Georgia Institute of Technology

ECE 6555 - Bonus Assignment 2

due before Friday 16, 2022 - v1.0

- There is 1 problem over 5 pages (including the cover page).
- Each question is graded as follows: no credit without meaningful work, half credit for partial work, full credit if essentially correct.
- Unless otherwise specified, you should concisely indicate your reasoning and show all relevant work.
- The grade on each question is based on our judgment of your level of understanding as reflected by what you have written. If we cannot read it, we cannot grade it.
- Please use a pen and not a pencil if you handwrite your solution.
- **You must submit your assignment on Gradescope**.

Problem 1: Synthetic example of particle filter

Consider the following scalar non-linear state-space model.

$$x_k = \frac{1}{2}x_k + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2(k-1)) + u_k \tag{1}$$

$$y_k = \frac{1}{10}x_k^2 + v_k \tag{2}$$

where $\{u_k\}$ and $\{v_k\}$ are zero-mean white Gaussian noise sequences with unit variance. Assume that the initial state is $x_0 = 0.1$, and the initial estimate is $\hat{x}_0 = 0$ with variance $\sigma_0^2 = 2$. Note that both the state and the measurement processes are non linear. In particular, the measurement loses all the information about the sign of the state. The goal is to compare the performance of a particle filter and an EKF.

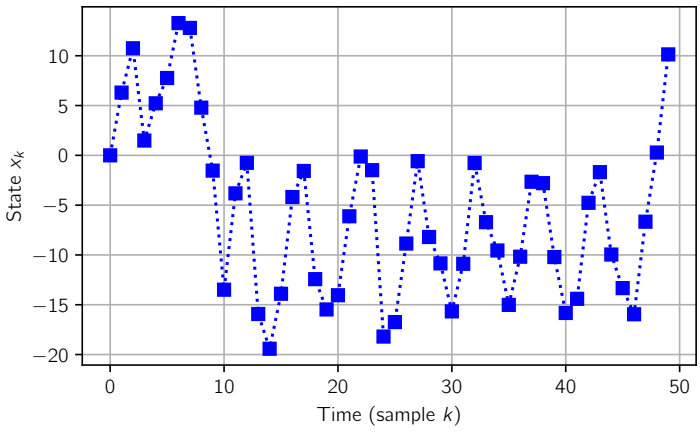
We will track the system over 50 times steps as given in the Python code. Make sure to reuse the same code with the same seed.

```
np.random.seed(202212)
NumSteps = 50
TimeScale = np.arange(1,NumSteps,1)
x0=0
sigma=1

x = [x0]
y = [0]
for k in TimeScale:
  xk = 0.5*x[-1]+25*x[-1]/(1+x[-1]**2)+8*np.cos(1.2*(k-1))+np.random.randn()
  yk = 1/20*xk**2+np.random.randn()
  x.append(xk)
  y.append(yk)
```

[Q1] Make a plot of the trajectory. This will serve as a reference throughout this problem.

Below is an example of what the trajectory should look like.



We start by studying the particle filter associated to the model. As described in class, the particle filter can be described in terms of the following steps.

1. Draw n samples from the prior $x_0^{(i)} \sim p(x_0) \quad i = 1 \dots n$ and set $w^{(i)} = \frac{1}{n}$ for $i = 1 \dots n$
2. For each $k = 1 \dots T$
   (a) Draw samples $x_k^{(i)}$ from importance distributions

$$x_k^{(i)} \sim \pi(x_k | x_{0:k-1}^{(i)} y_{1:k}) \quad i = 1 \dots n$$

   (b) Compute new weights

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(y_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)})}{\pi(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})}$$

   and normalize
3. If the effective number of particles n_{eff} get too low, resample and reset to uniform weights, where

$$n_{\text{eff}} \approx \frac{1}{\sum_{j=1}^{n} (w_k^{(i)})^2}$$

However, we did not specify *how* to choose the importance distribution and how to perform the sampling. In what follows, we choose

$$\pi(x_k | x_{0:k-1}^{(i)} y_{1:k}) \triangleq p(x_k | x_{k-1}^{(i)}), \tag{3}$$

i.e., we only push the current particle $x_{k-1}^{(i)}$ through a process update.

[Q2] Given a set of particles $\{x_{k-1}^{(i)}\}$, show that sampling from the importance distribution $\pi(x_k | x_{0:k-1}^{(i)} y_{1:k})$ then reduces to computing

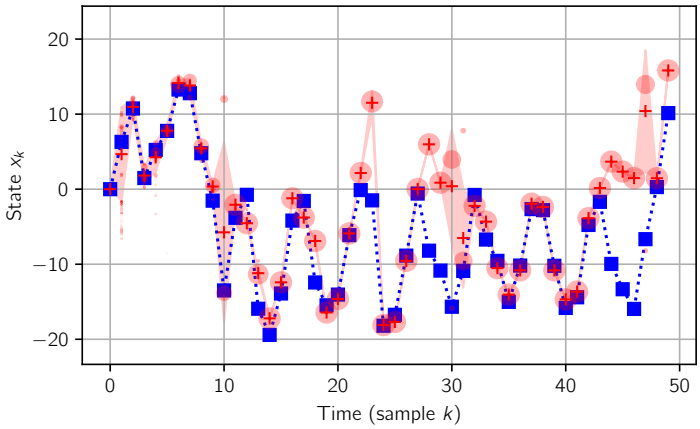
$$\frac{1}{2} x_{k-1}^{(i)} + \frac{25 x_{k-1}^{(i)}}{1 + (x_{k-1}^{(i)})^2} + 8 \cos(1.2(k-1)) \tag{4}$$

and adding the realization of zero mean white Gaussian noise.

[Q3] Provide an explicit expression for your computation of the weights $w_k^{(i)}$ as a function of $w_{k-1}^{(i)}$. Make sure you make sure of the subsequent normalization to avoid unnecessary computations.

[Q4] Implement the particle filter *without* the resampling step. Use 200 particles. Provide a graph illustrating the behavior of the algorithm, showing at least the mean and variance of your filter superposed to the generated data. Also plot the effective number of particles as a function of the time index k.

As a suggestion, here is a graph I made.



The plot shows at each time step the true state (■). At each time step, the particles with large enough weight are drawn as red circles (○) whose width is proportional to the weight of the particle. The 2σ standard deviation is also plotted as a red band around the estimate.

You should realize that the number of effective particles gets very small quickly. To make up for that, we need to account for the resampling step. Once the weights have been calculated, note that we are effectively approximating the posterior distribution $p(x_k|y_{1:k})$ as the discrete distribution

$$p(x|y_{1:k}) \approx \sum_{i=1}^{n} w_k^{(i)} \delta[x - z_k^{(i)}].$$

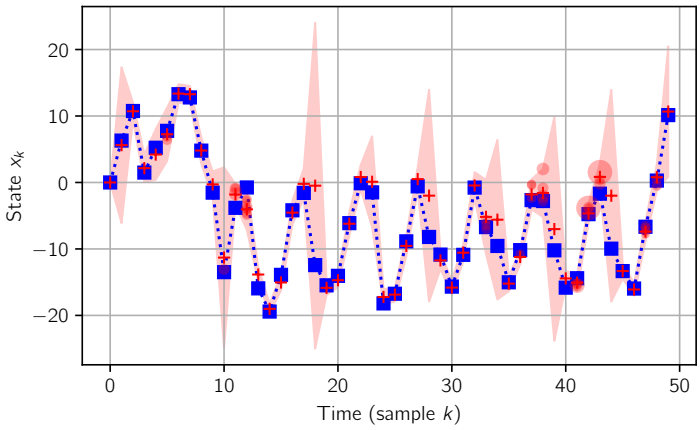
We need to resample from that distribution to generate a new set of n particles with weight $1/n$.

[Q5] Let X be a random variable width PDF p_X and CDF F. Let U be a random variable uniformly distributed in $[0, 1]$. Show that the variable $F^{-1}(U)$ is distributed according to p_X.

Using this result, we can then sample from $\sum_{i=1}^{n} w_k^{(i)} \delta[x - z_k^{(i)}]$. Note that this will not be a very efficient resampling, but that will be good enough for our simple problem.

[Q6] Implement the particle filter *with* the resampling step. Provide a graph illustrating the behavior of the algorithm, showing at least the mean and variance of your filter superposed to the generated data. Also plot the effective number of particles as a function of the time index k. Make sure to discuss your result in comparison with what was obtained earlier.

Your filter should show some improvements as shown below.



[Q7] Derive a linearized version of the non-linear system. Be careful, the system contains a time dependent offset that should be handle carefully.

[Q8] Implement an EKF as discussed in class. Remember that you should linearize around your current estimate.