.

1 Introduction

The objective of Lab 4 is to implement a Particle Filter. Data provided is for a system which consists of a 1D position, moving on a line at varying velocities. The position zig-zags back and forth across 0 but between -20 and 20. The data is provided in the file "magnets-data.txt" and consists of actual position, actual velocity, and sensor reading on each column for a time t.

2 Methods

2.1 Initialization

Before designing the particle filter, a set of variables have to be initialized. These values are:

1. Time Period T, which is the time period between each sample. T is set at 1s.
2. Number of Particles M, which is typically set between 10 and 1000.
3. Two magnets are placed a x_{m1} = -10 and x_{m2} = 10.
4. The estimated positon, velocity and the sensor readings of the particle filter at time t=1 is set to 0. The probability distribution function $p(y_t \mid x_t)$ is also initialized to zero.
5. The weights of the Particle filter for time t=1 is initialized to $1/M$

2.2 Estimation of State Variables and Output

After initialization , the following steps are executed:

1. Each particle m is propagated throught the state transistion equations. The state transition equation for the system being tracked is given below:

$$x_{t+1} = x_t + \dot{x}_{t+1} \tag{1}$$

$$\dot{x}_{t+1} = \begin{cases} 2 & \text{if } x_t < -20 \\ \dot{x}_t + |a_t| & \text{if } -20 \leq x_t < 0 \\ \dot{x}_t - |a_t| & \text{if } 0 \leq x_t \geq 20 \\ -2 & \text{if } x_t > 20 \end{cases} \tag{2}$$

The value of a_t is is a random amount derived from the Gaussian distribution $N(0,{\sigma_a}^2)$.

2. The new measurement vector y_t is determined using the equation:

$$y_t^{(m)} = \frac{1}{\sqrt{2\pi}\sigma_m^2} exp(\frac{-(x_t^{(m)} - x_{m1})^2}{2\sigma_m^2}) + \frac{1}{\sqrt{2\pi}\sigma_m^2} exp(\frac{-(x_t^{(m)} - x_{m2})^2}{2\sigma_m^2}) \tag{3}$$

3. The probability distribution function for the particle m, $p(y_t \mid x_t)$ is given by,

$$p(y_t|x_t^{(m)}) = \frac{1}{\sqrt{2\pi}\sigma_n^2} exp(\frac{-(y_t^{(m)} - y_t)^2}{2\sigma_n^2}) \tag{4}$$

   The value of σ_n is set to 0.003906.

4. The weight of the particle is then updated as follows,

$$\widetilde{w}_t^{(m)} = w_{t-1}^{(m)}.p(y_t|x_t^{(m)}) \tag{5}$$

5. The updated weights are then normalized such that their sum is 1. This is done as shown below:

$$w_t^{(m)} = \frac{\widetilde{w}_t^{(m)}}{\sum_{m=1}^{M} \widetilde{w}_t^{(m)}} \tag{6}$$

6. The values are then checked to see if resampling is necessary or not. For this, the Effective Sample Size (ESS) is calculated.

$$ESS = \frac{M}{1 + CV} \tag{7}$$

   The coefficient of variation (CV) is determined by the following formula.

$$CV = \frac{1}{M} \sum_{m=1}^{M} (M w_t^{(m)} - 1)^2 \tag{8}$$

7. If the ESS is less than 0.5 times M, then resampling is done by select with replacement and the state variables and the weights are updated accordingly.

8. The value for t is then incremented and the above steps are repeated for each t.

3 Results

The Particle Filter was first executed by setting M=1000. The graph, Fig 1. given below shows the Particle filter output (resampled) and the true position of the system. From the graph, it can be seen that PF output is coming close to following the true position of the system toward the end.

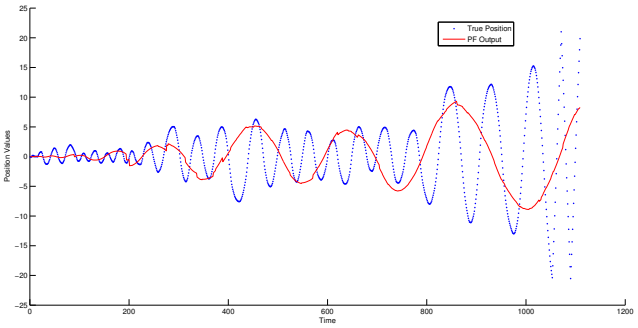

Figure 1: Plot of PF Output(Resampled) and true position (M=1000)

The PF filter was then executed for M=10 and the resulting graph is shown in Fig 2. With less number of particles, the filter struggles to track the position within the given timeframe.

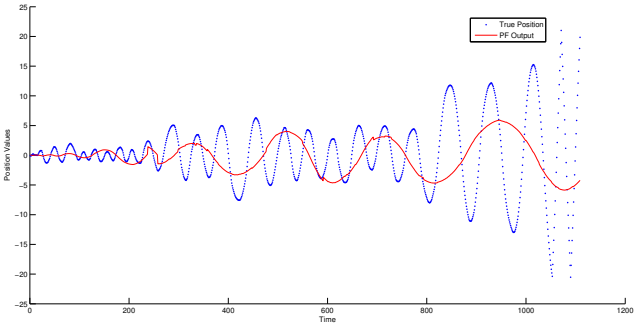

Figure 2: Plot of PF Output(Resampled) and true position (M=10)

The graph , Fig 3., shows a comaprison of the PF output for both values of M.

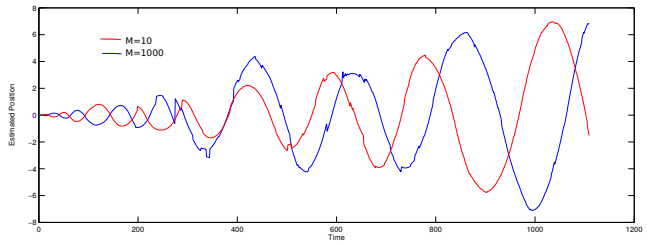

Figure 3: Comparison of PF output for M=10 (red) and M=1000 (blue)

The output of the filter without resampling at M=1000 is shown in Fig 4.

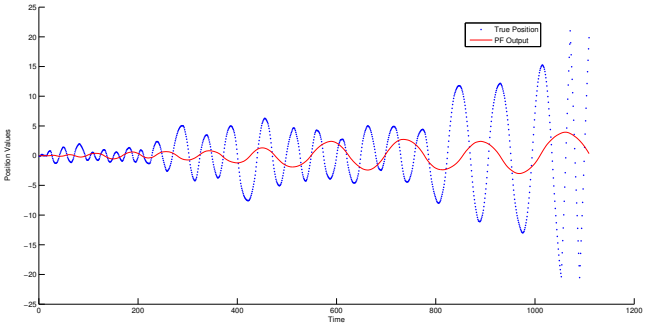

Figure 4: Plot of PF Output (Unsampled) and true position (M=1000)

The comparison of the resampled and unsampled outputs are given in Fig 5.

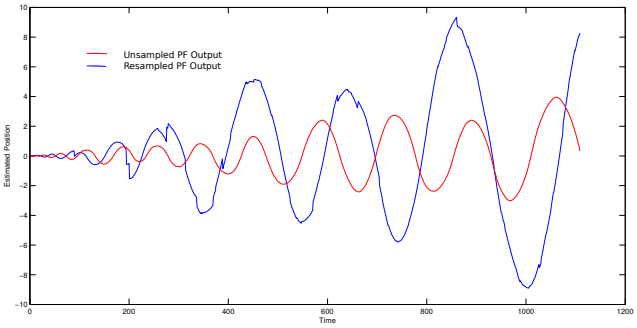

Figure 5: Plot of PF Output (Sampled) and PF Output (Unsampled) (M=1000)

The graph show below shows the particle distribution at time t=100.

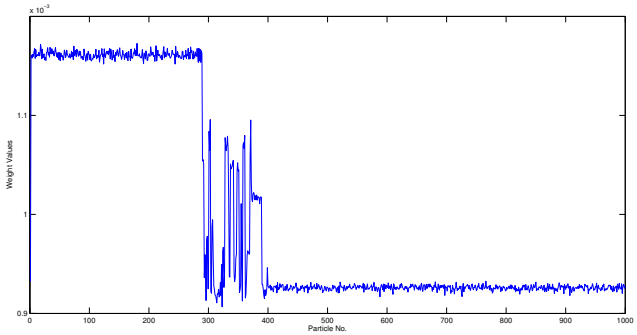

Figure 6: Particle Distribution at t=100

4 Conclusions

Thus, a particle filter for the given system has been developed. The filter does not track well during the given time samples but seems to get close to tracking it towards the end of the data

5 Appendix : Matlab Code

```
clear;
data=load('magnets-data.txt');

%given values
sig_a=0.0625;
xm1=-10;
xm2=10;
sig_n=0.003906;
sig_m=4.0;

%Initialization
T=1;                              %sampling rate
N=10;                             %No. of particles
Yt=data(:,3);
for i=1:N
    x(1,i)=0;                      %Position Variable
    x_dot(1,i)=0;                  %Velocity variable
    y(1,i)=0;                    %output..sensor reading
    pdf(1,i)=0;                    %probability distribution function
    w(1,i)=1/N;                    %initial weights
end

%state transition equation
for t=2:length(data(:,1))
    Exp=0;CV=0;
for i=1:N
   x(t,i)=x(t-1,i)+x_dot(t-1,i)*T;
   a_t=normrnd(0,(sig_a*sig_a));
   if (x(t-1,i)<-20)
        x_dot(t,i)=2;
   elseif (-20<=x(t-1,i) && x(t-1,i)<0)
        x_dot(t,i)=x_dot(t-1,i)+abs(a_t);
   elseif (0<=x(t-1,i) && x(t-1,i)<=20)
        x_dot(t,i)=x_dot(t-1,i)-abs(a_t);
   elseif (x(t-1,i)>20)
        x_dot(t,i)=-2;
   end
   n_t=normrnd(0,sig_n^2);
    %y(t,i)=normpdf(x(t,i),xm1,sig_m)+normpdf(x(t,i),xm2,sig_m);
    y(t,i)=normpdf(x(t,i),xm1,sig_m)+normpdf(x(t,i),xm2,sig_m);
    im(t,i)=normpdf(x(t,i),xm1,sig_m)+normpdf(x(t,i),xm2,sig_m);
    pdf(t,i)=normpdf(y(t,i),data(t,3),sig_n);
    %pdf(t-1,i)=normpdf(im(t-1,i),y(t-1,i),sig_n);
    w(t,i)=w(t-1,i)*pdf(t,i);
```

```
end
for i=1:N
    wn(t,i)=w(t,i)/sum(w(t,:));
    Exp=Exp+x(t-1,i)*wn(t,i);
end
    w=wn;
    E(t)=Exp;
    CV=sum((N*w(t,:)-1).^2)/N;
    ESS=N/(1+CV);
if(ESS<0.5*N)
    Q=cumsum(w(t,:));
    t1=rand(N+1);
    t2=sort(t1);
    t2(N+1)=1;
    j=1;k=1;
    while (j<=N)
        if (t2(j)<Q(k))
            index(j)=k;
            j=j+1;
        else
            k=k+1;
        end
    end

for i=1:N
    x(t,i)=x(t,index(i));
   x_dot(t,i)=x_dot(t,index(i));
    w(t,i)=1/N;
end
end
end
```