ECE 8540

Lab 6 - Particle Filter

Tim Driscoll

November 16, 2021

1 Introduction

This lab report concerns the problem of applying a particle filter to a set of measurements. The measurements are numerical data that has been recorded over time in order to provide samples to filter. The measurements were recorded for a system that consists of a 1D position, moving on a line. The motion pattern of the system is an oscillation over the line. The sensor used in the system recoded a field strength that was the sum from two magnets at fixed positions. The particle filter was used to actively filter the measurements read in at each time step, allowing for more accurate state predictions. Unlike the previously used Kalman filters, the particle filter lets us model systems where there is non-Gaussian noise. The particle filter can filter data from a wider range of systems. It can handle systems where the noise is believed to be Gaussian or non-Gaussian and it can also handle systems that are linear or non-linear. Both the Kalman and extended Kalman filters are limited to the range of problems they are able to filter. The particle filter improves upon these methods to provide a solution that can be applied to a wider range of problems with proper knowledge of the system. Since the particle filter has been introduced it has widely been considered the best filtering method to accurately track system states.

2 Methods

The particle filter follows a cycle of prediction and update. In this cycle the next state is calculated for each particle. Then the measurement from the current time step is obtained. Using the new measurement the weight for each particle is updated. The weights are then normalized so that they all sum to one. A desired output is then computed. The desired output used in this lab was the expected value. Finally, there is a test to determine whether or not resampling is required. Resampling is performed if necessary and the process is repeated over the next time step.

When using the particle filter it is important to note that certain filter variables needed to be initialized. For this specific implementation of the lab those variables included the initial state and weight for each particle, the number of particles, the resampling threshold and a variety of noise constraints. The initial state for each particle was set to be 0. The initial weight for each particle was set to $1/M$ where M is the number of particles. The value of M was experimented with but fell in the range from 100 to 1000. The steps mentioned in the previous paragraph are further broken down below and the necessary equations are as follows:

In this problem there was two state variables, position and velocity.

$$X_t = \begin{bmatrix} x_t \\ s\dot{x}_t \end{bmatrix} \tag{1}$$

1. Calculate the next state for each particle m with the following state transition equation:

$$\{x_t^{(m)} = f(x_{t-1}^{(m)}, a_t^{(m)})\}_{m=1}^{M} \tag{2}$$

   Where $f(x_{t-1}^{(m)}, a_t^{(m)})$ is defined as the following:

$$f(x_t, a_t) = \begin{bmatrix} x_{t+1} = x_t + \dot{x}_t T \\ \dot{x}_{t+1} = \begin{cases} 2 & \text{if } x_t < -20 \\ \dot{x}_t + |a_t| & \text{if } -20 \leq x_t < 0 \\ \dot{x}_t - |a_t| & \text{if } 0 \leq x_t \leq 20 \\ -2 & \text{if } x_t > 20 \end{cases} \end{bmatrix} \tag{3}$$

   In equation 3 the input x_t represents the state variable vector and at represents the dynamic noise. The dynamic noise, a_t was drawn from a zero-mean Gaussian distribution $N(0, \sigma_a^2)$. Where σ_a was defined to be 2^{-4}.

2. Obtain the measurement of magnetic strength from the sensor.

$$Y_t = \begin{bmatrix} y_t \end{bmatrix} \tag{4}$$

3. Calculate the ideal measurement using the observation equation for the model.

$$g(x_t, n_t) = \left[y_t = \frac{1}{\sqrt{2\pi}\sigma_m} \exp\left(\frac{-(x_t - x_{m1})^2}{2\sigma_m^2}\right) + \frac{1}{\sqrt{2\pi}\sigma_m} \exp\left(\frac{-(x_t - x_{m2})^2}{2\sigma_m^2}\right) + n_t \right] \tag{5}$$

   In equation 5, $X_{m1} = -10$ and $X_{m2} = 10$ representing the locations of the magnets in the model. The value $\sigma_m = 4$. The input variable n_t represents the measurement noise and is pulled from the zero-mean Gaussian distribution $N(0, \sigma_n^2)$. Where σ_n was defined to be 2^{-8}. When solving for the ideal measurement of each particle it is assumed that $n_t = 0$ to represent no noise.

4. Use the ideal measurement calculated for each particle to compare against the actual measurement from the model of the measurement noise. This comparison can be done using the following equation:

$$p(y_t | x_t^{(m)}) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(\frac{-(y_t^{(m)} - y_t)^2}{2\sigma_n^2}\right) \tag{6}$$

5. Calculate the weight updates for each particle using the following equation:

$$\tilde{w}_t^{(m)} = w_{t-1}^{(m)} \cdot p(y_t | x_t^{(m)}) \tag{7}$$

6. The weights then needed to be normalized again so they all summed to one. This normalization was done using the following equation:

$$w_t^{(m)} = \frac{\tilde{w}_t^{(m)}}{\sum_{m=1}^{M} \tilde{w}_t^{(m)}} \tag{8}$$

7. Using the updated position state variable and weight the expected value was calculated and used as the filter output to estimate the position of the object.

$$E[x_t] \approx \sum_{m=1}^{M} x_t^{(m)} \cdot w_t^{(m)} \tag{9}$$

8. Finally, the algorithm needed to check if resampling should be performed. In order to determine if resampling should occur the coefficient of variation statistic and effective sample size needed to be calculated. The coefficient of variation statistic was calculated using the following equation:

$$CV = \frac{1}{M} \sum_{m=1}^{M} (M \cdot w^{(m)} - 1)^2 \tag{10}$$

   Then the effective sample size was calculated using the below equation.

$$\text{ESS} = \frac{M}{1 + \text{CV}} \tag{11}$$

   The threshold for resampling was determined by comparing the effective sample size against a percentage of the number of particles. This threshold was adjusted and determined how often the particle filter model would resample its particles. If resampling was required then the select with replacement method was used. This method functioned behind the idea of removing particles with bad weights and replacing them with particles that have large weights. Once the particles were replaced the weights would be reset to $1/M$ and the algorithm would continue over its next step.

9. Loop (t is incremented by Δt)

These steps conclude the algorithm that was used to create a particle filter for this lab and specific system. In this lab they were iterated over the number of total measurements which was 1109.

2.1 MATLAB Implementation

Once it was determined how the particle filter algorithm functioned steps 1-9 were implemented in software. The 2020a version of MATLAB on a Windows operating system was used to implement the code. All the initial matrices and constants were defined in the code and a for loop was used to iterate over the data for each time step. It was assumed that a new measurement would be read every second, thus ΔT was defined to be 1. The loop

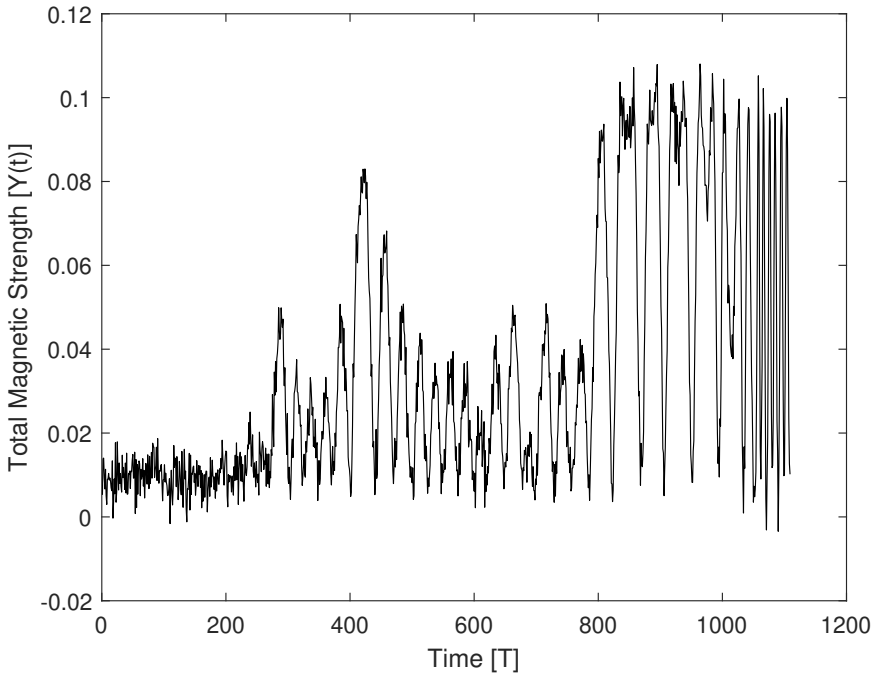

Figure 1: Raw sensor reading data.

was ran over the entire length of the measurement data. The estimated state was calculated and stored in a separate vector at the end of each loop. This saved data was then used to plot the results of the filtering. It is important to note that there were multiple nested for loops within the main loop iterating through the measurements. These nested for loops were required so that the defined calculations could be performed for each one of the particles.

3 Results

After testing the particle filter implementation on the proposed system some final design choices were decided on. I concluded that the filter produced accurate results that were not too computationally expensive when the number of particles used was 300. I also found effective results when resampling occurred if the effective sample size was less then 30 or 10% of the number of particles. Figure 1 shows a plot of the raw sensor readings used in this lab. The sensor recorded the total magnetic strength present on an object as it was passed by two magnets. As shown in the figure there is a large amount of noise in the sensor readings. The sensor does not produce a smooth increasing and decreasing curve as the object passes each magnet. Using the same raw sensor readings pictured in figure 1 the particle filter was able to accurately track the position of the object in the system. The estimated position produced by the filter was directly compared to and graphed against the actual position. Figure 2 shows the estimated state plotted in red over the true state plotted in black. As shown by the figure once the particles start to get good weights they

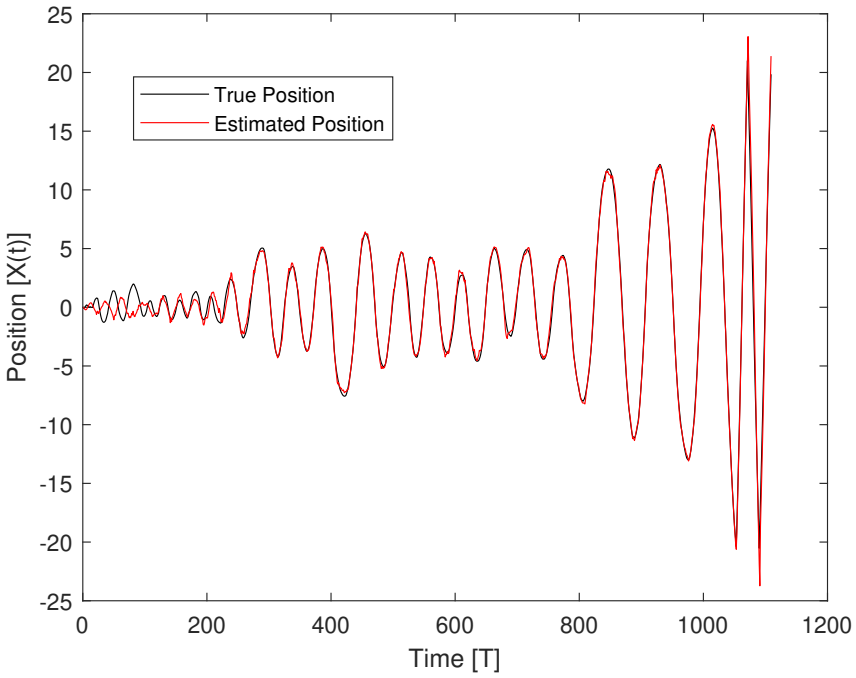

Figure 2: Estimated position plotted over true position.

are able to almost perfectly track the position of the object in the system. Although the filter was able to produce great results there was a parameter of the system that caused the particles to track a location 180 degrees out of phase. Since the problem was symmetrical in its noise distribution and magnet placement sometimes the estimated position would be out of phase with the actual position. Figure 3 shows an example of this tracking. This output would happen randomly depending on the random noise injected into the system. In both figures 2 and 3 the estimated position represents the mean of all the particles multiplied by their respective weights. Figure 4 shows a complete particle distribution at iteration 603 for one of my tests. This figure plots the weight of each particle versus the position of that particle. The red circle represents the true position of the object in the system, the weight is arbitrary. This particle distribution was captured when the system was set to run with only 50 particles. The distribution was also captured right before a resample was performed. It is evident that a resample needed to be performed since most of the particles had bad weights (almost zero). It is also important to note that the few particles with good weight were all close to the actual position of the object. Figure 5 represents the same distribution immediately after the resample was performed. The weight of each particle were reset to $1/M$ which was 0.02 in this case. The eliminated particles were all copied to the positions of particles that had higher weights prior to the resample. This is why there only appears to be about 7 different particles when there is still 50. The particles after the resample have a much smaller variance. The postion ranges from about -4.3 to -3.4. Prior to the resample the particles position ranged from about -7 to -1. All of the resampled particles have a position much closer to true position.

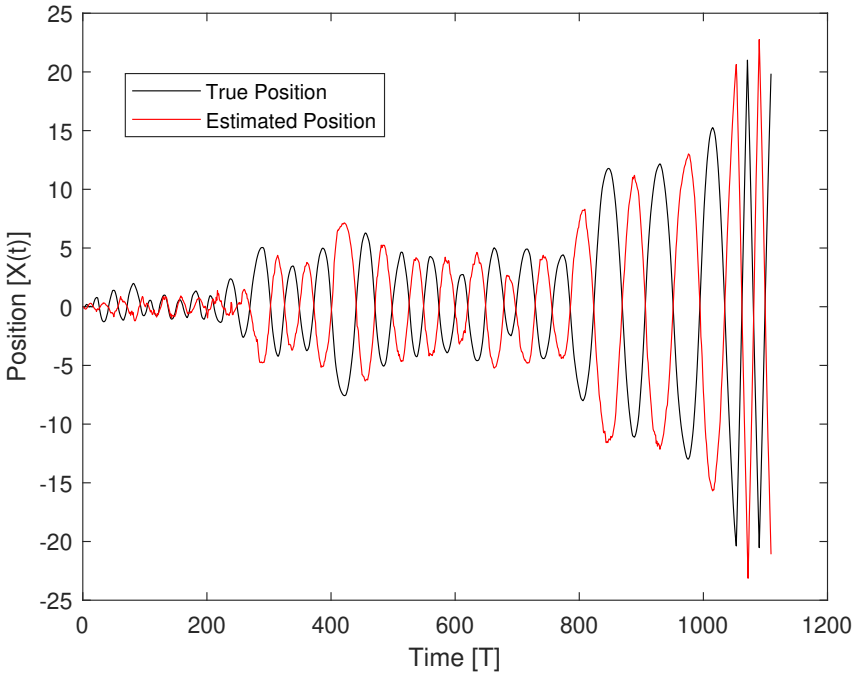

Figure 3: Estimated position plotted over true position when out of phase.

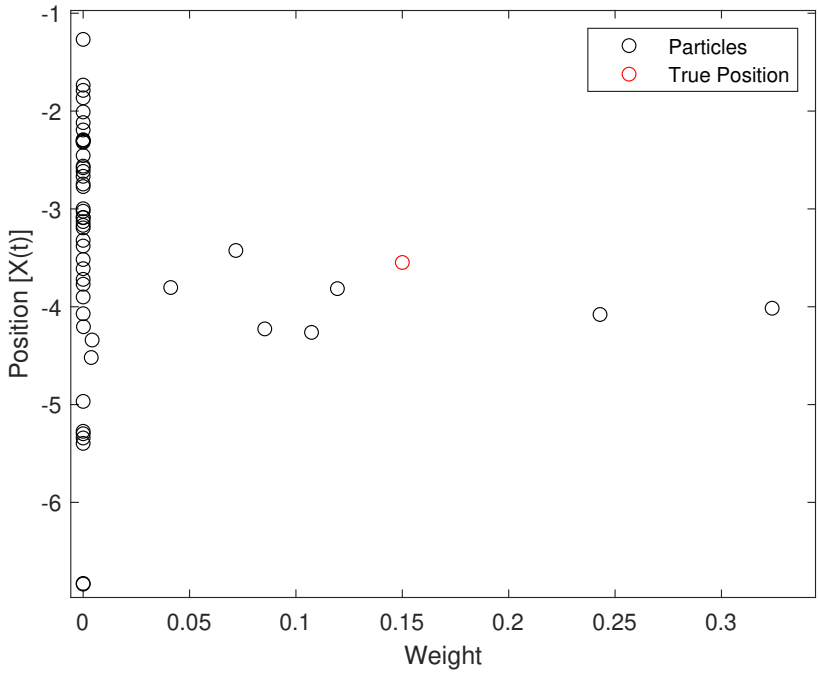

Figure 4: Particle distribution at iteration 603 prior to resample.

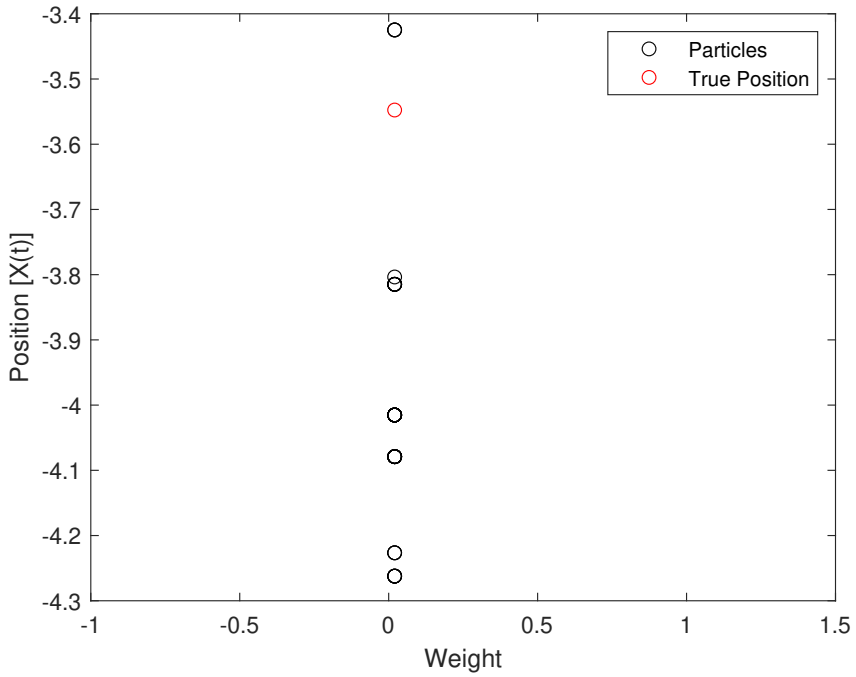

Figure 5: Particle distribution at iteration 603 after resample.

4 Conclusion

This lab showed the particle filters ability to accurately produce state estimates for complex systems with noisy measurement inputs. The particle filter showed a wider range of capabilities when compared to previous filters such as the Kalman and extended Kalman filter. The particle filter is able to handle linear and non-linear models like the Kalman and extended Kalman filter respectively. The particle filter excels in its ability to model non-Gaussian noise distribution in a system. For example the noise distribution modeled in the example was the addition of two Gaussian distributions. This allows for the particle filter to model almost any system if the proper model is known. Similar to other filters the largest restricting factor to the effectiveness of the particle filter is how well one can model the system being filtered. In this lab the particle filter was able to very accurately track the true position of the object as shown in 2. The largest error that occurred in the tracking of this system was sometimes the estimated positions would be 180 degrees out of phase. As discussed in the results section this happened due to the symmetrical nature of the problem. Figures 4 and 5 showed the importance of resampling in the particle filter. As the problem iterates an increasing number of particle will have bad weights that are almost zero. It is important to eliminate these particles and recenter everything around the positions of particles with higher weights. Overall, the particle filter has the ability to handle a wide range of systems and produced extremely accurate results with a properly tuned model.