Assignment 2: Particle Filters

I expect a small paper (or pdf) report from you on this assignment that consists of:

1) An answer to all questions, item by item, including plots and your discussions.

2) Your Matlab program. No need to streamline it, just to see you haven't made serious errors.

3) Nothing more!

Standard Particle Filter

To get some hands-on experience with Particle Filters you have to write a Matlab program to study its behaviour. The model is a simple 1D model given by:

$$x^{n+1} = x^n + 20\frac{x^n * x^n}{1 + \exp(x^n - 7)} + \beta^n \tag{1}$$

in which n is the time index, and β^n is a random number drawn from $N(0, 0.3)$.

1) Generate a Matlab code that runs this model for 2000 time steps to initialise the model. Plot the evolution of x with time.

2) Generate a truth run starting from the last time step running for 200 time steps. Measure this truth run every 5 time steps to generate the observations. Add random noise to each of these observations with observation errors drawn from $N(0, 0.01)$. Plot this run and the observations with their error bars (see code given below). Are the observations spread around the truth as expected?

3) Write a code that does stochastic universal sampling, to be used as the resampling scheme in the particle filter.

4) Generate the particle filter code that runs 20 particles forward in time to the first observation time, weight each particle, resample, and continue to run the ensemble to the next observation, all the way to the end of the time window at 200 time steps. (Hint: it is possible that the weights get too low before normalisation to be represented by Matlab. In that case calculate $-\log w_i$ for each particle, subtract the maximal value of all particles, and only than take the exp.)

5) Plot the squared difference between the truth and the mean of the particle filter ensemble as function of time. Also calculate the time-averaged squared difference between the two.

6) Plot in one figure the truth, the observations with their error bars, and the ensemble of particles against time. Describe this plot. Do the same for as run of 20 time steps to be able to concentrate on the details. Explain this plot with the values for the weights

that you obtained. Is the filter degenerate (calculate e.g. the number of particles with weight (before resampling) larger than $1/N$, in which N is the number of particles)? You can use a plotting code similar to that given below

```
xxplot=zeros(T,1);
e = zeros(NM-1,1);
e(:) = sobs;
figure(30)
xxplot(:)=hpf(1,:);
plot([1:T],xxplot,'g-');hold on
for i=1:MM
xxplot(:)=hpf(i,:);
plot([1:T],xxplot,'g-');
end
errorbar([1:NM-1]*ns,y(1:NM-1), e,'rx');
plot([1:T],xtrue(:),'k-');
hold off
drawnow
```

in which T is the total number of time steps an NM is the total number of observations, hpf contains the ensemble members, xtrue holds the truth, y contains the observations, and e contains the standard deviations of the observations.

Particle filter with proposal density

To improve the performance of the particle filter we will explore the idea of a proposal density. To this end the model equation is changed to:

$$x^{n+1} = x^n + 20\frac{x^n * x^n}{1 + \exp(x^n - 7)} + K(t)(y^m - x^n) + \beta^n \tag{2}$$

where $K(t)$ varies linearly from 0 to 4/5 between observations. Note that the first observation in the future is used, denoted here as that one at time m.

1) Copy and modify the code generated above to include the term $K(t)(y^m - x^n)$.

2) Since we added the relaxation term to the model equation we have to change the weights to compensate for this, so calculate $p(x^{n+1}|x^n)/q(x^{n+1}|x^n, y^m)$ for each time step between observations, multiply these together, and finally multiply the result with the likelihood. Normalise to get the final weight of each particle. Don't forget the resampling step.

3) Generate similar plots as for the standard particle filter.

4) Compare the time-averaged root-mean-square difference between ensemble mean and truth of this version of the particle filter with the standard particle filter. Is this version less degenerate?

5) Can you think of ways to solve the degeneracy problem even further?