Solution to analysis in Home Assignment 2 (fixed 2d result)

Yongzhao Chen(yongzhao@chalmers.se)

April 20, 2023

1 A first Kalman filter and its properties

1.1 a

Generate the data points using the method from the small pre-tasks.

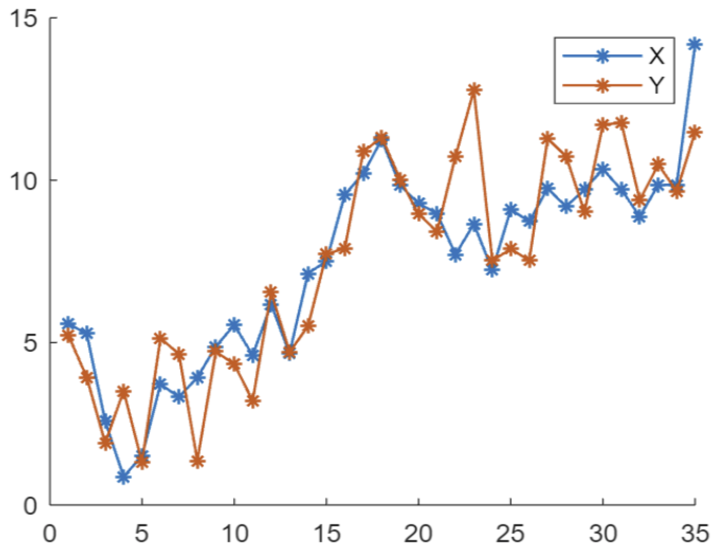
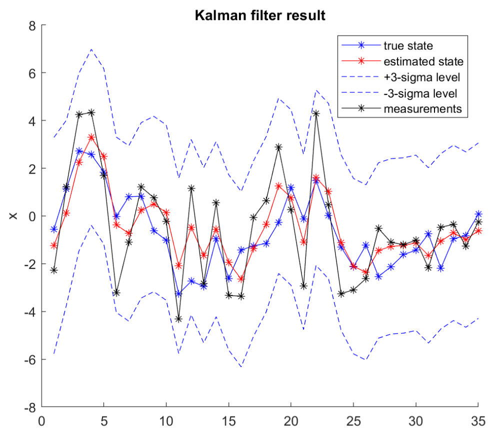


Figure 1: capital

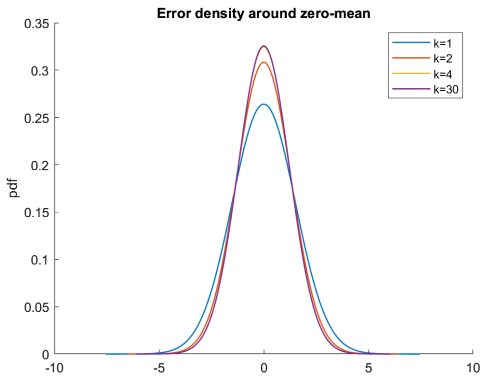
From figure 1, we can see that measurement Y is basically following the trend of model state X.

The 'matrix' A and H in this scenario are scalar 1, which means the curve of Y and the curve of X should coincide without noises, while there exist motion and measurement noises and the covariance of R is high compared to Q, some Y and X are very widely separated.

1.2 b



(a) Kalman filter result



(b) Error density around zero-mean

Figure 2: Figures for 1b

The first figure 2a shows the result of the Kalman filter, where the blue line represents the true state sequence, the red line represents the estimated state sequence, the black asterisks represent the measurements, and the blue dashed lines represent the $+/-3\sigma$ level of the estimated state sequence. As we can see, the estimated state sequence follows the true state sequence quite well and varies in a narrow zone. Since the measurement values nearly hit the 3σ borders and the borders change following the trend of the true state, it means the uncertainty represented by the error covariance is reasonable.

The second figure 2b shows the error density around zero-mean for time instances k = [1; 2; 4; 30]. The error density is represented by the Gaussian density with mean zero and standard deviation equal to the square root of the corresponding element of the error covariance matrix. As we can see, the error density becomes narrower and taller as the time instance k increases, which indicates that the uncertainty in the estimate decreases as we observe more measurements.

1.3 c

In this part, we give the Kalman filter a wrong prior and see what happens.

One thing that needs to care about here is that we only need to change the *x_0* for `kalmanFilter` because the true state is something really exists in the real world that does not change according to some wrong prior. The prior should only influence the filter.

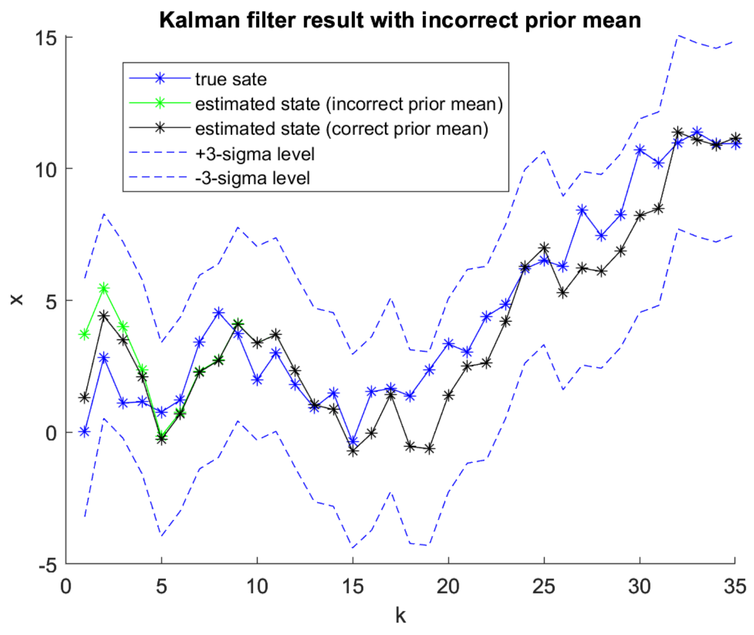


Figure 3: Kalman filter result with incorrect prior mean

In the plot, we can see that the estimates generated by the Kalman filter with the incorrect prior mean (green line) are initially far from the true states (blue line) and the true estimate state(black line). However, as more measurements are observed, the estimates converge toward the true states. After 10 new measurements, the incorrect estimate totally merges into the correct measurements. This means that while the estimates generated by the Kalman filter with the incorrect prior mean will deviate from the true states initially, they will become more accurate over time. If the time is long enough, there shouldn't be any difference between the incorrect estimate and the correct estimate.

1.4 d

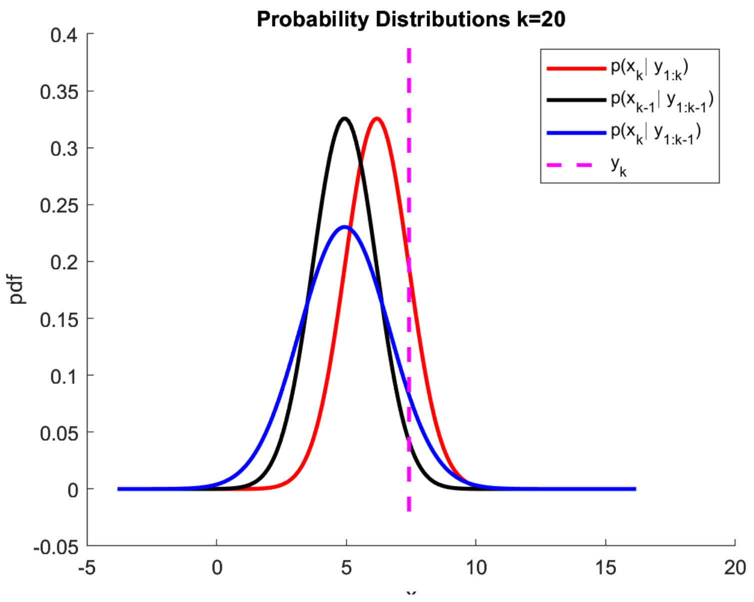


Figure 4: Four curves

In figure 4, we can see a clear process of updating the probability distribution of the state.

Prior information is depicted by $p(x_{k-1}|y_{1:k-1})$, which is the black curve on the left.

The blue curve is $p(x_k|y_{1:k-1})$, the estimated probability, the prediction. From the figure, we can see it has more uncertainty than the prior and has the same mean with the prior.

The measurement, y_k, the pink dotted line on the right, offers more information to us and gets to the current state posterior $p(x_k|y_{1:k})$. It 'drags' the prior curve to the right.

The red curve $p(x_k|y_{1:k})$ is posterior. We can see the mean of posterior is between prediction and measurement, and it has less uncertainty that it has a narrower shape then prediction.

Overall, the behavior of the prediction and update steps of the Kalman filter seems reasonable based on this plot.

1.5 e

1.5.1 Error $x_k - \hat{x}_{k|k}$

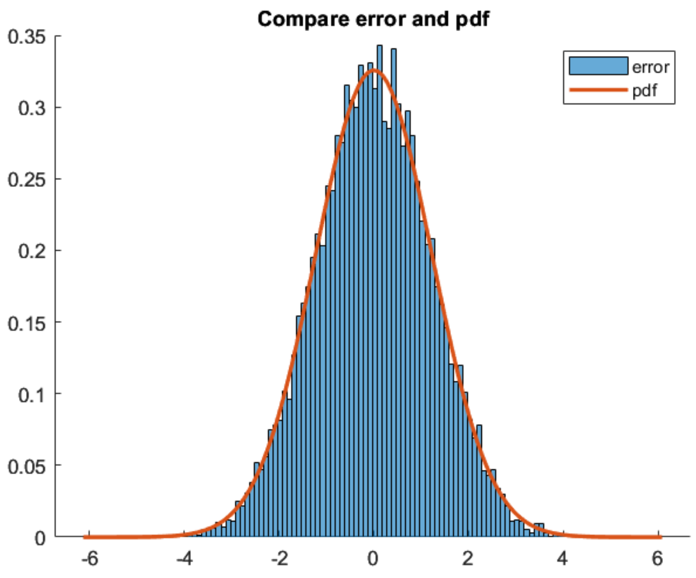


Figure 5: Compare the histogram to the pdf $N(x; 0, P_{N|N})$

From figure 5, we can see after the normalization, the histogram matches the pdf curve.

This result supports the theory behind the Kalman filter that the estimation errors are normally distributed and that the filter provides optimal estimates with respect to the mean squared error criterion. Moreover, the fact that the histogram matches the expected pdf also suggests that the filter is working correctly and that the covariance matrix $P_{N|N}$ represents the uncertainty in the estimates well.

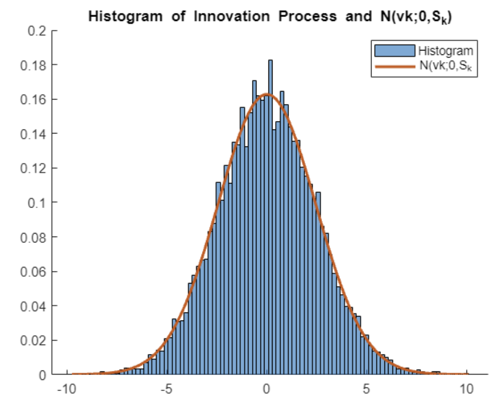
1.5.2 Mean and Correlation of v_k

From the lecture, the Kalman filter should have innovation consistency, which means v_k should satisfy:

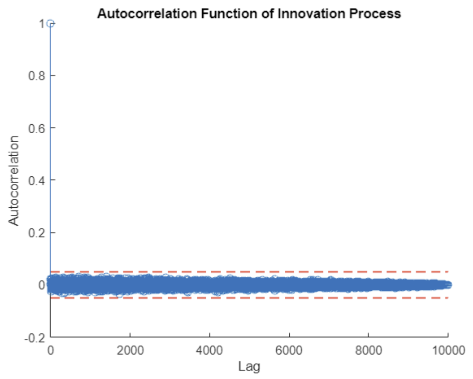
$$p(\mathbf{v}_k|\mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{v}_k; \mathbf{0}, \mathbf{S}_k)$$

$$Cov(v_k, v_{k-i}) = \begin{cases} cov\{v_k\} & \text{if i=0} \\ 0 & \text{otherwise} \end{cases}$$

Based on these two properties, I drew two pictures as in figure 6.



(a) Histogram of Innovation Process and $N(v_k; 0, S_k)$



(b) Autocorrelation Function of Innovation Process

Figure 6: Innovation Consistancy

From figure 6a, we can see that the histogram of innovation process fits the pdf $N(v_k; 0, S_k)$.

The two red lines in figure 6b are +/- 0.05, which shows that the innovations are not correlated except at the *0* position, so it tells the process fits the property of cov.

Since the consistency condition is satisfied, it implies that the Kalman filter is optimal in the sense that it provides the best linear unbiased estimate of the state given the measurements and that the estimation error is the minimum mean square error.

2 Tuning a Kalman filter

2.1 a

The model is :

$$y_k^v = C(v_k + r_k^v) \tag{1}$$

From the model we can get:

$$
\begin{aligned}
mean(y) &= C * mean(v_k) + C * mean(r_k) \\
var(y) &= C^2 * var(r_k)
\end{aligned}
\tag{2}
$$

Base on equation 2, we can get three diffent value of C and then calculate the final mean of C by calculating the average mean of C:

```
data=[CalibrationSequenceVelocity_v0;
   CalibrationSequenceVelocity_v10;
   CalibrationSequenceVelocity_v20];
meanv0 = mean(data(1,:));%c*mean(rk)
meanv10 = mean(data(2,:));%c*10+c*mean(rk)
meanv20 = mean(data(3,:));%c*20+c*mean(rk)
C(1)=(meanv10-meanv0)/10
C(2)=(meanv20-meanv10)/10
C(3)=(meanv20-meanv0)/20
C=mean(C)
```

Still according to equation 2, get covariance of r_k by calculating the average mean r_k:

```
for i =1:3
meanv(i) = mean(data(i,:));
cov(i) = var(data(i,:))/C^2
end

P_r = mean(cov)
```

The final results are:

$$
\begin{aligned}
C &= 1.1059 \\
P_r &= \mathrm{Var}[r_k^v] = 2.4702
\end{aligned}
\tag{3}
$$

2.2 b

I have come up with two ideas to solve the lack of $Y_{position}$ problem.

The first one is to use prediction to fix the lack of $Y_{position}$.To be detailed, if there is only a speed measurement available, use the predicted state estimate to estimate the position and update the state estimate and covariance matrix based on the estimated position and the speed measurement.

This method may make some sense because it can use every $Y_{velocity}$, but the missing data is replaced with estimated values, in which way imputing missing data may introduce bias or other errors in the analysis.

The second method is to skip the Y data couple if the $Y_{position}$ is missing. In this way, we will lose half of the $Y_{velocity}$ but it does not have the risk of introducing unnecessary perturbations.

Finally, I applied the second method, and create my `kalmanFilterfusedskip` function below:

```
    function [X, P,v] = kalmanFilterfusedskip(Y, x_0, P_0, A, Q
       , H, R, N)
    N = size(Y,2);

    n = length(x_0);
    m = size(Y,1);

    % Initialize state estimate and covariance
    X(:,1) = x_0;
    P(:,:,1) = P_0;

    for i=1:N
        % Prediction step
        [prex, preP] = linearPrediction(X(:,i), P(:,:,i), A, Q)
           ;

        if  isnan(Y(1,i)) % Check if position measurement is
           available
            % Update step
            [X(:,i+1), P(:,:,i+1),v(:,i)] = linearUpdate(prex,
               preP, Y(:,i), H, R);
        else
            % Skip update step if position measurement is missing
            v(:,i)=0;
            X(:,i+1) = prex;
            P(:,:,i+1) = preP;
        end
    end

    % Remove the first element of X and P (the initial state estimate
      and covariance)
    X = X(:,2:end);
    P = P(:,:,2:end);
    v=v(:,1:2:end);
    end
```

2.3 c

For `CV model`, the matrice are define like this:

```
    A=[1 Ts;0 1];%A for state update
    H=[1 0; 0 C]; %H for y=Hx
    Qcon=1;
    R=diag([Qcon C^2*cov_rvk]);%noise covariance for measurement
    % R=diag([16 16]);
    gamacv=[0;1];
    Qc=0.0001;%covariance of continuous time motion update
    Q=gamacv*Qc*gamacv';%discrete time covariance, motion
    x_0=[0 0]'; % just for prior, set it myself
    P_0=diag([10 10]);%just for prior , set it myself
```

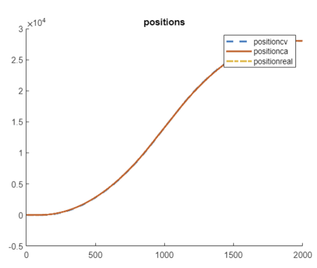
For `CA model`, the matrice are define like this:

```
    Aca = [1 Ts Ts^2/2;0 1 Ts;0 0 1];
    Hca = [1 0 0; 0 C 0;];
    gamaca=[0;0;1];
    Qca=gamaca*Qc*gamaca';%discrete time covariance, motion
    x_0ca=[0 0 0]'; % just for prior, set it myself
    P_0ca=diag([10 10 10]);%just for prior , set it myself
```

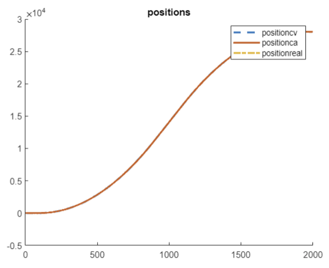
In the matrice above, the most important value that affects our results the most is Qc. The bigger the Qc, the more we believe in the measurement. The smaller the Qc, the more we believe in the process.

Also, the prior affects something, especially in the beginning part of the results. But as the same result from scenario 1, after enough process, the estimate will converge to the true estimate that because our step is 2000 that these priors do not matter at all.

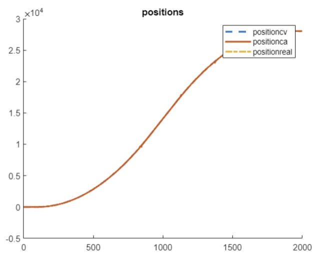
It can not tell from the plot when Qc is within 1 to 4, so I set Qc to some extreme value to see the difference:



(a) positionQc=0.0001

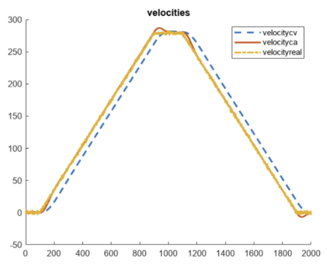


(b) positionQc=1

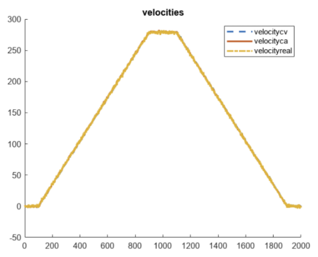


(c) positionQc=10000

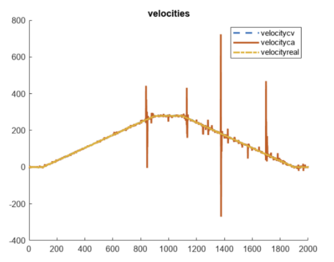
Figure 7: Change Qc and Observe Position curves



(a) velocityQc=0.0001



(b) velocityQc=1



(c) velocity when Qc=10000

Figure 8: Change Qc and Observe Velocity curves

From figure 8, the change of Qc mainly affects the velocity prediction of both models.

After attempts, the best strategy is to keep Qc around a small value, which means the model believes more in the process, then the curves of both models converge to the real values.

2.4 d

After tuning the constant velocity (CV) and constant acceleration (CA) models and comparing their performance, it appears that the `CA model` performs better in this scenario.

Advantages of CV model	**Disadvantages of CV model**
Simpler model with fewer parameters to tune	May not perform well in scenarios where the acceleration of the object being tracked is changing rapidly
Suitable for scenarios where the acceleration of the object being tracked is assumed to be constant	May not accurately capture sudden changes in velocity or direction
Smoother estimates of the position and velocity	

Table 1: Advantages and disadvantages of the CV model

Advantages of CA Model	**Disadvantages of CA Model**
Can capture rapid changes in acceleration and velocity	More complex model with more parameters to tune
More flexible model that can handle varying acceleration scenarios	May produce noisier estimates due to the increased complexity

Table 2: Advantages and disadvantages of the CA model.

In this scenario, from the true state value we can see the train is assumed to have constant acceleration, so the `CA model` performs better than the `CV model`.

As for the quantitative analysis, I have calculated the corresponding vk inside the code, which can be used to analyze the quality of the corresponding model through the `autocorr` function. The result are ploted as figure 9 and figure 10, from which it is obvious that the `CA model` has better estimate quality in this scenario.

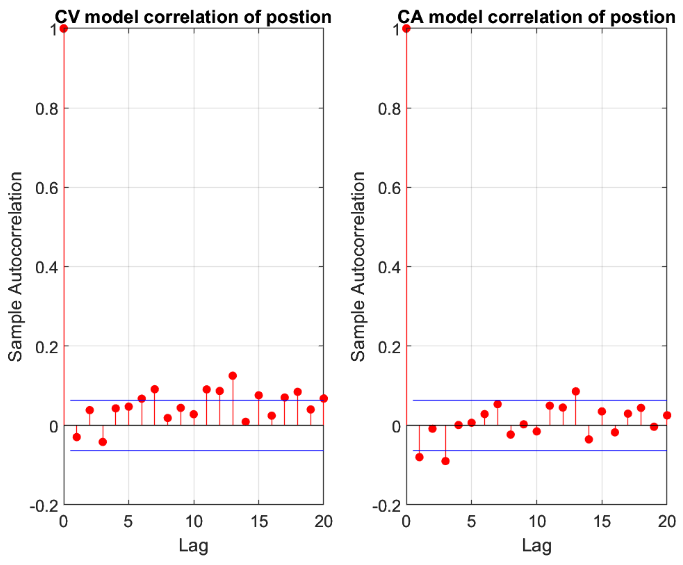


Figure 9: correlation of position

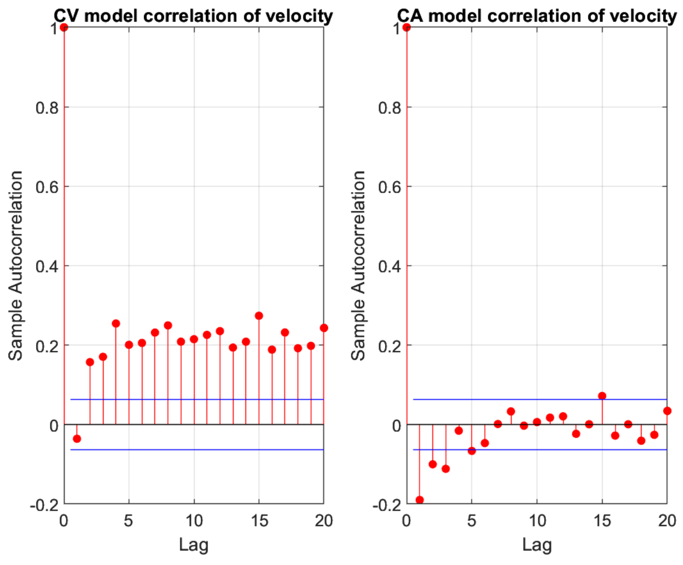


Figure 10: correlation of velocity

Overall, the choice of the appropriate model depends on the specific scenario being considered and the assumptions made about the object being tracked. For our specific scenario, we should use `CA model`.