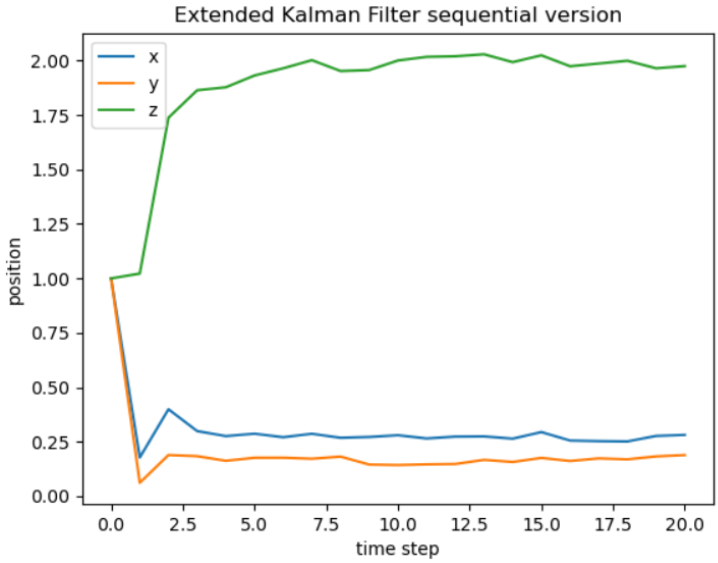
ROB530 Homework 3

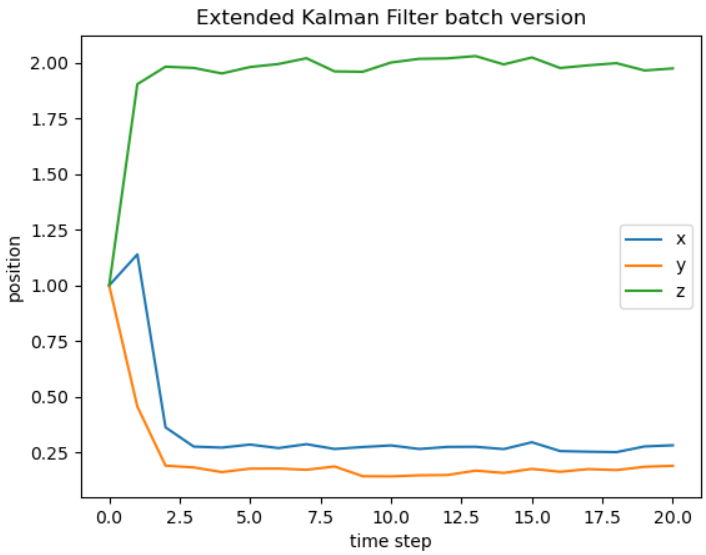
#50036744 Kuan-Ting Lee (leekt)

1. EKF
    a. See code
    b. See code
2. Particle Filter (PF)
    a. See code
    b. See code
3. Comparisons and Conclusion
    a. Estimated trajectory plot and final estimated object position:
        i. Sequential EKF



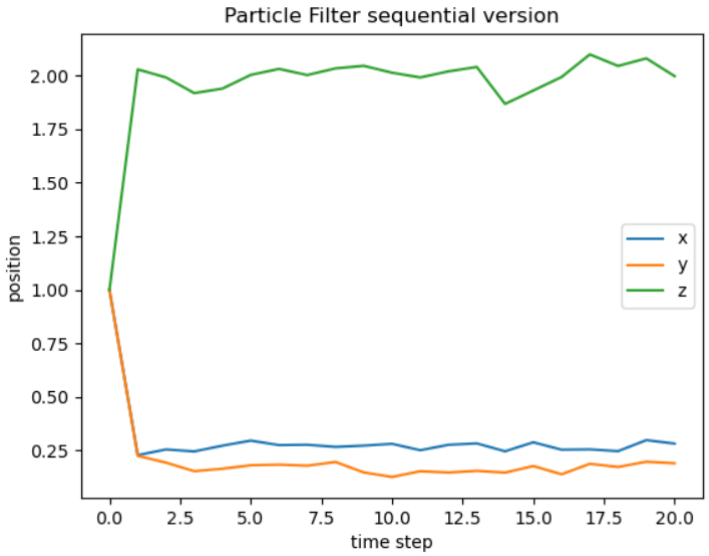
Estimated object position =
[[0.28177771]
[0.18915503]
[1.97464187]]

        ii. Batch EKF



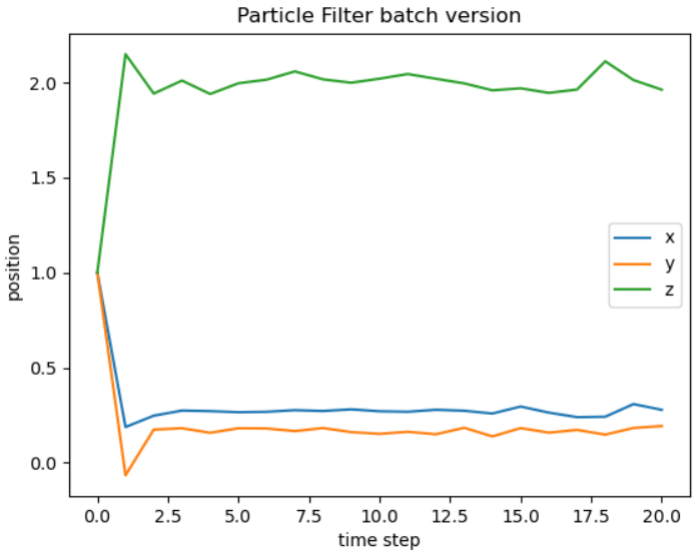
Estimated object position =
[[0.28164334]
[0.18938664]
[1.97500586]]

iii. Sequential PF



Estimated object position =
[[0.28279037]
[0.19099344]
[1.99754562]]

iv. Batch PF

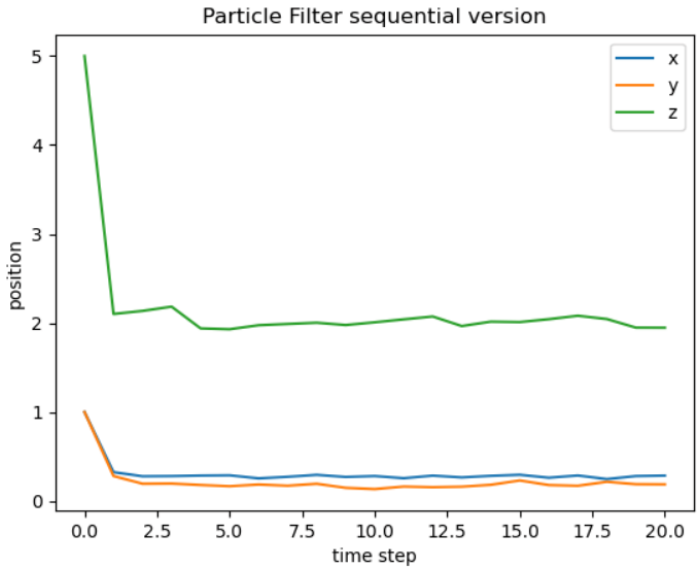


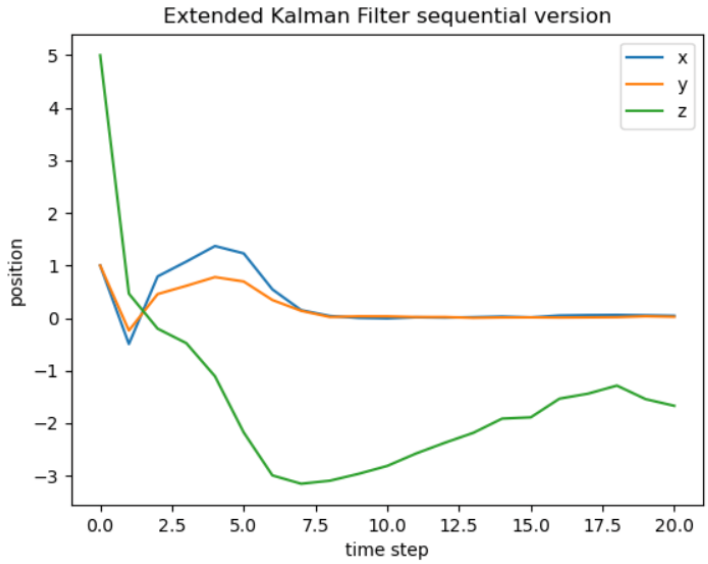
Estimated object position =
[[0.27813454]
[0.19255552]
[1.96271915]]

b. Filter recommendation:

To answer this question, I would like to compare two main factors: (1) run time, and (2) robustness. For each of the four filter, I could tune the sensor noise, motion noise, and initial guess of mean and covariance to make the state estimation converge to a certain state. However, this is not enough, run time is also important. For EKF batch version, the runtime is 0.16 second and for PF batch version, the runtime is 1.01 second. As we can see, the EKF is more efficient in terms of computation time. For robustness, PF performs better when the initial guess of the state is far from the convergent state. I ran an experiment to test this property. I set the initial guess of the state to be (1, 1, 5) to make the state far from the convergent state. As shown in the below graph, EKF's estimation couldn't converge while PF's estimation performed just as before.

Therefore, I will recommend EKF for customers that are searching for fast runtime, and recommend PF to someone that wants high robustness of the filter.


Particle Filter sequential version
x
y
z
position
time step
0
1
2
3
4
5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Extended Kalman Filter sequential version
x
y
z
position
time step
−3
−2
−1
0
1
2
3
4
5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0