EE422/CS421

INTRODUCTION TO ROBOTICS

HOMEWORK

Due Date: May 30th, 2019

Name **Number** **Signature**

Consider the following non-linear state-space equations.

$$x_t = \frac{x_{t-1}}{2} + \frac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t$$

$$y_t = \frac{x_t^2}{20} + n_t$$

where v_t and n_t are zero mean Gaussian random variables with variances $\sigma_v^2 = 10$ and $\sigma_n^2 = 1$, respectively.

Using this information, generate 1000 samples of x and y.

By using the following filters, find the minimum mean squared estimate (MMSE) of the state, which is shown by $\hat{x}_t$. For each filter, plot the original state, x_t, and its estimate $\hat{x}_t$. Compare your results that are obtained by 3 filters and elaborate on this. Which one performs better and why?

a) Bootstrap Particle Filter (For resampling, you can either use systematic resampling (given below by pseudocode) or Residual Resampling (code is provided) or any other method you prefer)

b) Extended Kalman Filter

c) Kalman Filter

You can use either MATLAB or Python for coding. You can also use commands from the Statistical Sensor Fusion Toolbox whose manual has been provided before.

You should create a GitHub repository and place your codes in it. The link should be provided and the repository should be active for at least 20 days following the submission date of this homework.

Systematic Resampling pseudocode:

==========

$(j) = SR(N, i)$

Generate random number $u \sim \mathrm{U}\left(0, \frac{1}{N}\right)$

s = 0

for $i = 1 : N$

    k = 0

    s = s + $\tilde{w}_t^{(i)}$

    while $s > u$

        k = k + 1

        $u = u + 1/N$

    end

$j(i) = k$

end

==========

Residual Resampling:

```
function [theta]=Res_Res(theta,wt)
N=length(theta);
theta_temp = theta;
wn_res=(N.*wt);
N_sons=fix(wn_res);
N_res=N-sum(N_sons);
if (N_res~=0)
```

```
   wn_res=(wn_res-N_sons)/N_res;
   dist=cumsum(wn_res);
   u=fliplr(cumprod(rand(1,N_res).^(1./(N_res:-1:1))));
   j=1;
   for i=1:N_res
      while(u(1,i)>dist(1,j))
         j=j+1;
      end
      N_sons(1,j)=N_sons(1,j)+1;
   end
end
ind=1;
for i=1:N;
   if(N_sons(1,i)>0)
    tp1=theta_temp(i);
    for j=ind:ind+N_sons(1,i)-1
       theta(j)=tp1;
    end
 end
 ind=ind+N_sons(1,i);
end
```