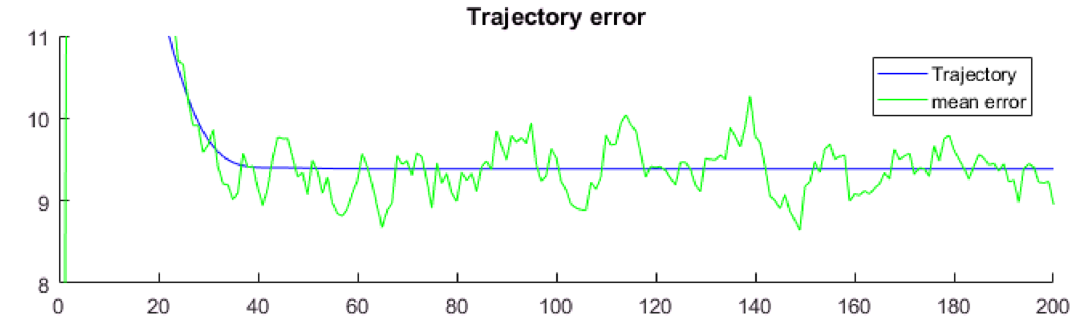
part I: Development of optimal Kalman filter in conditions of correlated state noise

```
for j=1:M
    A = gaussMarkov( n, sigmaA, sigmaXi, lambda, t);
    Noise = normrnd(0, sigmaN, 1, n);

    [ X, V, Z ] = calcTrajectory3( A, Noise, x1, v1, t);
    XVA = [ X; V; A ];

    [ Xk, SigmaX ] = calcKalman3(Z, sigmaZeta, sigmaN, x1, v1, F, G, H, P, 0);
    ErrCur = ( XVA - Xk ).^2;
    ErrSum = ErrSum + ErrCur;
end

FinalError = ( ErrSum./(M-1) ).^0.5;
```

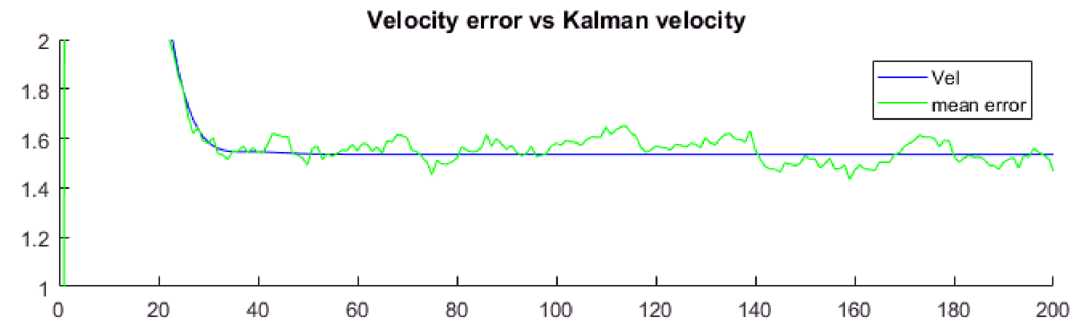
```
figure('position', [0, 0, 800, 600]) ;  hold on
    plot(X,'blue');
    plot(Z, 'green');
    plot(Xk(1,:), 'red');
    title( ['Real trajectory vs Measured and filtered with lambda = ', num2str(lambda) ] );
    legend('Trajectory', 'Measurements', 'Kalman Extrapolation');
```



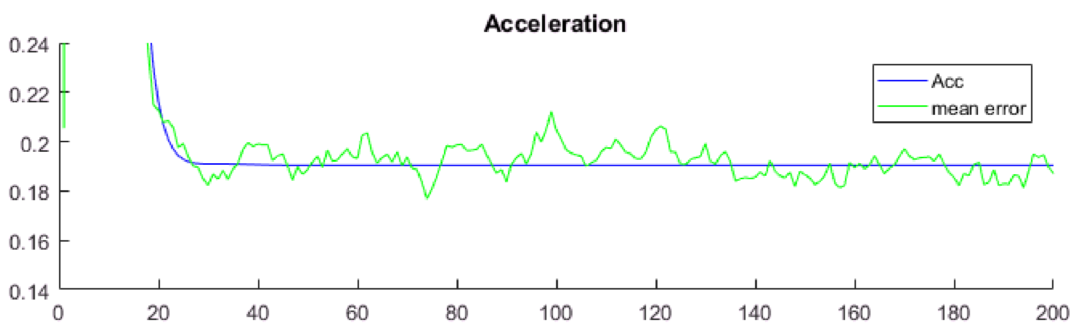

```
plotStartIndex=1;
figure('position', [0, 0, 800, 200]); hold on;
    plot(SigmaX(1, plotStartIndex:end),'blue');
    plot(FinalError(1, plotStartIndex:end), 'green');
    axis([0 n 8 11])
    title('Trajectory error');
    legend('Trajectory', 'mean error');
```




```
figure('position', [0, 0, 800, 200]); hold on;
    plot(SigmaX(2, plotStartIndex:end),'blue');
    plot(FinalError(2, plotStartIndex:end), 'green');
    axis([0 n 1 2])
    title('Velocity error vs Kalman velocity');
    legend('Vel', 'mean error');
```

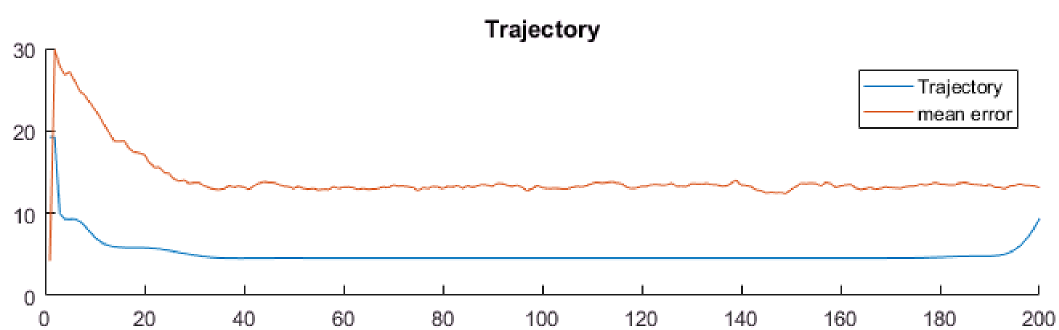



```
figure('position', [0, 0, 800, 200]); hold on;
    plot(SigmaX(3, plotStartIndex:end),'blue');
    plot(FinalError(3, plotStartIndex:end), 'green');
    axis([0 n 0.14 0.24])
    title('Acceleration');
    legend('Acc', 'mean error');
```

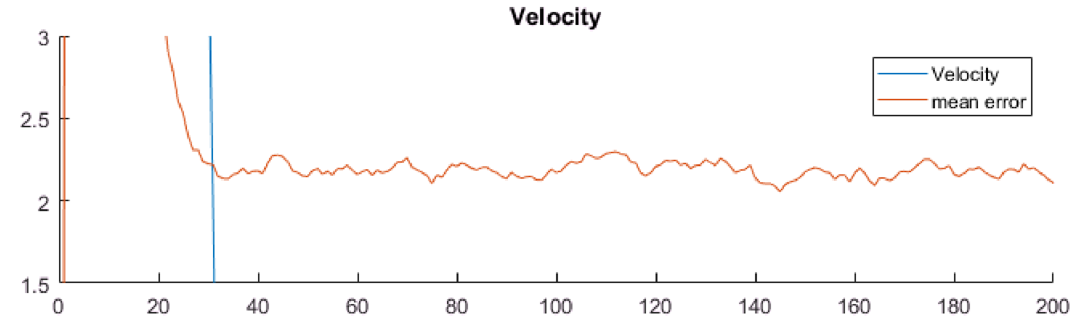


part II: Development of optimal smoothing to increase the estimation accuracy

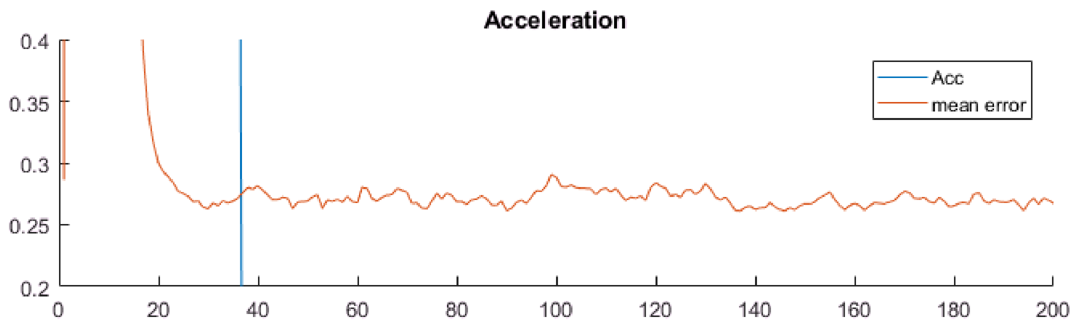
```
for j=1:M
    A = gaussMarkov( n, sigmaA, sigmaXi, lambda, t);
    Noise = normrnd(0, sigmaN, 1, n);

    [ X, V, Z ] = calcTrajectory3( A, Noise, x1, v1, t);
    XVA = [ X; V; A ];

    [ Xk, SigmaX, PiN] = calcKalmanSmooth(Z, sigmaZeta, sigmaN, x1, v1, F, G, H, P, 0);
    ErrCur = ( XVA - Xk ).^2;
    ErrSum = ErrSum + ErrCur;
end

FinalError = ( ErrSum./(M-1) ).^0.5; %true estimation error

for i=1:n
PP = PiN{i};
PP1(i) = sqrt(PP(1,1));
PP2(i) = sqrt(PP(2,2));
PP3(i) = sqrt(PP(3,3));
end

figure ('position', [0, 0, 800, 200]); hold on
plot(PP1)
plot(FinalError(1,:))
title('Trajectory');
legend('Trajectory', 'mean error');
```




```
figure('position', [0, 0, 800, 200]); hold on
plot(PP2)
plot(FinalError(2,:))
axis([0 n 1.5 3])
title('Velocity');
legend('Velocity', 'mean error');
```




```
figure('position', [0, 0, 800, 200]); hold on
plot(PP3)
plot(FinalError(3,:))
axis([0 n 0.2 0.4])
title('Acceleration');
legend('Acc', 'mean error');
```



```
function [ Xk, SigmaX ] = calcKalman3(Z, sigmaA, sigmaN, x1, v1, F, G,
 H, P, bias )

    n = length(Z);

    Xk = zeros(3, n);
    Xk(:, 1) = [2; 0; 0];

    Q = sigmaA * (G*G');

    SigmaX = zeros(3,n);
    SigmaX(1,1) = sqrt(P(1,1));
    SigmaX(2,1) = sqrt(P(2,2));
    SigmaX(3,1) = sqrt(P(3,3));

    for i=2:n
        P=F*P*F'+Q;
        K=P*H'/(H*P*H'+ sigmaN^2);
        Xk(:,i) = F*Xk(:, i-1) + G*bias;
        Xk(:,i) = Xk(:,i)+K*(Z(i)-H*Xk(:,i));

        P = (eye(3)-K*H)*P;
        SigmaX(1,i) = P(1,1)^(1/2) ;
        SigmaX(2,i) = P(2,2)^(1/2) ;
        SigmaX(3,i) = P(3,3)^(1/2) ;
    end

end
```

```
Not enough input arguments.

Error in calcKalman3 (line 3)
    n = length(Z);
```

Published with MATLAB® R2016a

```matlab
function [ Xk, SigmaX, PiN ] = calcKalmanSmooth(Z, sigmaA, sigmaN, x1,
 v1, F, G, H, P, bias )

    n=length(Z);

    Xk = zeros(3, n);
    Xk(:, 1) = [2; 0; 0];

    Q=sigmaA * (G*G');
    Ak=cell(1,200);
    PiN=cell(1,200);
    Ppredict=cell(1,200);
    Pfiltrate=cell(1,200);
        Pfiltrate(1)={P};
        Ppredict(1)={P};
    SigmaX = zeros(3,n);
    SigmaX(1,1) = sqrt(P(1,1));
    SigmaX(2,1) = sqrt(P(2,2));
    SigmaX(3,1) = sqrt(P(3,3));

    for i=2:n
        P=F*P*F'+Q;
        Ppredict(i)={P};
        K=P*H'/(H*P*H'+ sigmaN^2);
        Xk(:,i) = F*Xk(:, i-1) + G*bias;
        Xk(:,i) = Xk(:,i)+K*(Z(i)-H*Xk(:,i));

        P = (eye(3)-K*H)*P;
        Pfiltrate(i)={P};

        SigmaX(1,i) = sqrt(P(1,1));
        SigmaX(2,i) = sqrt(P(2,2));
        SigmaX(3,i) = sqrt(P(3,3));
    end

    Xks(:,n)=Xk(:,n);
    PiN{n}=Pfiltrate{n};
   for i=n-1:-1:1
      P1=Ppredict{i};
      P2=Pfiltrate{i};
          Ak(i)={P2*F'*(inv(P1))};

           PiN(i)={Pfiltrate{(i)}+Ak{(i)}*(PiN{(i+1)}-
Ppredict{(i)})*(Ak{(i)}')};
           Xks(:,i)=Xk(:,i)+Ak{(i)}*(Xks(:,i+1)-F*Xk(:,i));
    end
```

Not enough input arguments.

Error in calcKalmanSmooth (line 3)
    n=length(Z);