Solution to analysis in Home Assignment 4

Nicholas Granlund, nicgra@student.chalmers.se

Analysis

In this report I will present my independent analysis of the questions related to home assignment 1. I have discussed the solution with Arvid Petersen, Louise Olsson and I swear that the analysis written here are my own.

1 Smoothing

Task a) filter and smooth the state trajectories

By replicating the steps from HA3, question 3 we can generate the turning motion. We then use the developed function nonLinRTSsmoother() to obtain the filtered estimates and smoothed estimates for the state trajectories. The filter and smoother types are EKF. The positions can be seen in Figure 1.1-1.3 below.

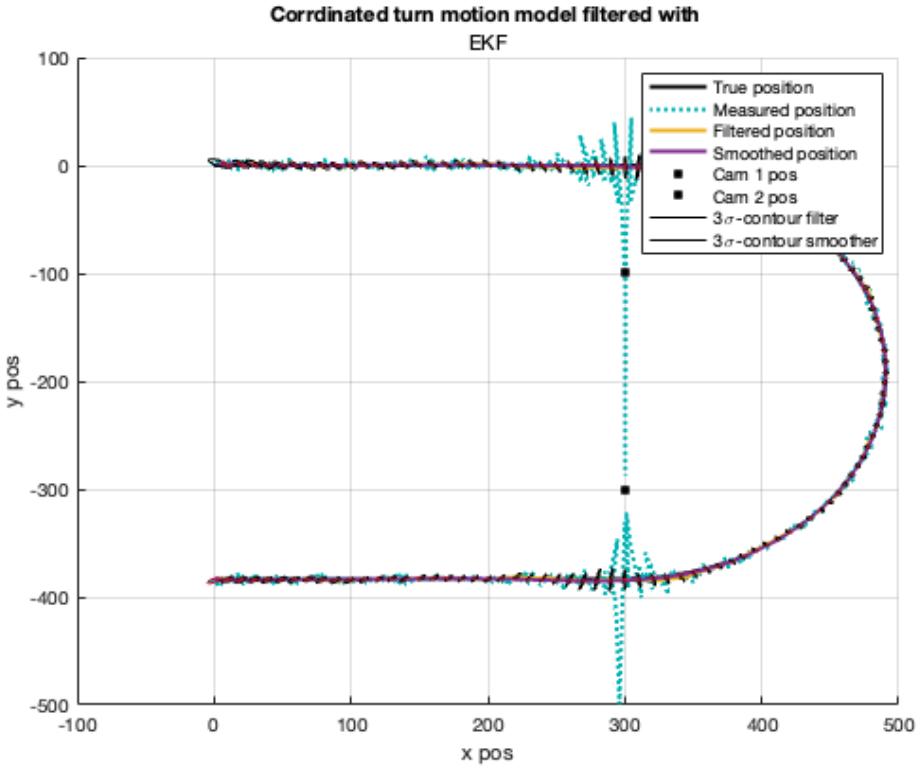

Figure 1.1: Coordinated turn motion model. Filtered and smoothed position with EKF

The Figure 1.1 illustrates the coordinated turn and the estimated position for the filtering and also the smoothing. As can be seen, tthe filtering and smoothing can track the actual position rather well. However, to see a larger difference, we have to look closer. The Figure 1.2 illustrates an close up of the Figure 1.1 around $x = 260$

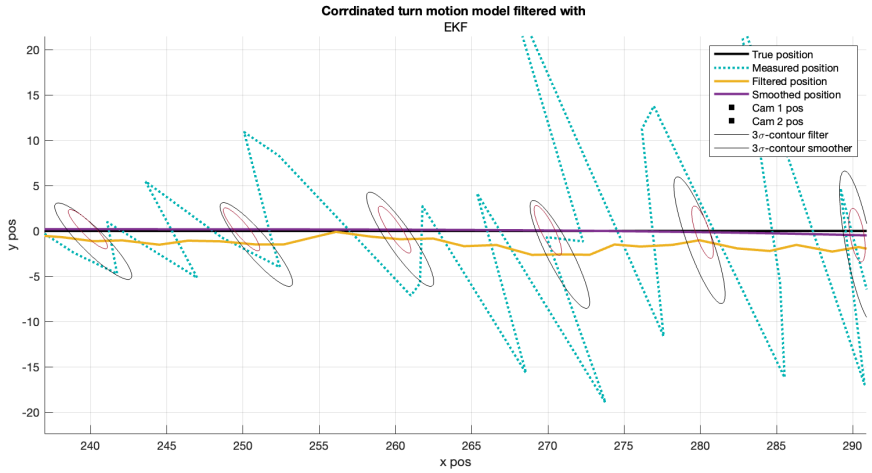

Figure 1.2: Close up of Coordinated turn motion model. Filtered and smoothed position with EKF.

Figure 1.2 gives an better illustration of what has occured. It can be seen that the filtered estimate deviates more from the actual position compared to the smoothed estimate, this is to be expected. The covariance for the smoothed estimate is also smaller than the filtered estimate covariance. The covariance for the smoothed estimate coincide with the covariance of the filtered estimate, but is smaller in both x and y.

furthermore we can consider the other filtered and smoothed state trajectories. These can be seen in Figure 1.3 below.

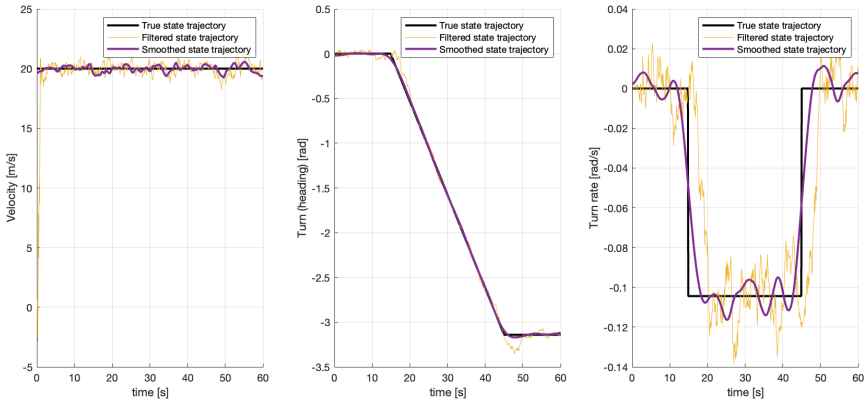

Figure 1.3: State-trajectories filtered and smoothed position with EKF.

Figure 1.3 shows that the smoothed estimate is superior for all state-trajectories. It is most evident in the turn-rate (rightmost plot) where the filtered estimate is corrupted by a large amount of noise, whereas the smoothed manages to track the true trajectory with higher precision.

Task b) Introduce an outlier at $k = 300$, how does the filter and smoother react to this?

By introducing an outlier manually into the measurements at $k = 300$, (in the middle of the turn), we get different filtered and smoothed state trajectories. Figure 1.4 illustrates the how the estimate has changed.

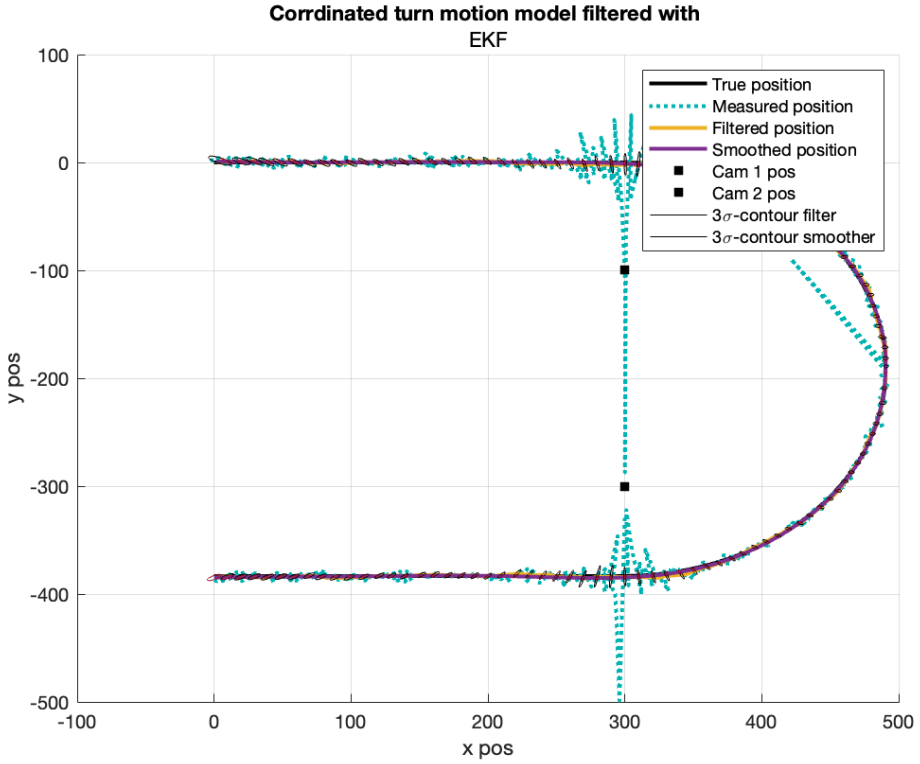

Figure 1.4: Coordinated turn motion model with an outlier at $k = 300$. Filtered and smoothed position with EKF

According to this plot, it seems like the outlier does not have such a large impact on the positional estimate, not for the filter nor smoother. If we enlarge the positional trajectory at $k = 300$ we can observe some more changes. The close up can be seen below in Figure 1.5

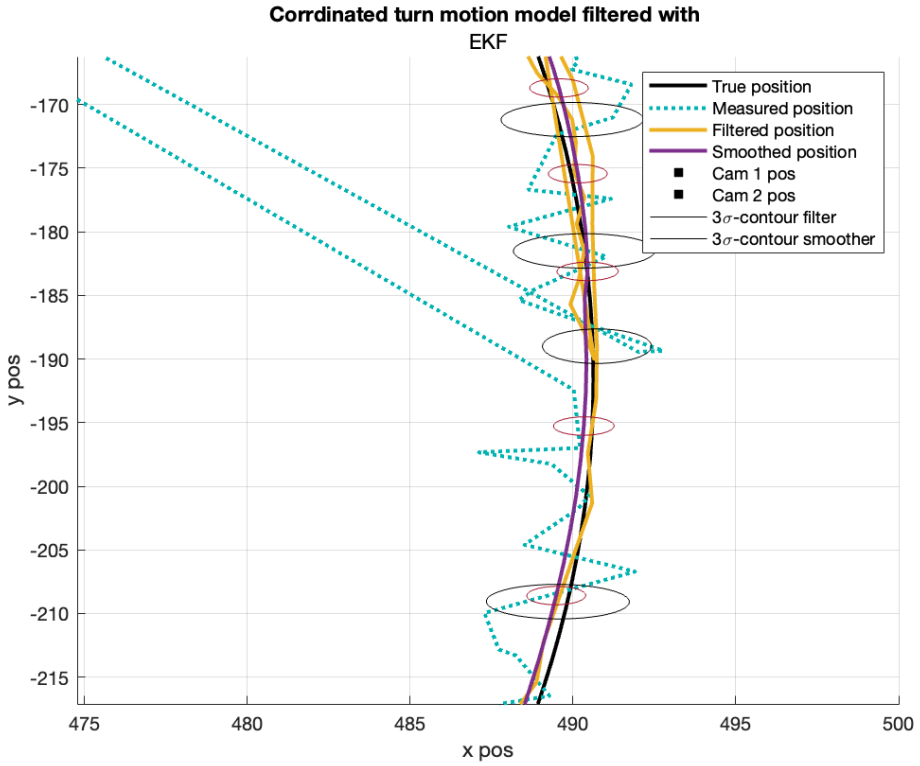

Figure 1.5: Coordinated turn motion model with an outlier at $k = 300$. Filtered and smoothed position with EKF

In Figure 1.5 it is illustrated how the measurement is far away from the actual position and the differences become rather clear.

This outlier will cause the filter to estimate the position completely wrong. According to the filter, the position has completely turned and is heading the other direction. If one observes the smoothed estimate, one can see that the estimate is heading in the right direction. The outlier avidently has great impact on the filtered estimate whereas the smoothed estimate seems more robust against these tyoes of errors. To examine further we consider the other state-trajectories for this scenario. These can be seen in Figure 1.6 below.

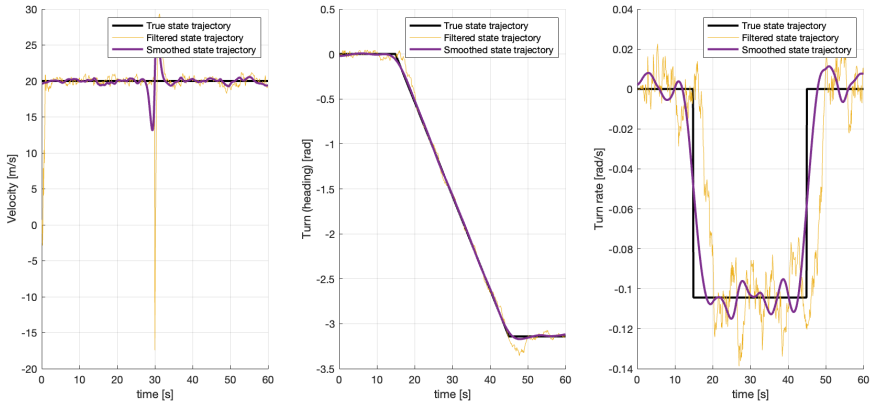

Figure 1.6: State-trajectories filtered and smoothed position with EKF. With an outlier at $k = 300$.

In the above figure, it is even clearer how the outlier has affected the estimate. In the

leftmost plot where the velocity is estimated, it can be seen that the filtered estimate reaches a negative value, pointing the position in the completely oposite direction. One can also see that the smoothed estimate is affected, but the velocity maintains a positive value, tracing the true state velocity better.

2 Particle filters for linear/Gaussian systems

By considering the random walk motion model we are able to generate a state sequence. This state sequence along with the corresponding measurements are used for three different filters. Kalman filter, particle filter with and without resampling. The filtered estimates along with the true state can be seen in Figure 2.1 below.

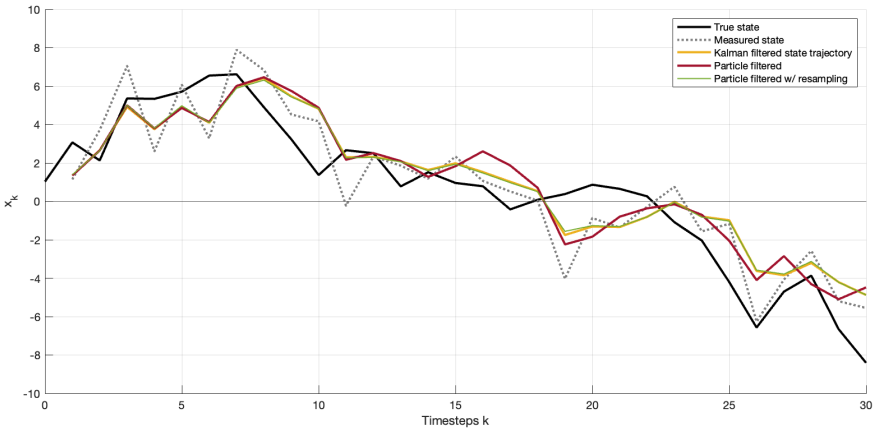

Figure 2.1: State trajectory with filtered estimates. The shown particle filter used $N = 1000$ particles for both the resampled and not resampled scenario.

The performance of the filters are evaluated by computing the MSE with regards to the true state. The MSE for different amount of particles can be seen in Table 1, where it is shown that it requires around $N = 1000$ particles to obtain a comparable accuracy to the Kalman filtered estimate. The amount of particles needed will however increase as the dimensions of the problem increase.

Table 1: MSE for different filters and particle amounts N.

	$N = 10$	$N = 100$	$N = 1000$	$N = 10000$
MSE KF	3.1221	3.1221	3.1221	3.1221
MSE PF	9.7278	8.0106	3.2332	3.3930
MSE PF w/ resampling	4.6000	3.0889	3.0468	3.1311

Interestingly enough, the particle filter with resampling performs rather well whilst only using $N = 10$ particles to trace the state, and using $N = 100$ particles, the performance is very similar to the Kalman filter. The particle filter without resampling shows a large deviation in performance and requires around $N = 1000$ to perform similarly to the other filters.

If we plot the covariances as errorbars, we get a sense of how the particles are distributed. The covariances in the estimates are seen in Figure 2.2

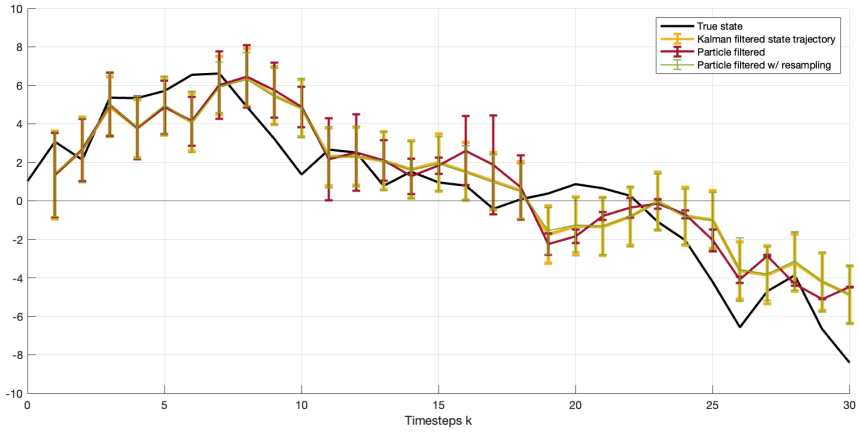

Figure 2.2: State trajectory with filtered estimates and covariances as errorbars. The shown particle filter used 1000 particles for both the resampled and not resampled scenario.

The most aparent thing to comment from this plot is that the particle filter without resampling has a rather low covariance. i.e the particle filter believes that it knows the position with high probability. This is why the MSE is highest for this type of filter, it is subject to trace the wrong path. The particle filter with resampling and the Kalman filter have similar covariances and performance.

Furthermore, we can plot the respective filters posterior distribution. Figure 2.3 below illustrates the posterior for the different filters evaluated at time instances $k = 1, 15, 30$ (top to bottom).

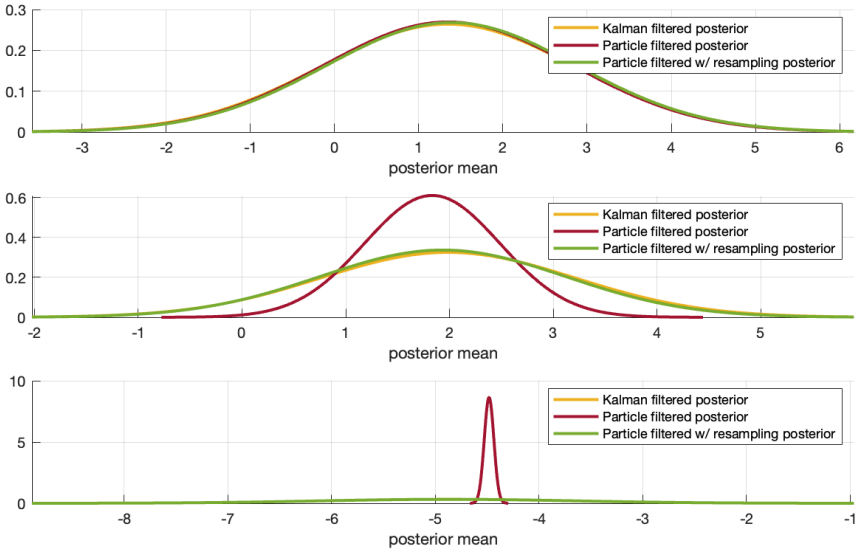

Figure 2.3: Kalman filter and particle filter posterior for time instances $k = 1, 15, 30$.

This figure illustrates the posterior and are equivalent to the Figure 2.2 at the given time instances. The main thing to notice here is that the KF an PF with resampling perform

rather similar at these time instances. Furthermore, the PF without resampling seems to yield lower covariances and thus approximates the mean with higher probability, even though the approximation is the worst of them all. This is clear in the last subplot at $k = 30$ where the estimation from the PF without resampling has exceptionally low covariance. (particle degeneracy)

Task b) Evaluate filter performance for incorrect prior.

By using the incorrect prior for the different filters we get different results than from the previous task. By giving the prior $x_0 \sim \mathcal{N}(-20, 2)$ we get the filtered estimates shown in Figure 2.4 below.

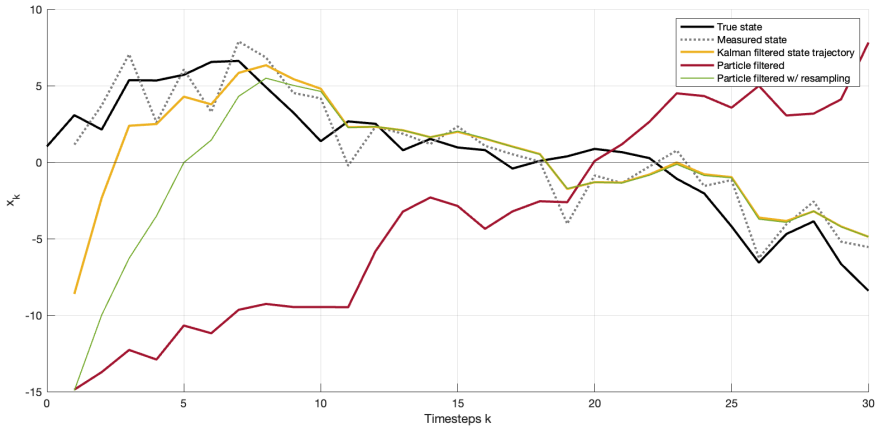

Figure 2.4: State trajectory with filtered estimates for incorrect prior. The shown particle filter used 1000 particles for both the resampled and not resampled scenario.

Evidently this has negatively impacted the performance of every filter. However, the Kalman filter and the particle filter with resampling manages to converge towards the most accurate path. The particle filter without resampling does not manage to converge towards the true trajectory and the estimate is far off. The estimation covariances can be seen in Figure 2.5 below.

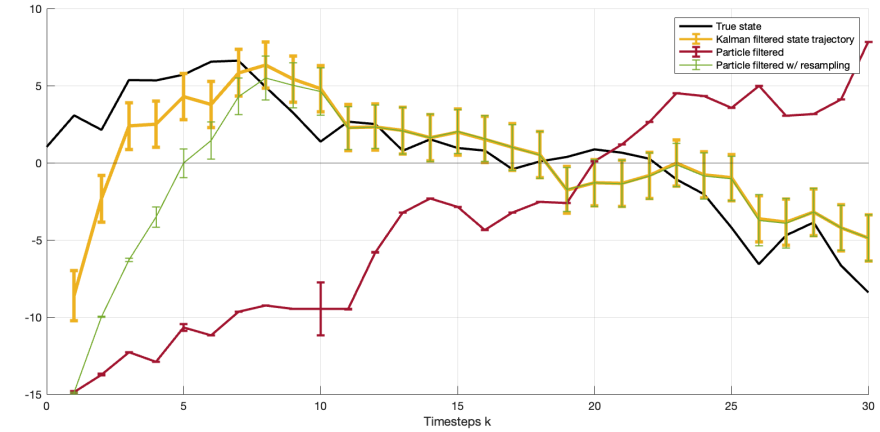

Figure 2.5: State trajectory with filtered estimates for incorrect prior plotted with covariance as errorbars. The shown particle filter used 1000 particles for both the resampled and not resampled scenario.

According to the above figure we can see that the covariances in the Kalman filter estimate and Particle filter with resampling are comparable. Interestingly enough, the particle filter without resampling has the lowest covariance of them all. This indicates that the estimate has gathered particles along the wrong trajectory, thus incresing the estimate probability and certainty, even though it is completely wrong. This is why the particle filter with resampling performs better - because the resampling will spread out the particles again and avoid tracing the unbeknownst wrong path.

Task c) Illustrate particle path without resampling

By tracing the path every particle takes we get a sense of how well the filter accurately tracks the true state. By using 100 particles and plotting their path for a Filter without resampling we get the result shown in Figure 2.6.

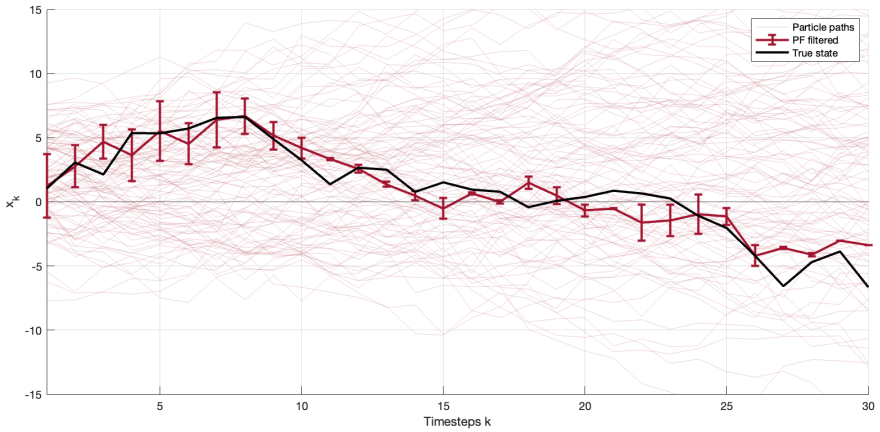

Figure 2.6: Particle path for particle filter without resampling. $N = 100$ particles was used for $k = 30$ timesteps.

As the plot indicates, the particles are spread out and thus decreases the accuracy. This filter has been given the right prior, and manages to trace the path with satisfying performance eitherway. However since the particles cannot trace the true state accurately, it is expected that the result will be worse if it is simulated for more timesteps. A plot for $k = 300$, i.e 10x more timestpes, is shown in Figure 2.7 below.

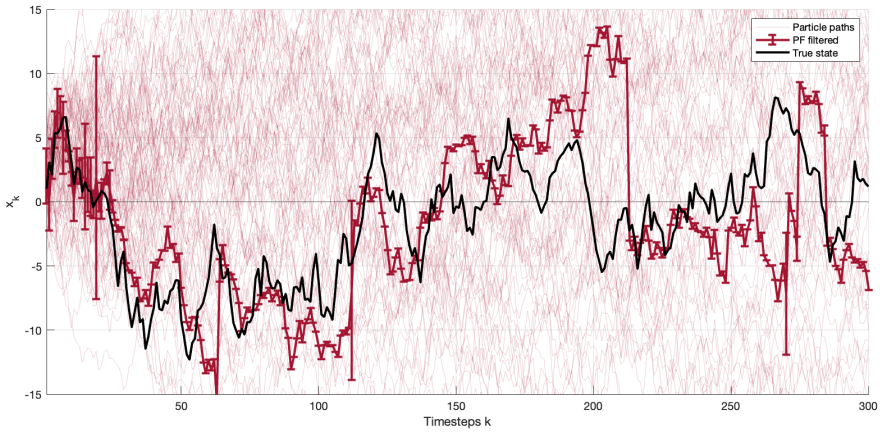

Figure 2.7: Particle path for particle filter without resampling. $N = 100$ particles was used for $k = 300$ timesteps.

This plot shows that the particle filter without resampling will generate poor estimates for higher k.

Task d) Illustrate particle path with resampling

If we instead use resampling we are better able to trace the true state. The results for particle tracing using resampling are shown in Figure 2.8 below.

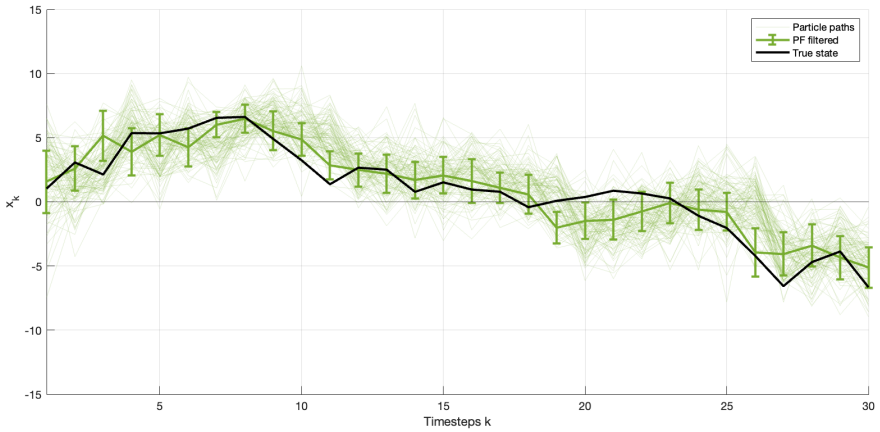

Figure 2.8: Particle path for particle filter with resampling. $N = 100$ particles was used for $k = 30$ timesteps.

The performance of this estimate is far better than the non-resampling counterpart. The particles stick together and do not deviate from eachoter significantly. This is also valid for longer durations of tracing as illustrated by Figure 2.9 below.

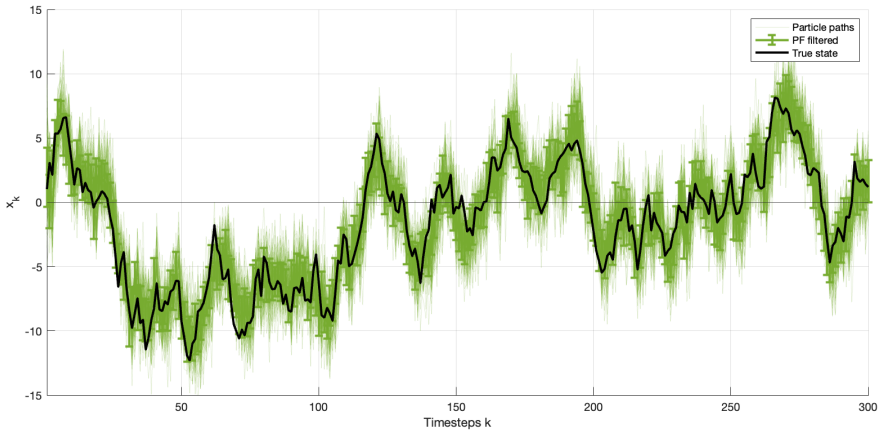

Figure 2.9: Particle path for particle filter with resampling. $N = 100$ particles was used for $k = 300$ timesteps.

By using resampling we are better able to track the true trajectory. This is because the resampling step helps in avoiding the problem of particle degeneracy. This is helpfull when most of the particles have low weights. This means that non of the particles are representing the true state with high probability. By resampling we use the weights of the particles to determine which particles we discard and which we keep. The discarded particle weights will then be distirbuted among the remaining particles, increasing the overall performance of the filter. It essentially makes the filter avoid getting stuck tracking a local minima, which is what is happening for the particle filter without resampling.

/ Nicholas Granlund