CS491/691: Introduction to Aerial Robotics

Topic: State Estimation – Coding Examples

Dr. Kostas Alexis (CSE)

Kalman Filter Design in MATLAB

- Consider the system:

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{B}\mathbf{u}(n)$$
$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) + \mathbf{D}\mathbf{u}(n)$$

- Where:

```
A = [1.1269   -0.4940    0.1129,
     1.0000         0         0,
          0    1.0000         0];

B = [-0.3832
      0.5919
      0.5191];

C = [1 0 0];

D = 0;
```

Kalman Filter Design in MATLAB

- Design of a Steady-State Kalman Filter: derive the optimal filter gain M based on the process noise covariance Q and the sensor noise coviariance R.

- Step 1: Plan definition

```
Plant = ss(A,[B B],C,0,-1,'inputname',{'u' 'w'},'outputname','y');
```

- Step 2: Covariance information

```
Q = 2.3; % A number greater than zero
```

```
R = 1; % A number greater than zero
```

Kalman Filter Design in MATLAB

- Step 3: Design the steady-state Kalman Filter

Time-update $$\mathbf{x}(n+1|n) = \mathbf{A}\mathbf{x}(n|n-1) + \mathbf{B}\mathbf{u}(n)$$

Measurement Update $$\mathbf{x}(n|n) = \mathbf{x}(n|n-1) + \mathbf{M}(\mathbf{y_v}(n) - \mathbf{C}\mathbf{x}(n|n-1))$$

  - Ask MATLAB to compute the Kalman gain for you

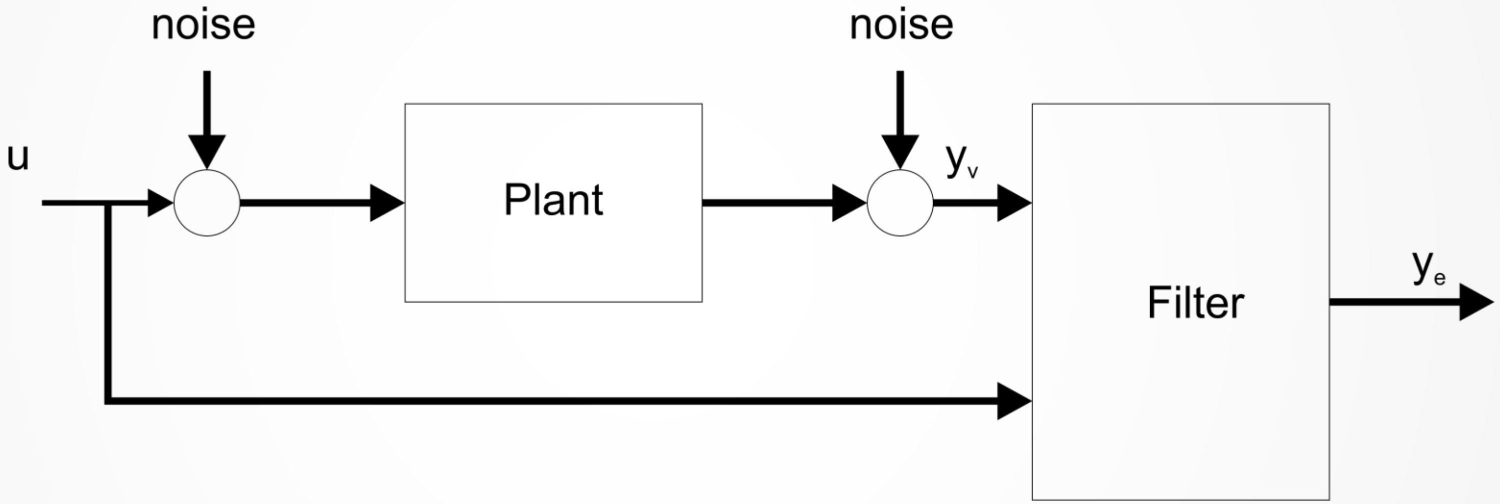
```
[kalmf,L,~,M,Z] = kalman(Plant,Q,R);
```

```
kalmf = kalmf(1,:);

M,   % innovation gain
```

  - M = $[0.5345, 0.0101, -0.4776]^T$

Kalman Filter Design in MATLAB

- Filter – System Block diagram:



Kalman Filter Design in MATLAB

- Step 4: Simulate the system connected with the filter

```
% First, build a complete plant model with u,w,v as inputs and
% y and yv as outputs:
a = A;
b = [B B 0*B];
c = [C;C];
d = [0 0 0;0 0 1];
P = ss(a,b,c,d,-1,'inputname',{'u' 'w' 'v'},'outputname',{'y' 'yv'});
```

```
sys = parallel(P,kalmf,1,1,[],[]);
```

```
SimModel = feedback(sys,1,4,2,1);
SimModel = SimModel([1 3],[1 2 3]);     % Delete yv form I/O
```

Kalman Filter Design in MATLAB

- Step 4: Create the block diagram in MATLAB

```
% First, build a complete plant model with u,w,v as inputs and
% y and yv as outputs:
a = A;
b = [B B 0*B];
c = [C;C];
d = [0 0 0;0 0 1];
P = ss(a,b,c,d,-1,'inputname',{'u' 'w' 'v'},'outputname',{'y' 'yv'});
```

```
sys = parallel(P,kalmf,1,1,[],[]);
```

```
SimModel = feedback(sys,1,4,2,1);
SimModel = SimModel([1 3],[1 2 3]);     % Delete yv form I/O
```

Kalman Filter Design in MATLAB

- Step 4: Conduct simulation
  - Insert time data and input data

```
t = (0:100)';
u = sin(t/5);
```

  - Insert process noise data

```
rng(10,'twister');
w = sqrt(Q)*randn(length(t),1);
v = sqrt(R)*randn(length(t),1);
```

  - Forward simulate

```
out = lsim(SimModel,[w,v,u]);

y = out(:,1);   % true response
ye = out(:,2);  % filtered response
yv = y + v;     % measured response
```

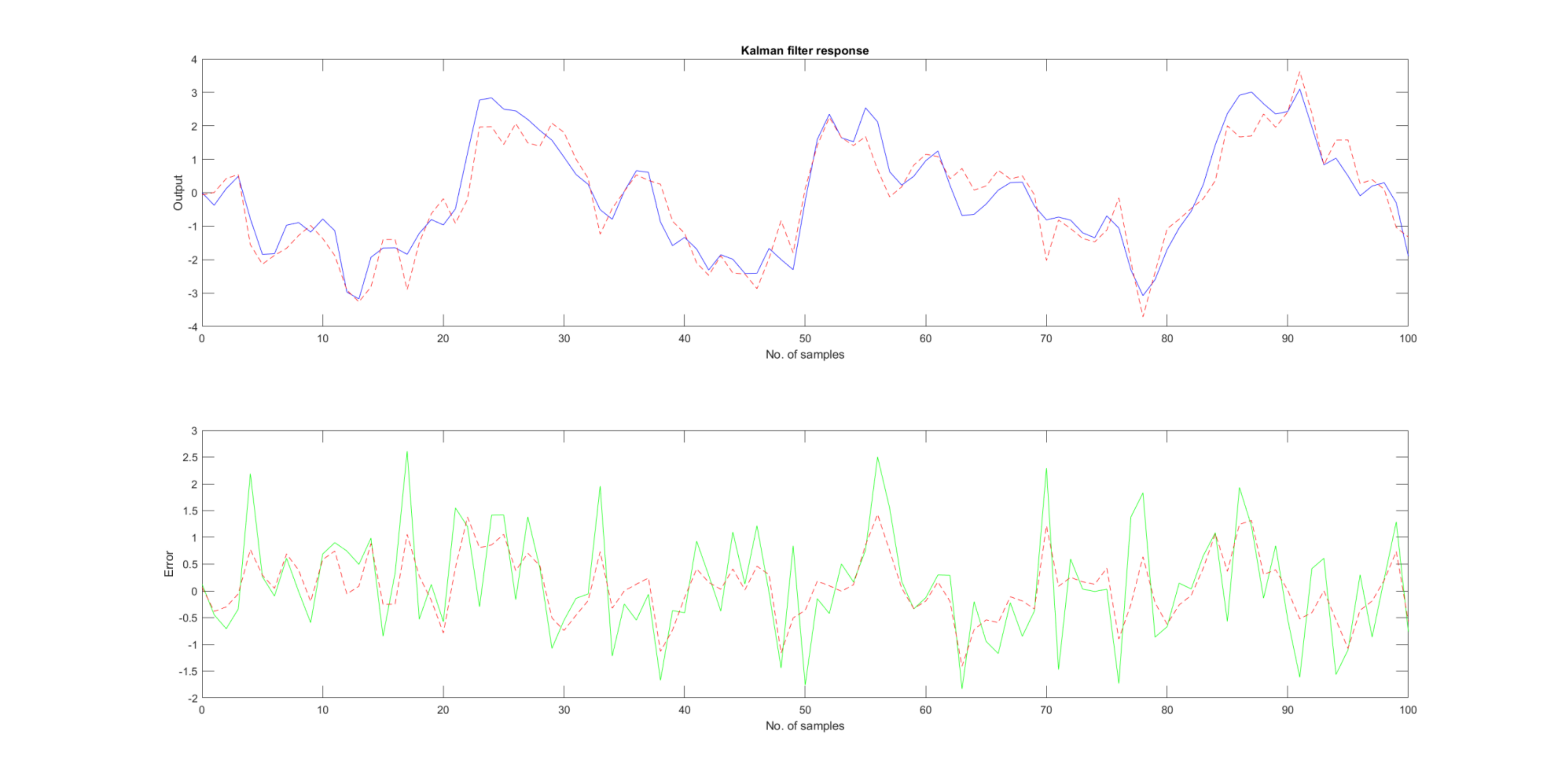
Kalman Filter Design in MATLAB

- Step 5: Visualize and compare the results

```
clf
subplot(211), plot(t,y,'b',t,ye,'r--'),
xlabel('No. of samples'), ylabel('Output')
title('Kalman filter response')
subplot(212), plot(t,y-yv,'g',t,y-ye,'r--'),
xlabel('No. of samples'), ylabel('Error')
```

Kalman Filter Design in MATLAB

- Step 5: Visualize and compare the results



Kalman Filter Design in MATLAB

- Step 5: Visualize and compare the results

```
MeasErr = y-yv;
MeasErrCov = sum(MeasErr.*MeasErr)/length(MeasErr);
EstErr = y-ye;
EstErrCov = sum(EstErr.*EstErr)/length(EstErr);
```

  - MassErrCov = 0.9871
  - EstErrCov = 0.3479

Kalman Filter Design in MATLAB

- Design of a Time-Varying Kalman Filter. A time-varying Kalman filter can perform well even when the noise covariance is not stationary. The time-varying KF is governed by:

Time update

$$\mathbf{x}(n+1|n) = \mathbf{A}\mathbf{x}(n|n) + \mathbf{B}\mathbf{u}(n)$$
$$\mathbf{P}(n+1|n) = \mathbf{A}\mathbf{P}(n|n)\mathbf{A}^T + \mathbf{B}\mathbf{Q}\mathbf{B}^T$$

Measurement update

$$\mathbf{x}(n|n) = \mathbf{x}(n|n-1) + \mathbf{M}(n)(\mathbf{y}_v(n) - \mathbf{C}\mathbf{x}(n|n-1))$$
$$\mathbf{M}(n) = \mathbf{P}(n|n-1)\mathbf{C}^T(\mathbf{C}\mathbf{P}(n|n-1)\mathbf{C}^T + \mathbf{R})$$
$$\mathbf{P}(n|n) = (\mathbf{I} - \mathbf{M}(n)\mathbf{C})\mathbf{P}(n|n-1)$$

Kalman Filter Design in MATLAB

- Step +1: Generate the noisy plant response

```
sys = ss(A,B,C,D,-1);
y = lsim(sys,u+w);   % w = process noise
yv = y + v;          % v = meas. noise
```

Kalman Filter Design in MATLAB

- Step +2: Implement Filter Recursions in a FOR loop

```
P=B*Q*B';          % Initial error covariance
x=zeros(3,1);      % Initial condition on the state
ye = zeros(length(t),1);
ycov = zeros(length(t),1);
errcov = zeros(length(t),1);

for i=1:length(t)
  % Measurement update
  Mn = P*C'/(C*P*C'+R);
  x = x + Mn*(yv(i)-C*x);  % x[n|n]
  P = (eye(3)-Mn*C)*P;     % P[n|n]

  ye(i) = C*x;
  errcov(i) = C*P*C';

  % Time update
  x = A*x + B*u(i);        % x[n+1|n]
  P = A*P*A' + B*Q*B';     % P[n+1|n]
end
```

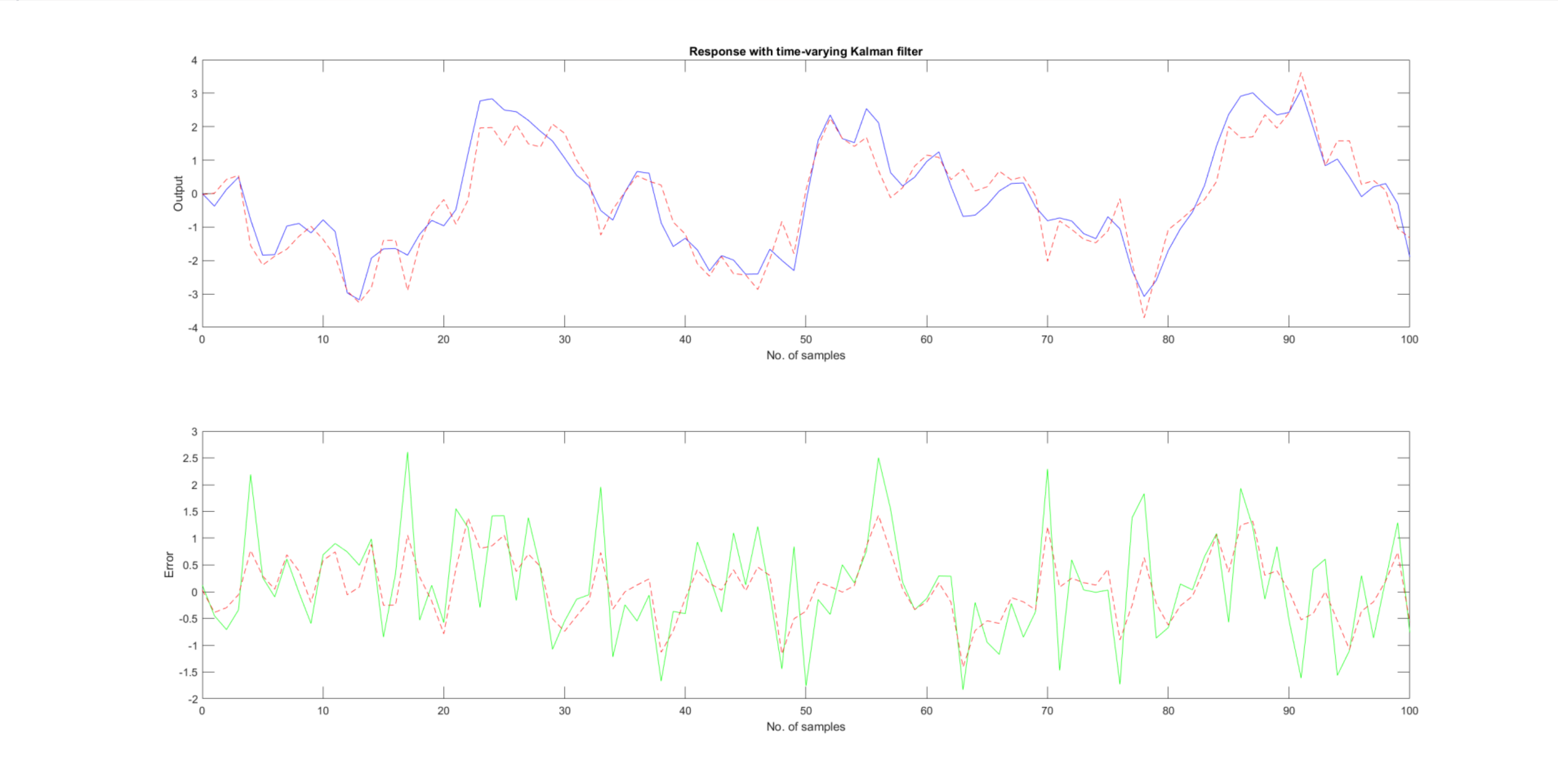
Kalman Filter Design in MATLAB

- Step +3: Compare the true response with the filtered response

```
subplot(211), plot(t,y,'b',t,ye,'r--'),
xlabel('No. of samples'), ylabel('Output')
title('Response with time-varying Kalman filter')
subplot(212), plot(t,y-yv,'g',t,y-ye,'r--'),
xlabel('No. of samples'), ylabel('Error')
```

Kalman Filter Design in MATLAB

- Step +3: Compare the true response with the filtered response



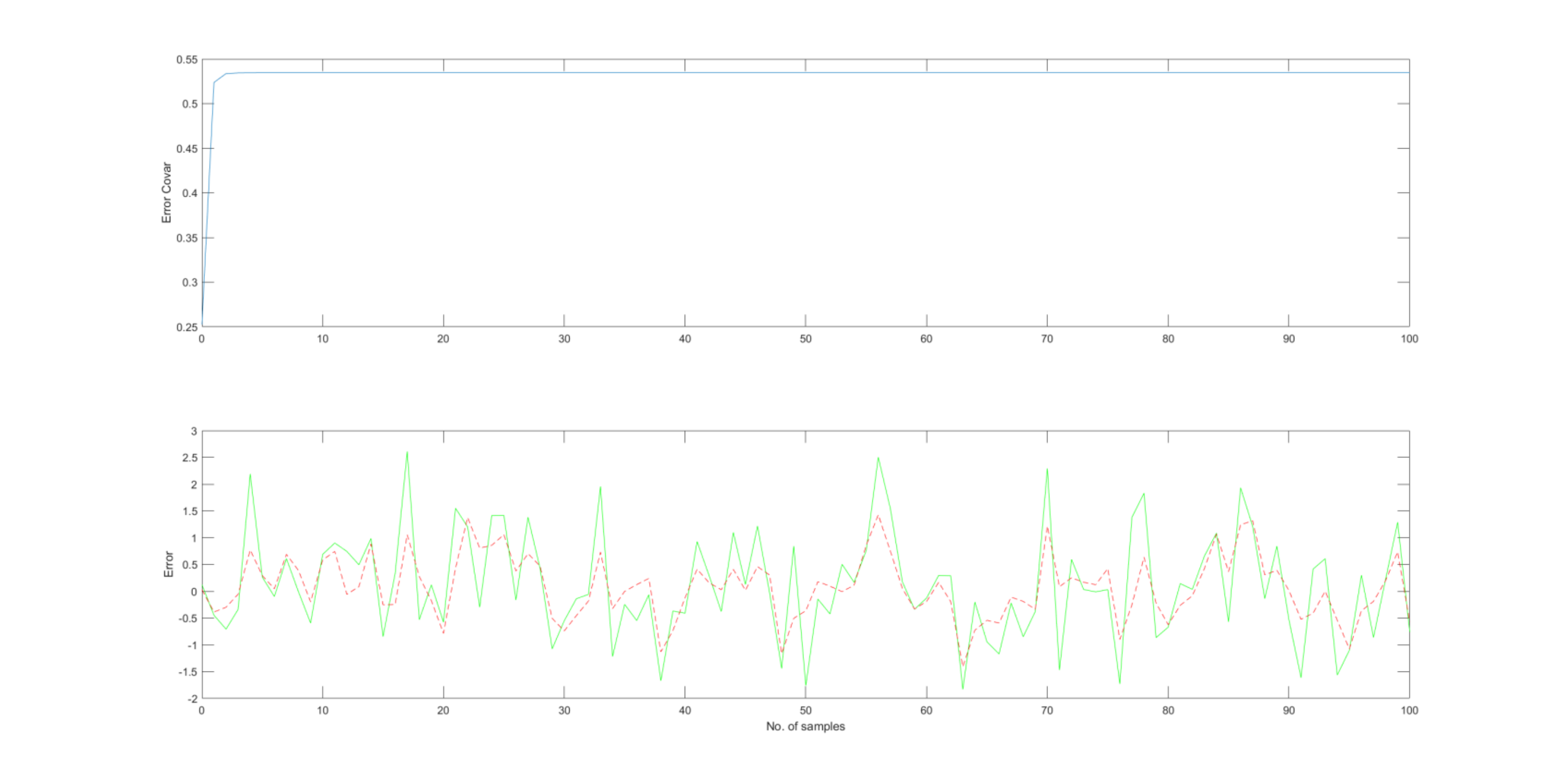
Kalman Filter Design in MATLAB

- Step +4: The time varying filter also estimates the output covariance during the estimation. Plot the output covariance to see if the filter has reached steady state (as we would expect with stationary input noise)

```
subplot(211)
plot(t,errcov), ylabel('Error Covar'),
```

Kalman Filter Design in MATLAB

- Step +4: The time varying filter also estimates the output covariance during the estimation. Plot the output covariance to see if the filter has reached steady state (as we would expect with stationary input noise)



Kalman Filter Design in MATLAB

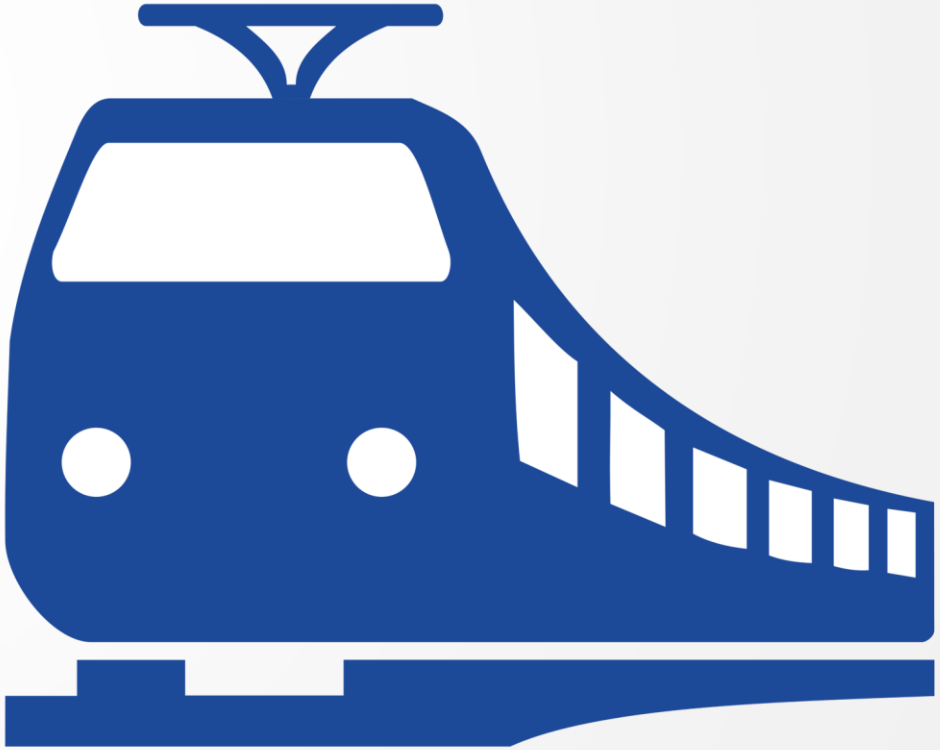
- Step +5: Compare covariance errors

```
MeasErr = y-yv;
MeasErrCov = sum(MeasErr.*MeasErr)/length(MeasErr);
EstErr = y-ye;
EstErrCov = sum(EstErr.*EstErr)/length(EstErr);
```

  - MeasErrCov = 0.9871
  - EstErrCov = 0.3479

Track a Train using the Kalman Filter

- **Problem statement:** Predict the position and velocity of a moving train 2 seconds ahead, % having noisy measurements of its positions along the previous 10 seconds (10 samples a second).



Based on: Alex Blekhman, An Intuitive Introduction to Kalman Filter

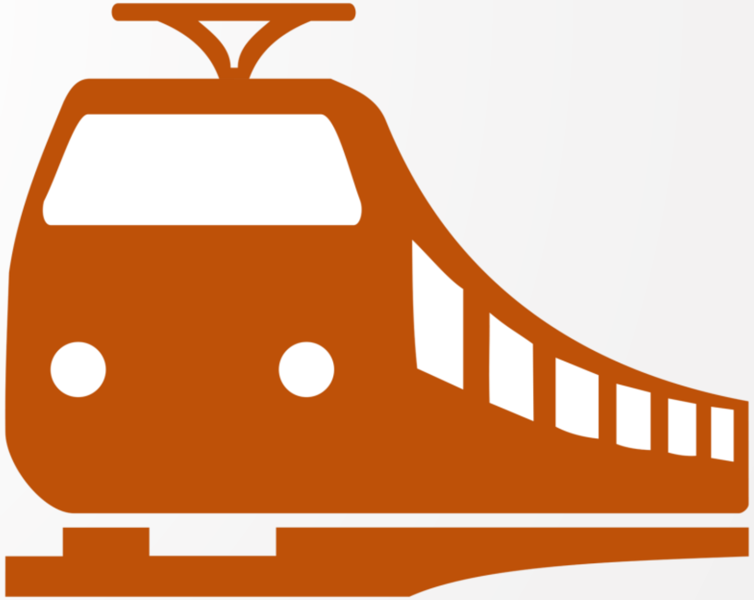
Track a Train using the Kalman Filter

- **Ground Truth:** The train is initially located at the point x = 0 and moves along the X axis with constant velocity V = 10m/sec, so the motion equation of the train is X = X0 + V*t. Easy to see that the position of the train after 12 seconds will be x = 120m, and this is what we will try to find.

Track a Train using the Kalman Filter

- **Approach:** We measure (sample) the position of the train every dt = 0.1 seconds. But, because of imperfect apparature, weather etc., our measurements are noisy, so the instantaneous velocity, derived from 2 consecutive position measurements (remember, we measure only position) is innacurate. We will use Kalman filter as we need an accurate and smooth estimate for the velocity in order to predict train's position in the future.

We assume that the measurement noise is normally distributed, with mean 0 and standard deviation SIGMA



Track a Train using the Kalman Filter

- **Ground truth**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ground truth %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Set true trajectory
Nsamples=100;
dt = .1;
t=0:dt:dt*Nsamples;
Vtrue = 10;

% Xtrue is a vector of true positions of the train
Xinitial = 0;
Xtrue = Xinitial + Vtrue * t;
```

Track a Train using the Kalman Filter

- **Motion Equations**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Motion equations %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Previous state (initial guess): Our guess is that the train starts at 0 with velocity
% that equals to 50% of the real velocity
Xk_prev = [0;
    .5*Vtrue];

% Current state estimate
Xk=[];

% Motion equation: Xk = Phi*Xk_prev + Noise, that is Xk(n) = Xk(n-1) + Vk(n-1) * dt
% Of course, V is not measured, but it is estimated
% Phi represents the dynamics of the system: it is the motion equation
Phi = [1 dt;
       0  1];
```

Track a Train using the Kalman Filter

Motion Equations

```
% Motion equation: Xk = Phi*Xk_prev + Noise, that is Xk(n) = Xk(n-1) + Vk(n-1) * dt
% Of course, V is not measured, but it is estimated
% Phi represents the dynamics of the system: it is the motion equation
Phi = [1 dt;
       0  1];

% The error matrix (or the confidence matrix): P states whether we should
% give more weight to the new measurement or to the model estimate
sigma_model = 1;
% P = sigma^2*G*G';
P = [sigma_model^2             0;
                 0 sigma_model^2];

% Q is the process noise covariance. It represents the amount of
% uncertainty in the model. In our case, we arbitrarily assume that the model is perfect (no
% acceleration allowed for the train, or in other words - any acceleration is considered to be a noise)
Q = [0 0;
     0 0];

% M is the measurement matrix.
% We measure X, so M(1) = 1
% We do not measure V, so M(2)= 0
M = [1 0];

% R is the measurement noise covariance. Generally R and sigma_meas can
% vary between samples.
sigma_meas = 1; % 1 m/sec
R = sigma_meas^2;
```

Track a Train using the Kalman Filter

- **Kalman Iterations**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Kalman iteration %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Buffers for later display
Xk_buffer = zeros(2,Nsamples+1);
Xk_buffer(:,1) = Xk_prev;
Z_buffer = zeros(1,Nsamples+1);

for k=1:Nsamples

    % Z is the measurement vector. In our
    % case, Z = TrueData + RandomGaussianNoise
    Z = Xtrue(k+1)+sigma_meas*randn;
    Z_buffer(k+1) = Z;

    % Kalman iteration
    P1 = Phi*P*Phi' + Q;
    S = M*P1*M' + R;

    % K is Kalman gain. If K is large, more weight goes to the measurement.
    % If K is low, more weight goes to the model prediction.
    K = P1*M'*inv(S);
    P = P1 - K*M*P1;

    Xk = Phi*Xk_prev + K*(Z-M*Phi*Xk_prev);
    Xk_buffer(:,k+1) = Xk;

    % For the next iteration
    Xk_prev = Xk;
end;
```

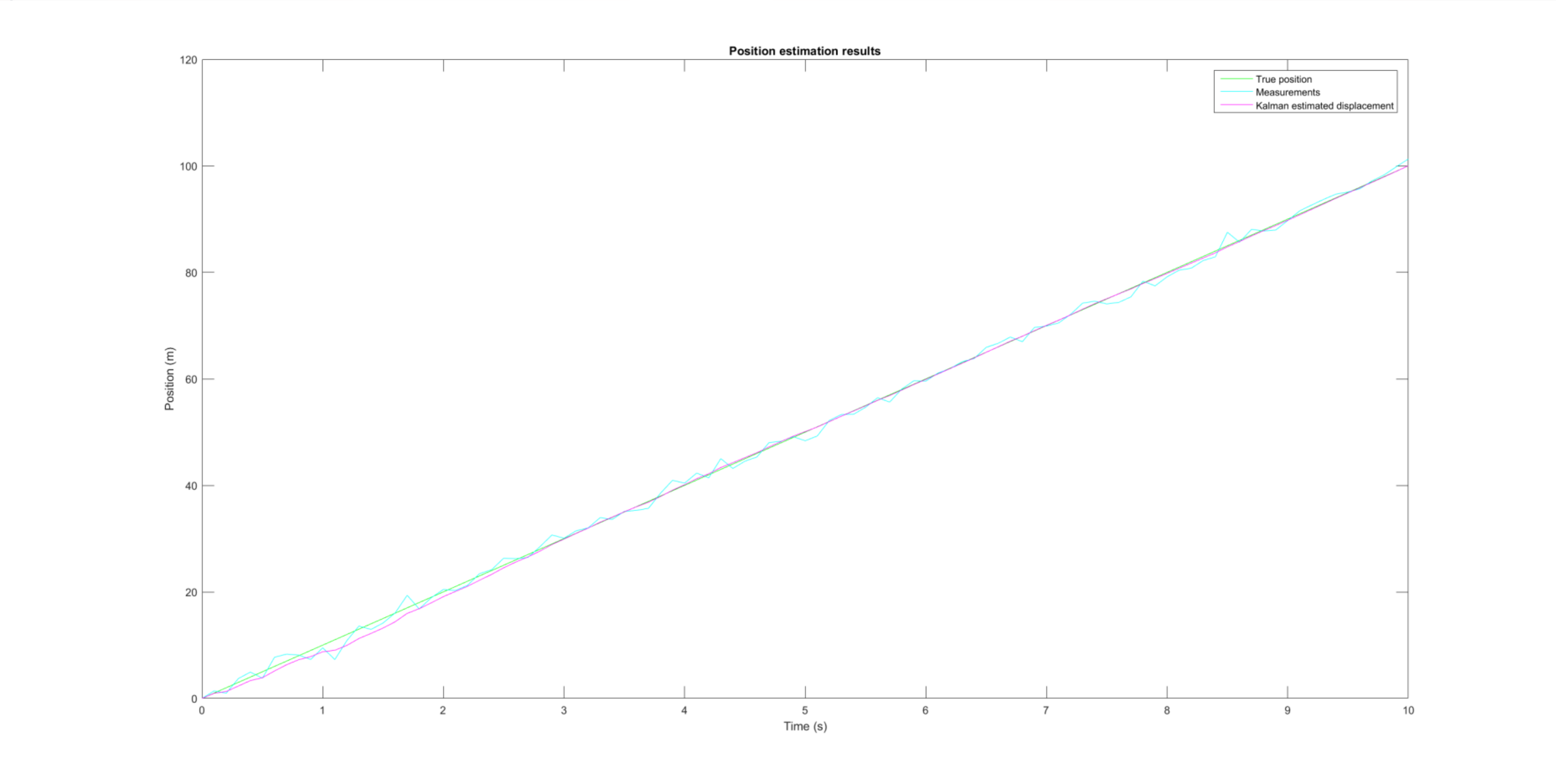
Track a Train using the Kalman Filter

- **Position Analysis**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Position analysis %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure;
plot(t,Xtrue,'g');
hold on;
plot(t,Z_buffer,'c');
plot(t,Xk_buffer(1,:),'m');
title('Position estimation results');
xlabel('Time (s)');
ylabel('Position (m)');
legend('True position','Measurements','Kalman estimated displacement');
```

Track a Train using the Kalman Filter

- **Position Analysis**



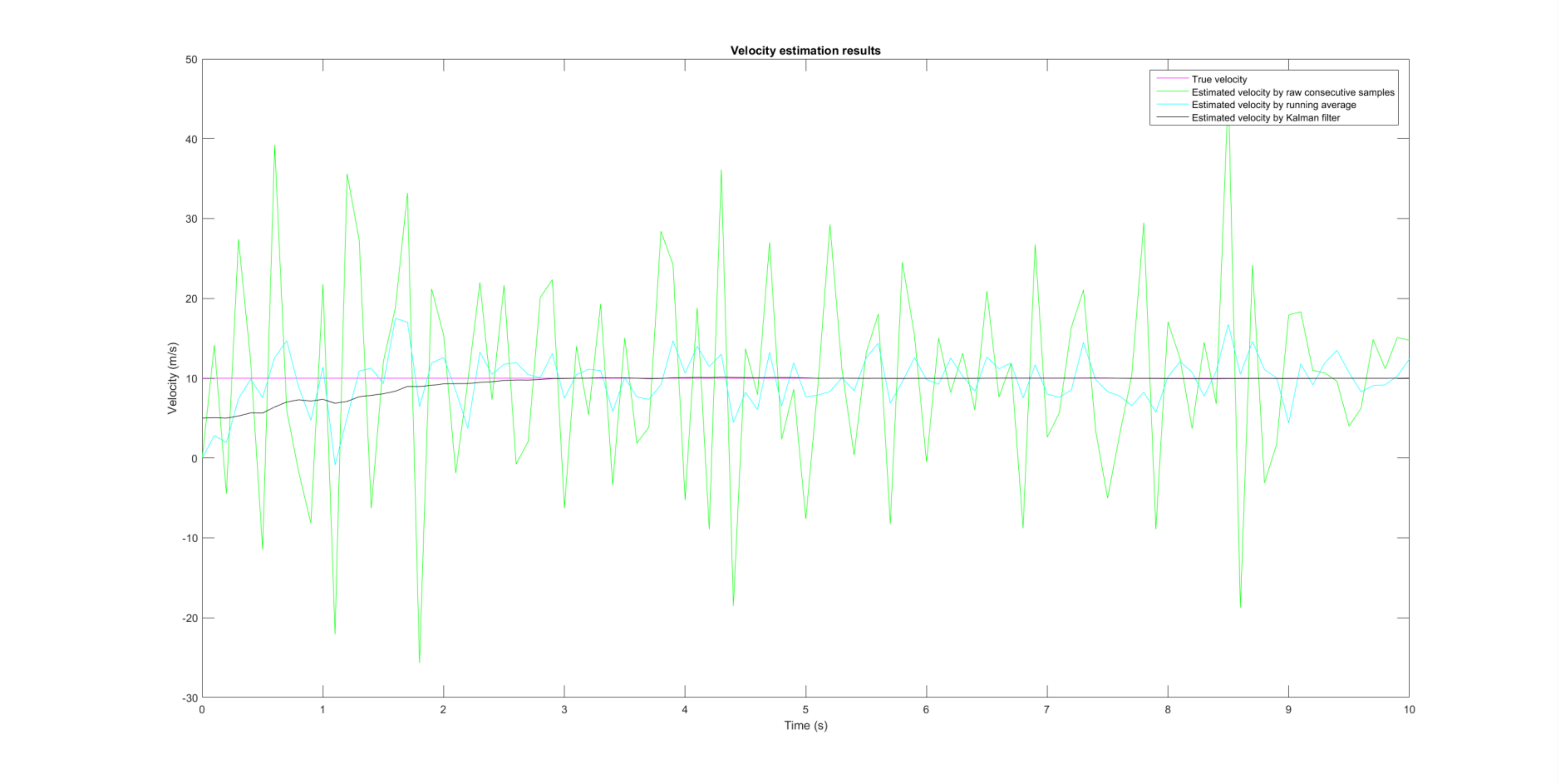
Track a Train using the Kalman Filter

- **Velocity Analysis**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Velocity analysis %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The instantaneous velocity as derived from 2 consecutive position
% measurements
InstantaneousVelocity = [0 (Z_buffer(2:Nsamples+1)-Z_buffer(1:Nsamples))/dt];

% The instantaneous velocity as derived from running average with a window
% of 5 samples from instantaneous velocity
WindowSize = 5;
InstantaneousVelocityRunningAverage = filter(ones(1,WindowSize)/WindowSize,1,InstantaneousVelocity);

figure;
plot(t,ones(size(t))*Vtrue,'m');
hold on;
plot(t,InstantaneousVelocity,'g');
plot(t,InstantaneousVelocityRunningAverage,'c');
plot(t,Xk_buffer(2,:),'k');
title('Velocity estimation results');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
legend('True velocity','Estimated velocity by raw consecutive samples','Estimated velocity by running average','Estimated velocity by Kalman filter');
```

Track a Train using the Kalman Filter

- **Velocity Analysis**



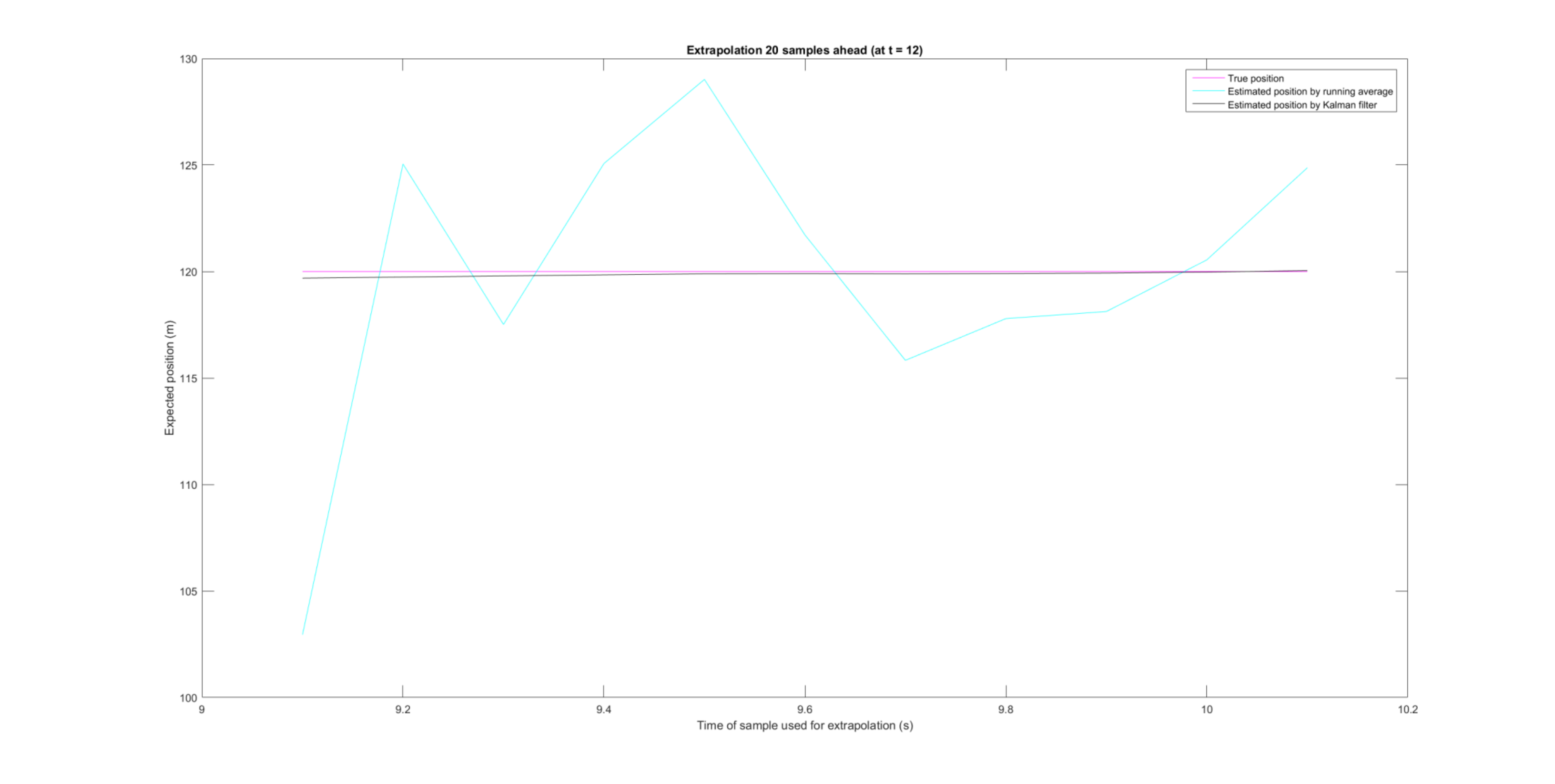
Track a Train using the Kalman Filter

- **Extrapolation ahead**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Extrapolation 20 samples ahead %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SamplesIntoTheFuture = 20;
Nlast = 10; % samples

% We take the last Nlast = 10 samples, and for each of these samples we try to see what would be the
% estimated position of the train at sample number Nsamples + SamplesIntoTheFuture
% if we took the position and the velocity that was known at that sample

TruePositionInTheFuture = Xinitial + (Nsamples + SamplesIntoTheFuture) * Vtrue * dt;

ProjectedPositionByRunningAverage = Xk_buffer(1,(Nsamples+1-Nlast):(Nsamples+1)) + ...
    ((SamplesIntoTheFuture+Nlast):-1:SamplesIntoTheFuture) .* dt .* InstantaneousVelocityRunningAverage((Nsamples+1-Nlast):(Nsamples+1));

ProjectedPositionByKalmanFilter = Xk_buffer(1,(Nsamples+1-Nlast):(Nsamples+1)) + ...
    ((SamplesIntoTheFuture+Nlast):-1:SamplesIntoTheFuture) .* dt .* Xk_buffer(2,(Nsamples+1-Nlast):(Nsamples+1));

figure;
plot(((Nsamples+1-Nlast):(Nsamples+1))*dt,ones(size(1:11))*TruePositionInTheFuture,'m');
hold on;
plot(((Nsamples+1-Nlast):(Nsamples+1))*dt,ProjectedPositionByRunningAverage,'c');
plot(((Nsamples+1-Nlast):(Nsamples+1))*dt,ProjectedPositionByKalmanFilter,'k');
title(['Extrapolation 20 samples ahead (at t = ' num2str((Nsamples + SamplesIntoTheFuture) * dt) ')']);
xlabel('Time of sample used for extrapolation (s)');
ylabel('Expected position (m)');
legend('True position','Estimated position by running average','Estimated position by Kalman filter');
```

Track a Train using the Kalman Filter

- **Extrapolation ahead**



Find out more

- http://www.kostasalexis.com/the-kalman-filter.html
- http://aerostudents.com/files/probabilityAndStatistics/probabilityTheoryFullVersion.pdf
- http://www.cs.unc.edu/~welch/kalman/
- http://home.wlu.edu/~levys/kalman_tutorial/
- https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python

- http://www.kostasalexis.com/literature-and-links.html

Thank you!

Please ask your question!