Solution to analysis in Home Assignment 2.2

Weilong Chen + weilong

Analysis

In this report I will present my independent analysis of the questions related to home assignment 2.2. I have discussed the solution with Yongzhao Chen but I swear that the analysis written here are my own.

1 Scenario 1 – A first Kalman filter and its properties

a

The measurement model states that the measured samples are normally distributed as $p(y_k|x_k) = N(y_k; x_k, R)$, where R is the measurement noise covariance. This can be observed in the figure 1.1, as the measurements seem to be distributed normally around the true state sequence.

b

As it shows in figure 1.2, the filter produces reasonable estimates that closely track the true state. The error covariance matrix represents the uncertainty in the estimates, it provides a good estimate of the true uncertainty. This is achieved by looking at the 3-sigma level curves in figure 1.2, all the measurements lie inside this region.

The Kalman filter is a tool for estimating a system's current state. It works by combining what it knows about the system's past behavior with a model of how the system is expected to behave in the future. By doing this, it can make more accurate predictions about the system's current state based on the available measurements. Without this combination of past information and model predictions, the estimated states would be much less reliable and prone to errors.

To visualize the uncertainty of the output and how well the estimated values match the true state, figure 1.3 plots the posterior density and true state at several time instances. The accuracy of the posterior distributions is evident, as the true value has a significant likelihood compared to the maximum. This means that the distributions accurately represent the probability of the true state falling within a certain range.

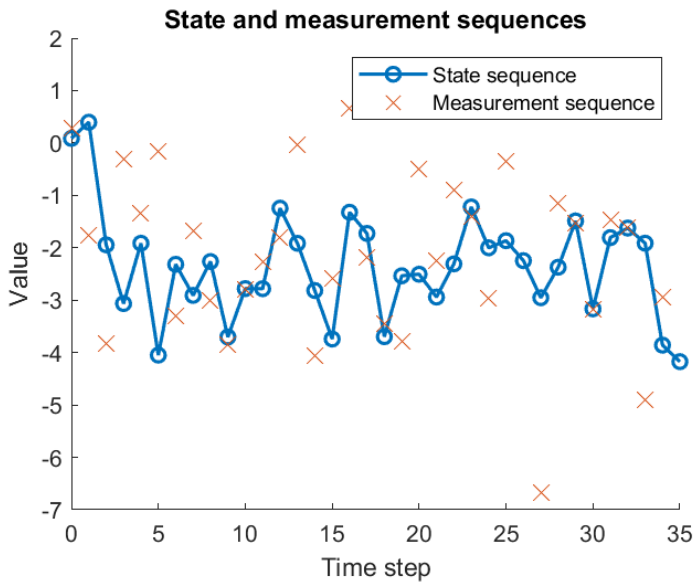

Figure 1.1: State and measurement from linear and gaussian model.

c

In figure 1.4, the plot reveals that the error state estimate gradually converges to the true state estimate, with the red curve starting with a wrong initial value. It appears that the Kalman filter is effectively estimating the state of the system over time, despite starting with an incorrect initial estimate. As the filter updates its estimates based on both measurements and the system model, it gradually converges to the true state, as evidenced by the plot of the sequence of estimates.

d

As we can see in figure 1.5 and 1.6, the mean of the prior distribution remains the same, but the covariance is increased to reflect the uncertainty associated with the predicted state. This increase in covariance is a reasonable outcome, given that the motion model provides an estimate of how the state is expected to change over time.

In the subsequent update step, the filter refines the estimate of the state based on both the predicted state and the observed measurement, resulting in an updated posterior distribution that reflects the most likely state of the system given the available information. That is why the mean of the posterior distribution is always located somewhere in between the actual measurement and the mean of the predicted distribution.

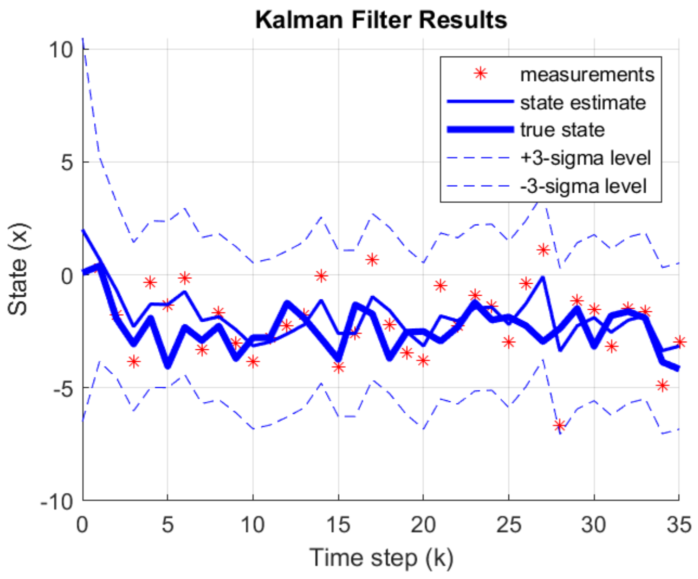

Figure 1.2: State, measurement sequence, state estimate and 3-sigma level.

e

In figure 1.7, the estimation error mean is close to zero, which indicates good filtering performance. The normal distribution $N(x; 0, PN|N)$ fits well with the histogram because the estimation error mean is close to zero and the estimation error covariance converges to a constant scalar.

As it's shown in figure 1.8, the auto-correlation at lag equal zero is much higher than at any other lags. This proves the conclusion that the innovation should ideally have a mean of zero and a covariance of zero for all l not equal to zero.

2 Scenario 2 – Tuning a Kalman filter

a

First, we can calculate the mean of each set of measurements. We can then use the mean values to find an estimate for C, the scaling constant. To do this, we take the mean of the mean velocity values for each of the three sets of measurements and divide by 3 and 10 (since the ideal average velocitie is 10). This gives us an estimate for C.

Next, we can calculate the covariance of the velocity sensor noise for each of the three sets of measurements. To do this, we divide the variance of each set of measurements by C^2. We then take the mean of the three covariance values to get an estimate for the variance of the velocity sensor noise Var.

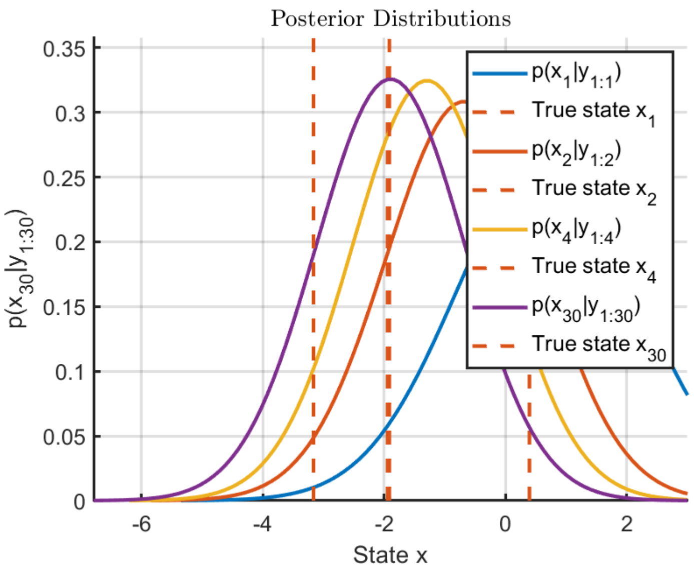

Figure 1.3: Posterior density and true state for $k = [1, 2, 4, 30]$.

Scaling constant C: 1.0999 Variance of the velocity sensor noise $Var[r_v k]$: 2.4972. The dataset plot is shown in figure 2.1.

b

Using the motion model to predict the position when a position measurement is not available can indeed introduce uncertainty into the measurement dataset. Therefore, it is not a recommended approach for handling missing data in the Kalman filter.

It's reasonable to skip the NaN data directly is an acceptable solution. However, this is not always an optimal solution, as it leads to missing data and reduces the effectiveness of the sensor fusion.

This can be achieved by directly modify the generated dataset.

c

Try to estimate the autocorrelation function of the measurement noise to get an idea of the correlation structure of the noise. This is important as it will help us determine the appropriate filter type and model order to use for the tracking problem.

Once we have estimated the ACF, we will then select the appropriate model type (i.e., constant velocity or constant acceleration) based on the level of correlation in the noise. If the noise is highly correlated, the constant acceleration

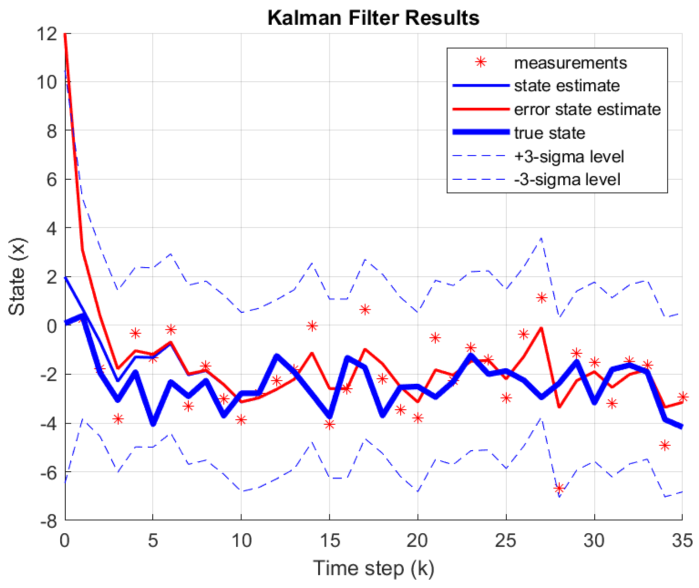

Figure 1.4: State, measurement sequence, state estimate, 3-sigma level, error initial curve as well.

model may work better, as it takes into account changes in acceleration over time.

As we can see in figure 2.2 and 2.3, when the Q term is small. The autocorrelation is out of boundary, which means we should increase the Q term, the figure 2.4 shows if we increse the Q term, it will work better. While in our example, the cv and ca model both work well compared to the generated data.

d

The position and velocity updation figures are shown in figure 2.5 and 2.6. CA model is more robust in this example. The velocity is not constant, although the cv model works by tuning. It's resonable to select CA in this case with more physical meaning. Besides, according to the autocorrelation diagram, the CA model works better than CV model as well.

Generally, the advantages of CV model is it can be appropriate for tracking targets that move at a relatively constant speed. It is also a simple model with only one parameter to tune. While the disadvantage is that it does not take into account changes in acceleration over time, which can result in errors if the target undergoes sudden changes in speed. It may also have difficulty tracking targets that change direction frequently or move in a nonlinear path.

For CA model, the advantage is that it takes into account changes in accel-

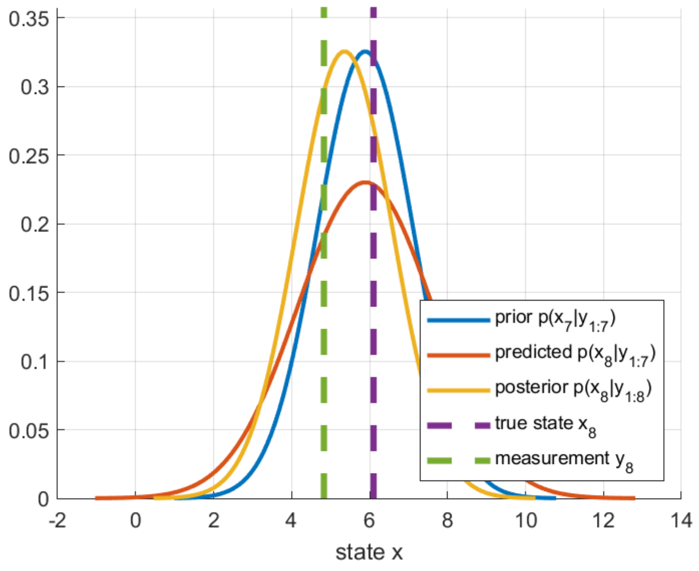

Figure 1.5: Prior, predicted and posterior density at k = 8.

eration over time, which can improve tracking accuracy for targets that undergo changes in speed, which means it can handle targets that move in a nonlinear path or change direction frequently. The disadvantage is it is a more complex model with two parameters to tune. It still has limitations, because it assumes a constant acceleration over time, which may not be appropriate for targets that experience sudden changes in acceleration.

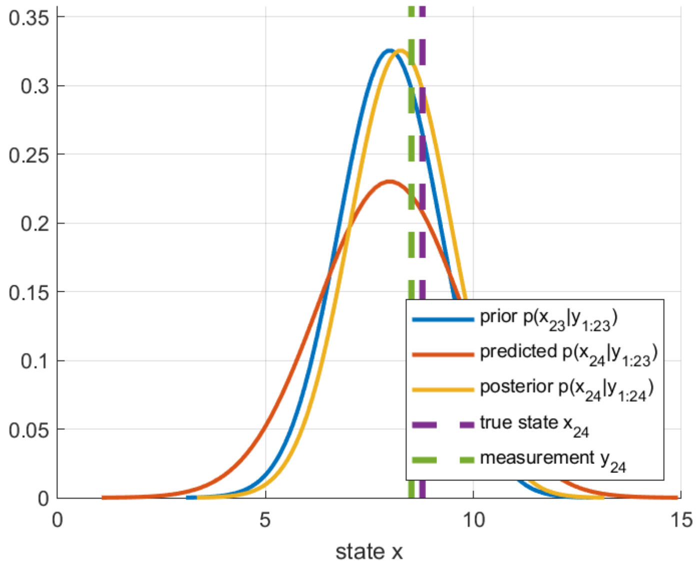

Figure 1.6: Prior, predicted and posterior density at k = 24.

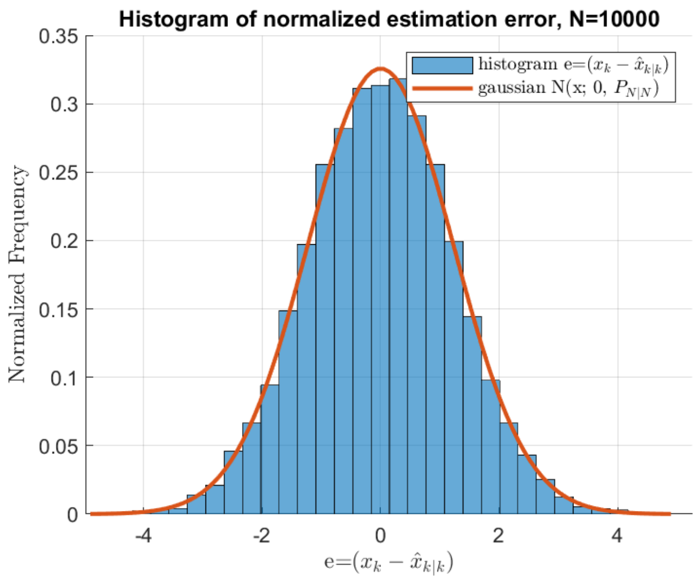

Figure 1.7: Comparing Kalman Filter Estimation Error to Probability Density Distribution for State Sequence of Length 1000

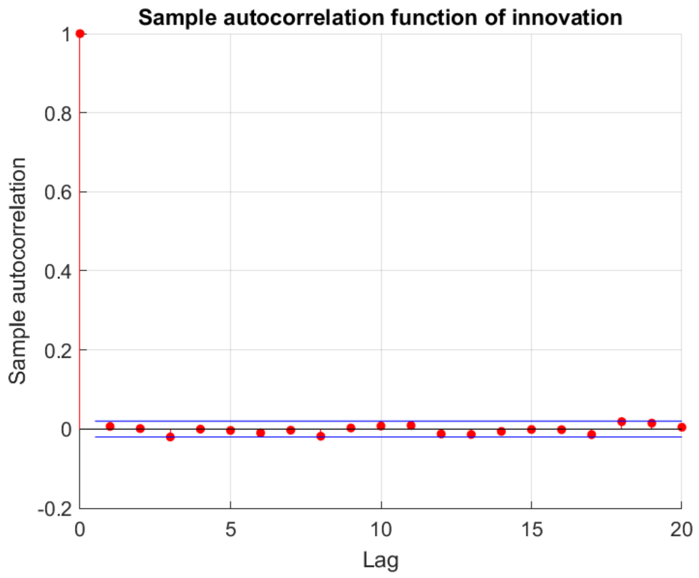

Figure 1.8: Normalized Auto-Correlation of Kalman Filter Innovation Process for 1000-Sample State and Measurement Sequence

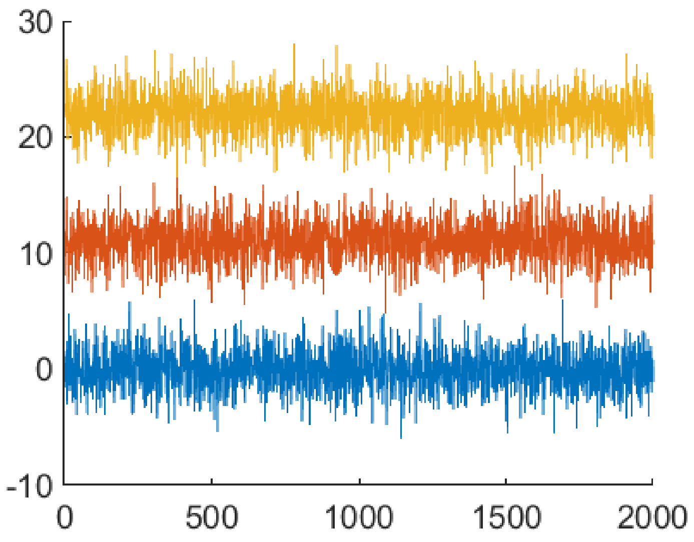

Figure 2.1: Calibration Data for Velocity Sensor

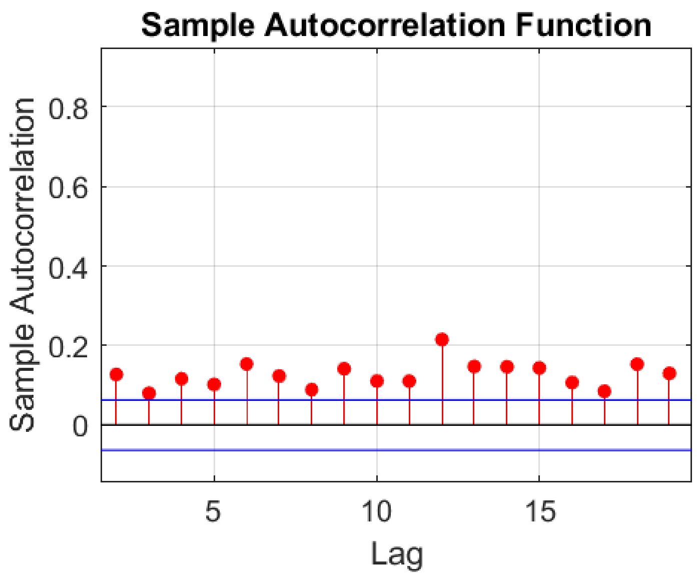

Figure 2.2: Q=[0,0,0,1] for cv position

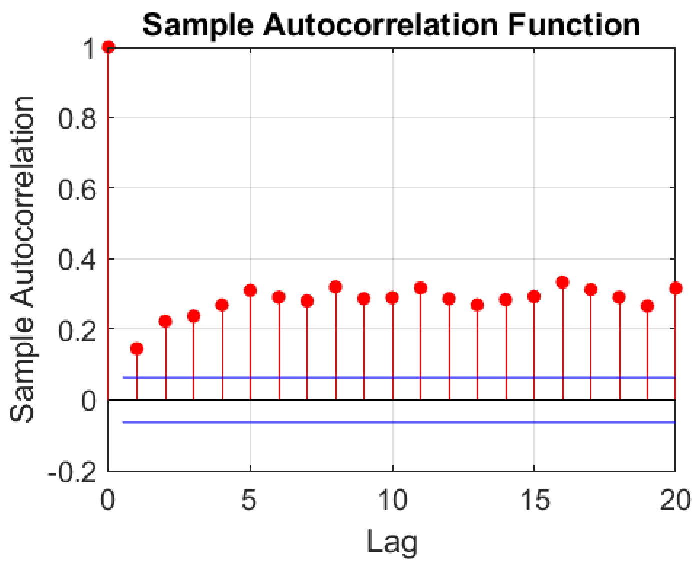

Figure 2.3: Q=[0,0,0,1] for cv velocity

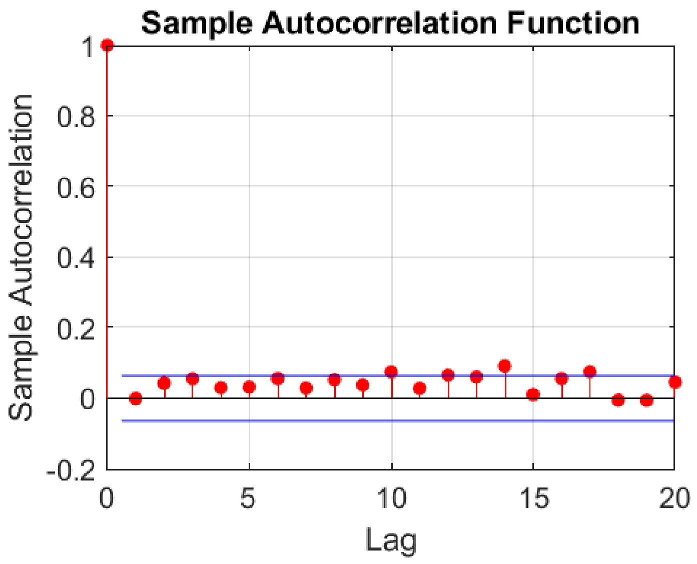

Figure 2.4: Q=[0,0,0,4] for cv velocity

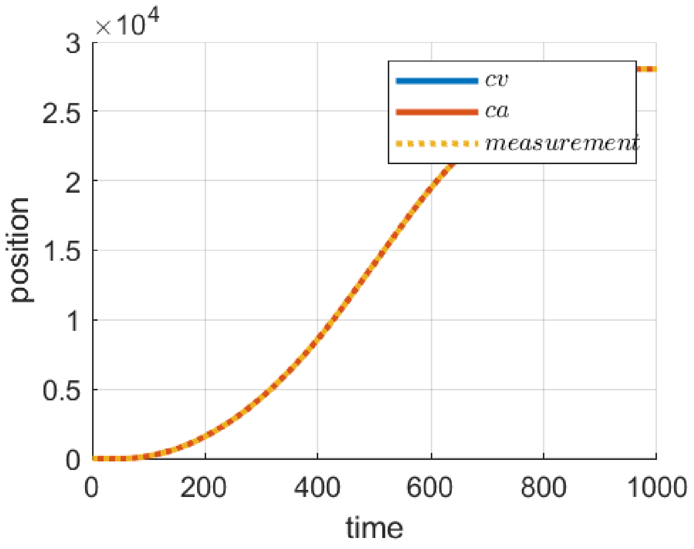

Figure 2.5: positon with cv, ca and data

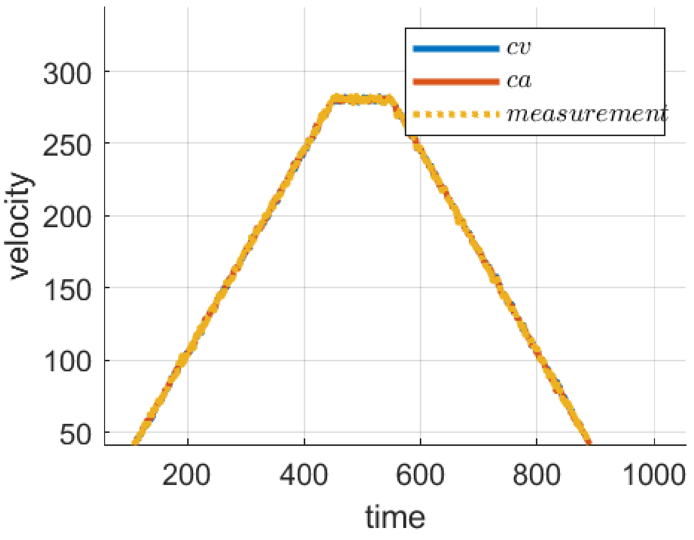

Figure 2.6: velocity with cv, ca and data