Solution to analysis in Home Assignment 4

Yongzhao Chen(yongzhao@chalmers.se)

May 15, 2023

1 Smoothing

1.1 Task a

In this part, mainly use the code from HA3 Q3, but use `nonLinRTSsmoother` function to generate the filtered output and the smoothed output.

And I realized my last HA3 had some bug, after the fix, the tuning value is chosen as :

$$sigma_v = 10;$$
$$sigma_w = pi/180 * 6;$$

The result shown in figure 1, legends are detailed. To be more readable and to concentrate on the differences, the picture is zoomed in:

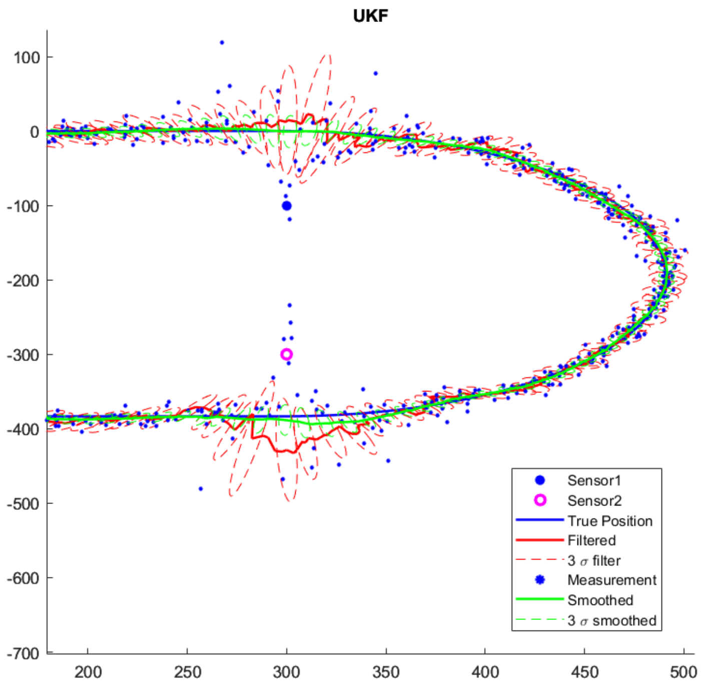

Figure 1: Task a: Comparison

1.1.1 Conclusion

From figure 1, the green curve, which represents the smoothed output is closer to the true position line compare with the red curve, which represents the filtered output.

And from the 5_{th} curves, the covariances of the smoothed output are smaller than the filtered one.

1.2 Task b

I increased the value at $k = 300$ by 10% when generating true state X to see the result:

```
    for i = 2:K+1
    if i ==300
    X(:,i) = 1.1*coordinatedTurnMotion(X(:,i-1),T);
    X(5,i) = 1.1*omega(i);
    else
    X(:,i) = coordinatedTurnMotion(X(:,i-1),T);
    X(5,i) = omega(i);
    end
end
```

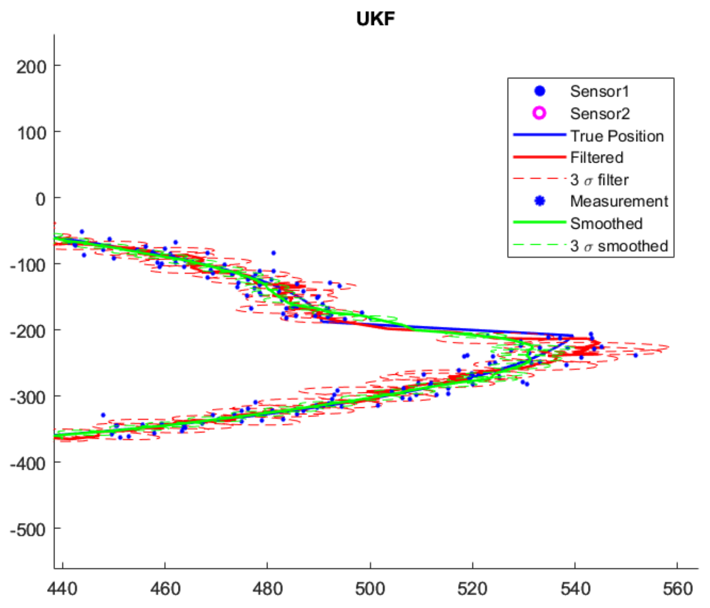

Figure 2: Task b: Manually outlier

From figure 2, the filtered output of the red curve follows the change of my manual outlier value, and has a big covariance at that point, while the smoothed output of the green curve goes less to the wrong value and has a smaller covariance compare to the filtered curve.

In summary, the filter is trying to incorporate the most recent measurement, including the outlier, into its estimate, while the smoother takes into account both past and future measurements when updating its estimate. As a result, the smoother can mitigate the effect of the outlier by considering the overall trajectory of the object and recognizing that the outlier is inconsistent with the other measurements.

2 Particle filters for linear/Gaussian systems

2.1 Task 2a

2.1.1 Part 1

`pfFilter` function offers xfp, which is the posterior means of particle filter , and Pfp, which is posterior error covariances of particle filter.

To calculate MSE, there is a function name as `mse` which is offered by Matlab.

The key codes are here:

```
    % figure;hold on;
    [xfpre, Pfpre, Xpre, Wpre] = pfFilter(x_0, P_0, Y, f, Q, h,
        R, ParticleNum, 1, myplot);
    % title('Resample')
    % figure;hold on;
    [xfp, Pfp, Xp, Wp] = pfFilter(x_0, P_0, Y, f, Q, h, R,
       ParticleNum, 0, myplot);
    % title('Noresample')

    %xfp is mean and Pfp is covariance
    disp('KF mse:')
    mse(X(2:end)-xf)
    disp('PF with resample')
    mse(X(2:end)-xfpre)
    disp('PF without resample')
    mse(X(2:end)-xfp)
```

I have tried a couple of numbers to see the trend of these MSEs:

Table 1: MSE and Number of Particles

	KF	PF	PF-resample
Np=20	1.3898	2.4238	3.6028
Np=30	1.4601	1.7728	5.6019
Np=40	2.0392	2.2256	7.1138
Np=50	1.7932	1.7836	4.8646

From table 1, the PF with resample can reach the precision of KF around the number of particles equal to 50.

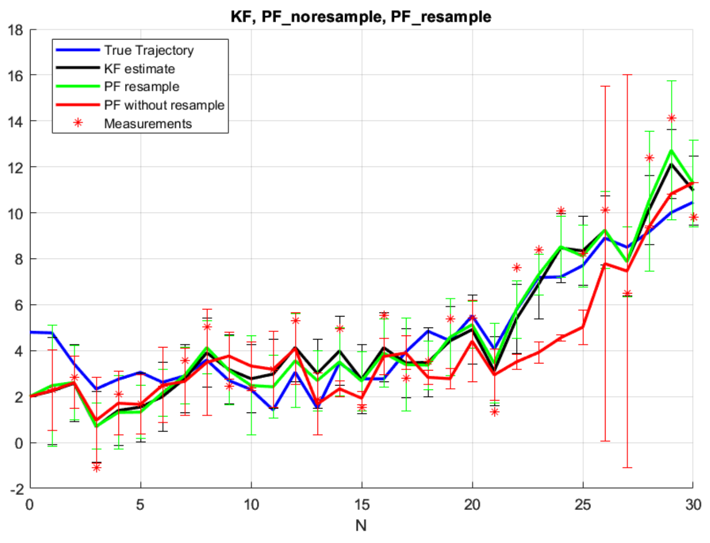

Figure 3: Filters and True trajectory

2.1.2 Part 2

I chose three values: $k = 1, 14, 28$

After some tuning, set $\sigma = 2.5$, and the pictures are:

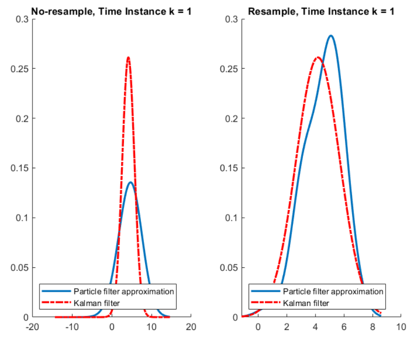

Figure 4: K=1

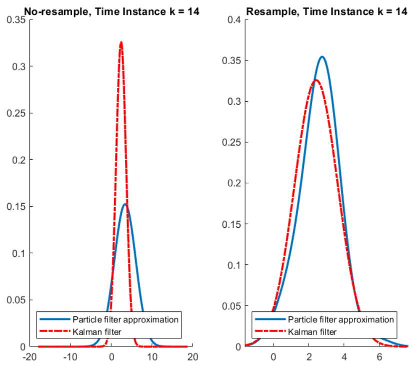

Figure 5: K=14

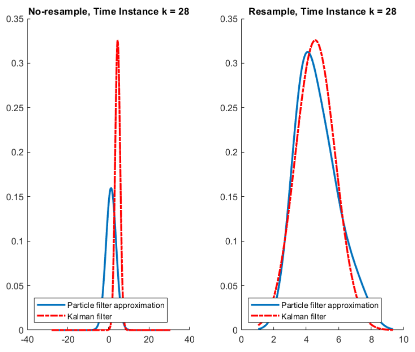

Figure 6: K=28

With the step going from 1 to 28, the resampled filter goes closer to the KF filter while the PF filter without resample goes further away from the KF one.

As the time step increases, the resampled PF updates its approximation by resampling particles based on their weights, which emphasizes the particles that better represent the true posterior. This process allows the resampled PF to adapt and refine its approximation over time.

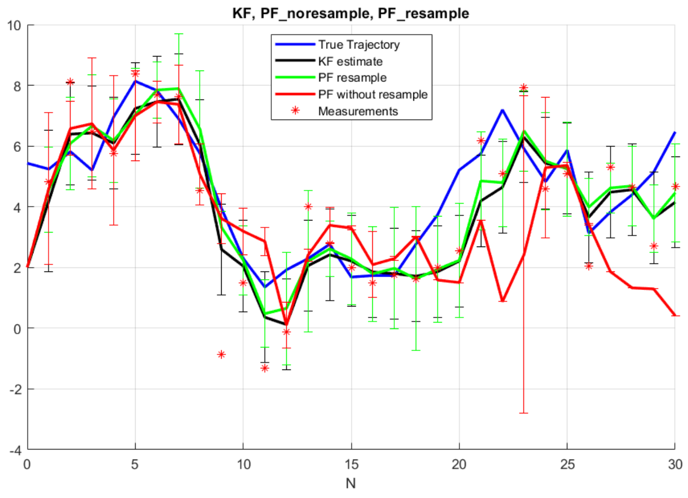

Figure 7: The filter figure for these three steps

However, the PF without resampling, on the other hand, does not update its particles based on their weights. Instead, it only performs the prediction and updates steps without resampling. This means that the particles are not adjusted to better represent the true posterior, potentially leading to a less accurate approximation. As the time step increases, the particles in the PF without resampling may deviate further from the true posterior, resulting in a larger discrepancy from the KF filter. Figure 7 proves this.

Overall, by selectively resampling particles based on their weights, the resampled PF can focus on regions of higher posterior probability and effectively track the true posterior distribution.

2.2 Task b

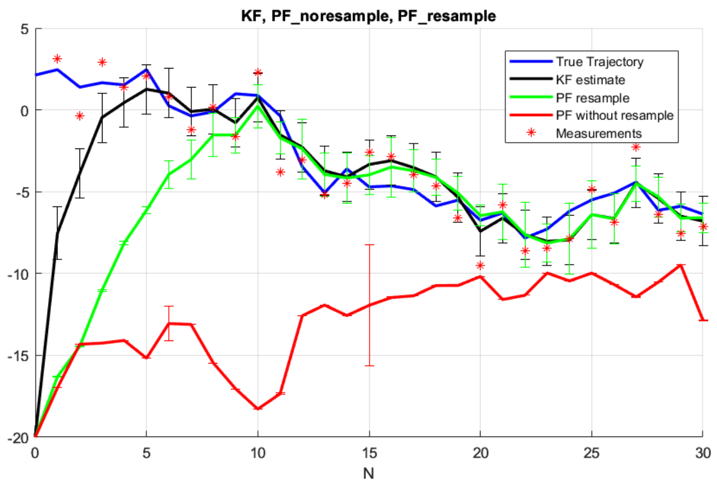

Figure 8: Filters with wrong prior

From figure 8, KF responds fastly and is the first one with these three filters that back to the correct position.

PF with resample is a little bit slower than the KF filter but it performs still well, and can finally converge to the real states curve.

Unfortunately, the PF without resample performs badly, even until the final step it can not back to an acceptable position.

The observation proves the analysis above in part 2, that compared with the other two, the PF without resampling lacks an adaptive mechanism, leading to a less accurate approximation.

2.3 c

In this part, need to change the plot function or cannot draw particle filter with more than 50 particles.

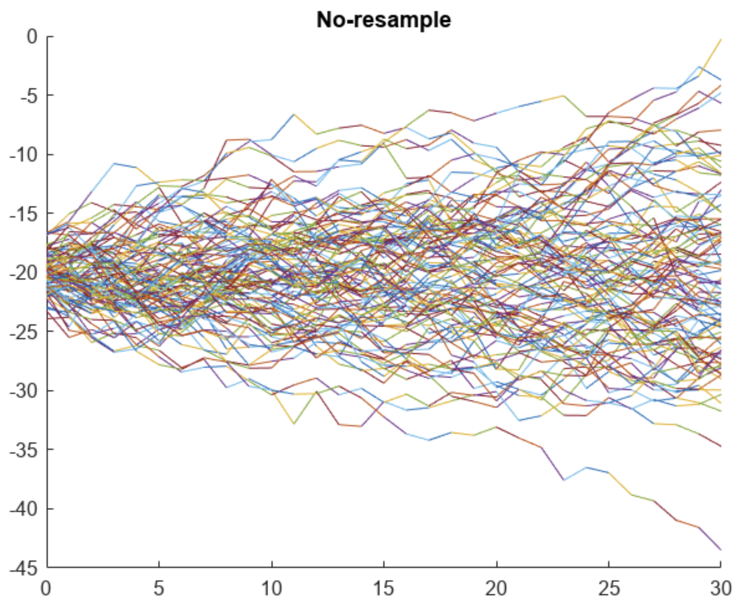

Figure 9: without resample

When illustrating the particle trajectories for a PF without resampling, I observe that as the iterations progress, the particles continue to diverge and spread out without any convergence trend. This lack of convergence can be attributed to the absence of resampling. Without resampling, the particles are not being weighted according to their likelihoods, leading to an unbalanced distribution that fails to capture the true underlying state accurately.

Consequently, the estimation performance of the filter is likely to deteriorate over time as the particles become increasingly scattered.

2.4 d

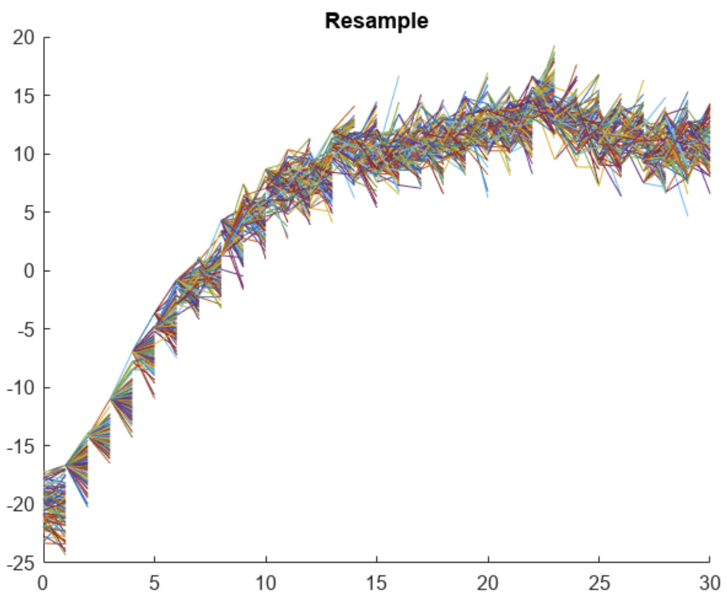

Figure 10: With resample

When illustrating the particle trajectories for a PF with resampling, initially, the particles are spread out, similar to the no-resampling case. However, as the iterations progress, the particles undergo resampling, which allows the more probable particles to be replicated and the less probable ones to be cut. This resampling process promotes the convergence of the particles toward the true underlying state. Consequently, I observe that the particle trajectories in the resampling case tend to cross and converge, indicating a better estimation performance compared to the no-resampling case.