Particle Filter

Lab-6

Submitted by:

Priyanshu Rawat

---

1 Introduction

The particle filter is a popular approach for solving estimation problems for non-linear and non-gaussian systems. The particle fitler approach combines the use of Bayesian estimation, Monte Carlo approximation, and sequential importance sampling.
The particle filter model consists of:

- X_t, a set of state variables
- A_t, the set of dynamic noises
- f(), the state transition equation
- Y_t, the set of measurements
- N_t, the set of measurement noises
- g(), the observation equation

The dynamic noise and measurement noise can be non-Gaussian, but they must be tractable.
The process of state estimation using particle filter includes:

1. Each particle m is propagated through the state transition equation:

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

   where $a_t^{(m)}$ represents the dynamic noise from $t-1$ to t, and is randomly and independently calculated for each particle m.

2. Using the new measurement vector y_t, the weight for each particle is updated:

$$\tilde{w}_t^{(m)} = w_{t-1}^{(m)} \cdot p(y_t | x_t^{(m)}) \tag{2}$$

3. Normalize the updated weights, so they sum to 1:

$$w_t^{(m)} = \frac{\tilde{w}_t^{(m)}}{\sum_{m=1}^{M} \tilde{w}_t^{(m)}} \tag{3}$$

4. Compute the desired output, such as the expected value (mean):

$$E[x_t] \approx \sum_{m=1}^{M} x_t^{(m)} \cdot w_t^{(m)} \tag{4}$$

5. Check if sampling is necessary, and if so, resample.

6. Let $t = t + 1$; iterate.

1.1 Problem Statement

To implement the particle filter using the following equations and the given measurements.
For this problem, there are two state variables:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} \tag{5}$$

The state transition equation is as follows:

$$f(x_t, a_t) = \begin{bmatrix} x_{t+1} = x_t + \dot{x}_t T \\ \dot{x}_{t+1} = \begin{cases} 2 & \text{if } x_t < -20 \\ \dot{x}_t + |a_t| & \text{if } -20 \leq x_t < 0 \\ \dot{x}_t - |a_t| & \text{if } 0 \leq x_t \leq 20 \\ -2 & \text{if } x_t > 20 \end{cases} \end{bmatrix} \tag{6}$$

Here, $\sigma_a = 2^{-4} = 0.0625$.
For observations, this problem uses a sensor that measures the totalmagnetic strength:

$$Y_t = \begin{bmatrix} y_t \end{bmatrix} \tag{7}$$

Two magnets are placed at $x_{m1} = -10$ and $x_{m2} = 10$. The value $\sigma_m = 4.0$ and the value of $\sigma_n = 2^{-8} = 0.003906$. The ideal measurement of the particle is calculated as follows:

$$g(x_t^{(m)}, 0) = \left[y_t^{(m)} = \frac{1}{\sqrt{2\pi}\sigma_m} \exp\left(\frac{-(x_t^{(m)} - x_{m1})^2}{2\sigma_m^2}\right) + \frac{1}{\sqrt{2\pi}\sigma_m} \exp\left(\frac{-(x_t^{(m)} - x_{m2})^2}{2\sigma_m^2}\right) \right] \tag{8}$$

The ideal measurement can then be compared against the actual measurement in the model of the measurement noise as follows:

$$p(y_t | x_t^{(m)}) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(\frac{-(y_t^{(m)} - y_t)^2}{2\sigma_n^2}\right) \tag{9}$$

Implement the particle filter using the above equations on the given data.

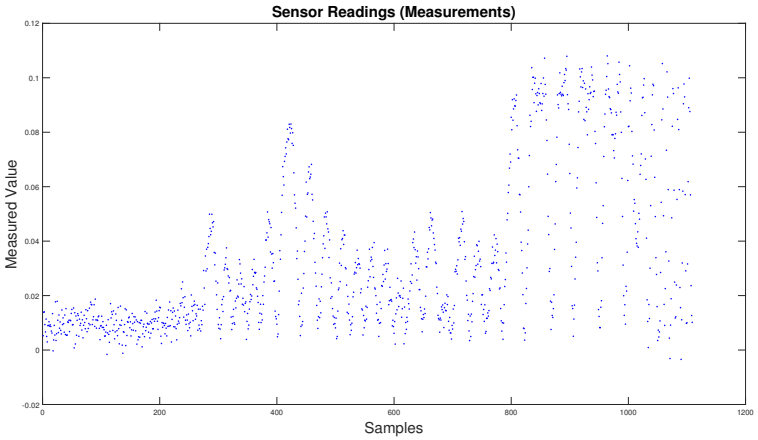

Figure 1:
Measured values of the field strength by the sensor

2 Results and discussion

2.1 Case I: without using resampling

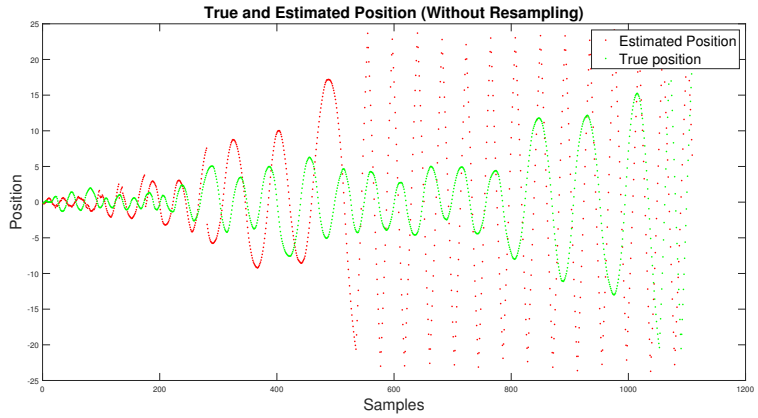

Figure 2:
Estimated position (when particles are not resampled)

The figure 2 above shows the estimated position from the particle filter when particles are not resampled.

2.2 Case II: when 50% of the particles are resampled

In this case, resampling of particles was done with a threshold set to 50% of the particles.

```
if (ESS < 0.5 M)
  resample
```

where, M is the total number of the particles. The method used for resampling is **select with replacement**. The idea is to eliminate particles having negligible weights, and replace them with the copies of the particles having large weights.

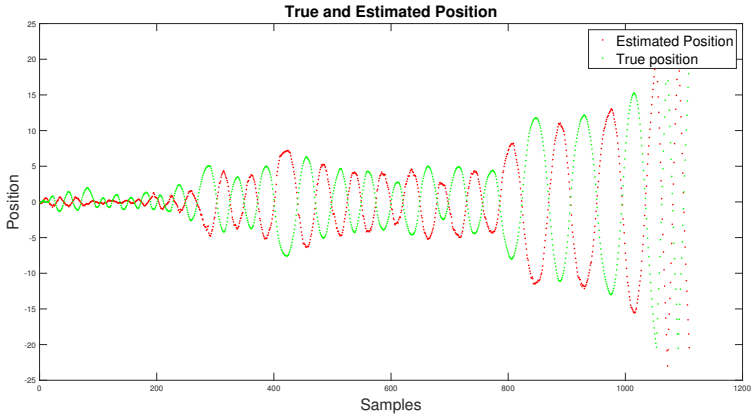

Figure 3:
Estimated position in the opposite direction (when 50% particles are resampled)

The figure 3 above shows the estimated position from the particle filter compared with the true position. In this figure, it can be seen that the estimated position is in the opposite direction than the actual direction of the system. The number particles in this case were taken to be 1000 and resampling threshold was set to 50% of the particles.

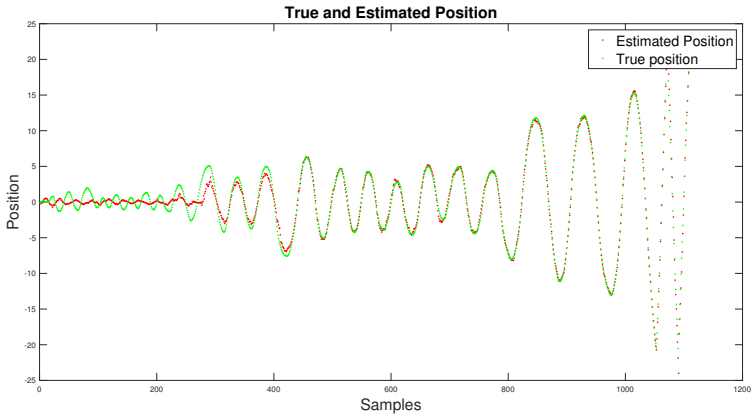

Figure 4:
Estimated position in the same direction (when 50% particles are resampled)

The figure 4 above shows the estimated position from the particle filter compared with the true position. In this figure, it can be seen that the estimated position is in the same direction as the actual direction of the system. The number of particles in this case again was taken to be 1000, and the resampling threshold was set to 50% of the particles.

2.3 Case III: when 90% of the particles are resampled

In this case, resampling of particles was done with a threshold set to 90% of the particles.

```
if (ESS < 0.9 M)
  resample
```

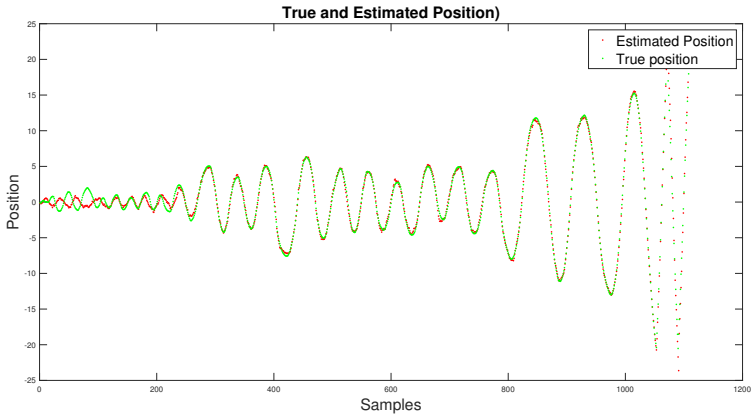

Figure 5:
Estimated position (when 90% particles are resampled)

The figure 5 above shows the estimated position from the particle filter compared with the true position. Here, 90% of the particles are resampled while the total number of the particles was the same 1000 as before. As seen, the estimated position tracks the true position much more accurately and earlier than when only 50% particles were resampled.

2.4 Case IV: when 10% of the particles are resampled

In this case, resampling of particles was done with a threshold set to 10% of the particles.

```
if (ESS < 0.1 M)
  resample
```

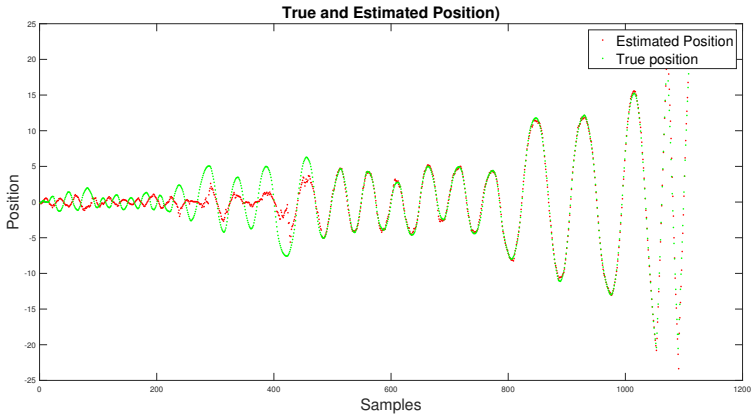

Figure 6:
Estimated position (when 10% particles are resampled)

The figure 6 above shows the estimated position from the particle filter compared with the true position. Here, 10% of the particles are resampled while the total number of the particles was the same 1000 as before. As seen, the estimated position tracks the true position less accurately than when 50% and 90% of the particles were resampled.

3 Conclusion

To conclude, as the number of particles were increased it was found that the accuracy of position estimates from the particle filter was improved. Also, with increasing percentage of the particles resampled the estiamted position was found to be more accurate in fewer iterations.
In summary, implementation of the particle filter can be summarized in 3 steps:

1. Generating a set of particles
2. Measure the probability of each particle being the actual position of the system
3. Resample based on the probability weight

4 Appendix (MATLAB script)

```
clc;
clear;

%% Loading the data in matlab
data = load("magnets-data.txt");
actual_position = data(:,1);
actual_velocity = data(:,2);
sensor_reading = data(:,3);

%% Plotting the measurements
figure
t = 1:1:length(data);
plot(t,sensor_reading,'.b')
title('Sensor Readings (Measurements)',FontSize=20)
ylabel('Measured Value',FontSize=20);
xlabel('Samples',FontSize=20);

%% Given Constants
sigma_a = 0.0625;
sigma_m = 4;
sigma_n = 0.003906;
x_m1 = -10;
x_m2 = 10;

%% Initial states
x=sensor_reading(1);
xdot=0;

%% Initalizing particles
M = 1000;
w(1:M,1) = 1/M;

%% initial states of the particles
xP(1:M,1) = x;
xdotP(1:M,1) = xdot;

%% Particle Filter Loop

i = 0; j = 0;
for i = 2:1:length(data)

    %% Actual Measurement
    yt = sensor_reading(i);
```

```
for j = 1:1:M

    %% Propagation through state transition eqns
    xP(j,2) = xP(j,1) + xdotP(j,1)*1;
    if xP(j,1) < -20
        xdotP(j,2) = 2;
    elseif xP(j,1) >= -20 && xP(j,1) < 0
        xdotP(j,2) = xdotP(j,1) + abs(normrnd(0,sigma_a));
    elseif xP(j,1) >= 0 && xP(j,1) <= 20
        xdotP(j,2) = xdotP(j,1) - abs(normrnd(0,sigma_a));
    elseif xP(j,1) > 20
        xdotP(j,2) = -2;
    end

    %% Calculting weights
    y(j,2) = (1/(sigma_m*sqrt(2*pi)))*(exp(-((xP(j,2)-x_m1)^2)/(2*(sigma_m^2)))) + (1/(s
    p(j,2) = (1/(sigma_n*sqrt(2*pi)))*(exp(-((y(j,2)-yt)^2)/(2*(sigma_n^2))));
    w_tilda(j) = w(j,1)*p(j,2);
end

%% Normalizing weights
 w(:,2) = w_tilda./sum(w_tilda);

%% Expected Value
exP = 0;
exP = xP(:,2).*w(:,2);

Ex(i,1) = sum(exP);

%%  Resampling
for j = 1:M
    v(j) = ((M*w(j,2))-1)^2;
end
cv = (1/M)*sum(v);
ESS = M/(1+cv);

if ESS < 0.5*M
    fprintf('Resampling %d \n', i);
    Q=cumsum(w(:,2));
    t=rand(M+1);
    T=sort(t);
    T(M+1)=1.0;
    l=1; k=1;
    while (k<=M)
        if (T(k) < Q(l))
            Index(k)=l;
```

```
                k=k+1;
            else
                l=l+1;
            end
        end

        for o=1:1:M
            NewP(o,1)=xP(Index(o));
            NewV(o,1)=xdotP(Index(o));
            NewW(o,1)=1/M;
        end
        xP(:,2) = NewP(:,1);
        xdotP(:,2) = NewV(:,1);
        w(:,2) = NewW(:,1);
    end
    xP(:,1) = xP(:,2);
    xdotP(:,1) = xdotP(:,2);
    w(:,1) = w(:,2);
end

%% Plotting the estimated position values
figure
t = 1:1:length(data);
scatter(t,Ex,'.')
hold on
scatter(t,actual_position,'.');
```