

Bonus assignment 2

● Graded

Student

Theodore Johann Wilkening

Total Points

6 / 8 pts

Question 1

(no title)

1 / 1 pt

+ 1 pt Correct

Question 2

(no title)

1 / 1 pt

+ 1 pt Correct

+ 0 pts Click here to replace this description.

Question 3

(no title)

1 / 1 pt

+ 1 pt Correct

+ 0 pts Click here to replace this description.

Question 4

(no title)

1 / 1 pt

+ 1 pt Correct

+ 0 pts Click here to replace this description.

Question 5

(no title)

1 / 1 pt

+ 1 pt Correct

+ 0 pts Click here to replace this description.

Question 6

(no title)

0 / 1 pt

+ 1 pt Correct

+ 0 pts Click here to replace this description.

Question 7

(no title)

1 / 1 pt

 + 1 pt Correct

+ 0 pts Click here to replace this description.

Question 8

(no title)

0 / 1 pt

+ 1 pt Correct

 + 0 pts Click here to replace this description.

Question assigned to the following page: [1](#)

Goal: Compare EKF w/ Particle Filter.

Scalar, non-linear model:

$$x_k = \frac{1}{2} x_{k-1} + \frac{25 x_{k-1}}{1 + x_{k-1}^2} + 8w_2(1.2(k-1)) + u_k \quad (1)$$

$$y_k = \frac{1}{20} x_k^2 + v_k \quad (2)$$

$\{u_n\}, \{v_n\}$ are white, Gaussian noise sequences w/unit variance.

$$Q_n = 1, R_n = 1$$

$$\text{initial state: } x_0 = 0.1, \hat{x}_0 = 0, \hat{\sigma}_0^2 = 2$$

note: both state & measurement processes are non-linear.

|Q1) Plot ^{the state process} via python (see python code for plot) |

Now for the Particle Filter

method:

1) draw n samples from the prior $x_0^{(i)} \sim p(x_0)$ $i=1 \dots n$
and set $w^{(i)} = 1/n$ for $i=1 \dots n$

2) For each $k=1 \dots T$

a) draw samples $x_k^{(i)}$ from importance distributions
 $x_k^{(i)} \sim \pi(x_k | x_{0:k-1}^{(i)}, y_{0:k})$ $i=1 \dots n$

b) compute new weights

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

3) If the effective n_{eff} of particle is too low, re-sample and reset to

uniform weights, where $n_{eff} \approx \frac{1}{\sum_{j=1}^n (w_j^{(i)})^2}$ (ie the n_{eff} of effective particles
at a particular time-step)

Questions assigned to the following page: [1](#) and [2](#)

But, in fact we did not specify how to choose the importance distribution and how to perform the sampling.

(Basically, try to get an as to real dist. as possible, and be Markovian)

Now, we choose

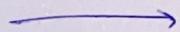
$$\pi(x_k^{(i)} | x_{0:n-1}^{(i)}, y_{1:n}) \triangleq p(x_k^{(i)} | x_{k-1}^{(i)}) \quad (3)$$

that is, 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 $\pi(x_k | x_{0:k-1}^{(i)}, y_{1:n})$ then reduces to computing:

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

and adding the realization of the 0-mean white Gaussian noise.



Questions assigned to the following page: [2](#) and [3](#)

Q2 cont'd |

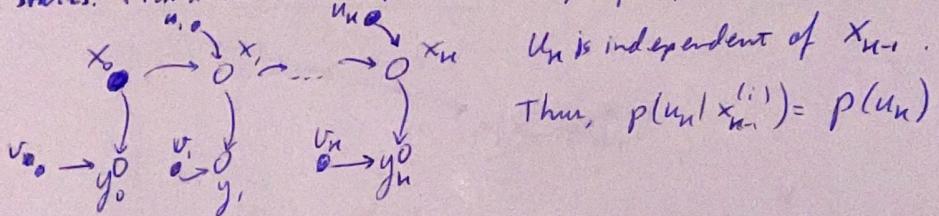
from (3) we have

$$\pi(x_n | x_{n-1}^{(i)}, y_{1:n}) \triangleq p(x_n | x_{n-1}^{(i)})$$

From eqtn (1) $\Rightarrow p(x_n | x_{n-1}^{(i)}) = p\left(\frac{1}{2}x_{n-1} + \frac{25x_{n-1}}{1+x_{n-1}^2} + 8 \cos(1.2(n-1)) + u_n | x_{n-1}^{(i)}\right)$

$$= \frac{1}{2}x_{n-1}^{(i)} + \frac{25x_{n-1}^{(i)}}{1+(x_{n-1}^{(i)})^2} + 8 \cos(1.2(n-1)) + p(u_n | x_{n-1}^{(i)})$$

Since u_n is Gaussian white noise, it is independent of past and present states. From Functional Dependence Graph perspective, we see that indeed



In summary,

$$\pi(x_n | x_{n-1}^{(i)}, y_{1:n}) \triangleq p(x_n | x_{n-1}^{(i)}) = \frac{1}{2}x_{n-1}^{(i)} + \frac{25x_{n-1}^{(i)}}{1+(x_{n-1}^{(i)})^2} + 8 \cos(1.2(n-1)) + p(u_n)$$

so that when we sample from the importance distribution, given $\{x_{n-1}^{(i)}\}$, then the resultant sample is eqtn (1) plus the realization of $\{u_n\}$.

[Q3] Explain expression for computation of weights $w_n^{(i)}$ as a function of $w_{n-1}^{(i)}$. Make the subsequent normalization to avoid unnecessary computations.



Question assigned to the following page: [3](#)

Q3 cont'd

$$w_n^{(i)} \propto w_{n-1}^{(i)} \frac{p(y_n | x_n^{(i)}) p(x_n^{(i)} | x_{n-1}^{(i)})}{\pi(x_n^{(i)} | x_{0:n-1}^{(i)}, y_{0:n})}$$

where $\pi(x_n^{(i)} | x_{0:n-1}^{(i)}, y_{0:n})$
 $= p(x_n^{(i)} | x_{n-1}^{(i)})$

~~how to deal w/ this?
is it the same as
 $\pi(x_n^{(i)} | x_{0:n-1}^{(i)}, y_{0:n})$?~~
I don't think so... + there's a
comma here...
↳ it is the same \Rightarrow should be
a comma already!

thus,

$$w_{n-1}^{(i)} \frac{p(y_n | x_n^{(i)}) p(x_n^{(i)} | x_{n-1}^{(i)})}{\pi(x_n^{(i)} | x_{0:n-1}^{(i)}, y_{0:n})} = w_{n-1}^{(i)} p(y_n | x_n^{(i)})$$

$\rightarrow p(x_n^{(i)} | x_{n-1}^{(i)})$

and $\mathbb{E}[p(y_n | x_n^{(i)})] = \mathbb{E}\left(\frac{1}{20}x_n^2 + v_n | x_n^{(i)}\right) = \mathbb{E}\left(\frac{1}{20}x_n^2 | x_n^{(i)}\right) + \mathbb{E}(v_n | x_n^{(i)})$

note: $\mathbb{E}(v_n | x_n^{(i)}) = \mathbb{E}(v_n)$ b/c v_n is independent of $x_n^{(i)}$

and thus $\mathbb{E}(v_n) = 0$, thus;

$$\boxed{\mathbb{E}[p(y_n | x_n^{(i)})] = \mathbb{E}\left(\frac{1}{20}x_n^2 | x_n^{(i)}\right) = \frac{1}{20}(x_n^{(i)})^2} \quad (5)$$

~~such that~~

$$\cancel{w_n^{(i)} \propto w_{n-1}^{(i)} p(y_n | x_n^{(i)})} = \cancel{w_{n-1}^{(i)} (x_n^{(i)})^2}$$

$$\Rightarrow \boxed{w_n^{(i)} \propto \frac{1}{20} w_{n-1}^{(i)} (x_n^{(i)})^2} \quad (6)$$

Question assigned to the following page: [3](#)

A3 cont'd)

each y_n is Gaussian distributed around the mean, due to the Gaussian noise $\{v_n\}$. Thus,

$$y_n \sim N(\mathbb{E}[y_n], 1) \quad \text{since } v_n \sim N(0, 1)$$

$$\begin{aligned} \text{Note: } K_y &= \mathbb{E}[(y_n - \mu_y)(y_n - \mu_y)^T] = \mathbb{E}\left[\left(\frac{1}{20}x_n^{(i)} + v_n - \frac{1}{20}x_n^{(i)}\right)\left(\frac{1}{20}x_n^{(i)} + v_n - \frac{1}{20}x_n^{(i)}\right)^T\right] \\ &= \mathbb{E}(v_n^2) = R_n = 1 \end{aligned}$$

$$\text{Now, } P_y(y_n | x_n^{(i)}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2} \left(\frac{y_n - \mu_y}{\sigma^2}\right)^2\right]$$

$$\Rightarrow \boxed{P_{y_n}(y_n | x_n^{(i)}) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2} (y_n - \frac{1}{20}x_n^{(i)})^2\right]} \quad (7)$$

such that,

$$\begin{aligned} w_n^{(i)} &\propto w_{n-1}^{(i)} p(y_n | x_n^{(i)}) \\ \Rightarrow \boxed{w_n^{(i)} &\propto w_{n-1}^{(i)} \left(\frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2} (y_n - \frac{1}{20}x_n^{(i)})^2\right] \right)} \end{aligned} \quad (8)$$

Question assigned to the following page: [4](#)

Q4] Implement particle filter (without) the resampling step. Use 200 part.

4.1) Provide a graph w/ at least mean & var. of filter superposed to data.

4.2) Plot the n_{eff} as a function of time index k .

→ see python code

(recall PF method on page 1, given in problem statement)

↓

1) draw n samples from the prior $x_0^{(i)} \sim p(x_0)$ $i=1\dots n$

and set $w_0^{(i)} = \frac{1}{n}$ for $i=1\dots n$.

↳ note: $x_0 \sim (0, 2)$ from problem statement: $[x_0 = 0.1, \hat{x}_0 = 0, \sigma_0 = 2]$

2) For each $k=1\dots T$

a) draw samples $x_n^{(i)}$ from importance distributions

$$x_n^{(i)} \sim \pi(x_n^{(i)} | x_{0:n-1}^{(i)}, y_{0:n})$$

$$\Rightarrow x_n^{(i)} \sim p(x_n^{(i)} | x_{n-1}^{(i)})$$

$$\text{where } p(x_n^{(i)} | x_{n-1}^{(i)}) = \frac{1}{2} x_{n-1}^{(i)} + \frac{25 x_n^{(i)}}{1 + (x_{n-1}^{(i)})^2} + 8 \cos(1.2(n-1)) + p(u_n)$$

$$\text{and } u_n \sim (0, 1)$$

b) compute new weights, and normalize:

$$w_n^{(i)} \propto w_{n-1}^{(i)} \frac{p(y_n | x_n^{(i)}) p(x_n^{(i)} | x_{n-1}^{(i)})}{\pi(x_n^{(i)} | x_{0:n-1}^{(i)}, y_{0:n})} \Rightarrow w_n^{(i)} \propto \frac{1}{20} w_{n-1}^{(i)} \left(\frac{x_n^{(i)}}{20} \right)^2 w_{n-1}^{(i)} \left(\frac{1}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} (y_n - \frac{x_n^{(i)}}{20})^2 \right] \right)$$

$$\text{then, } \hat{w}_k^{(i)} = \frac{w_k^{(i)}}{\|w_k^{(i)}\|} \sum_{i=1}^n w_k^{(i)}, i=1\dots n, \|w_k^{(i)}\| = \sqrt{\sum_{i=1}^n (w_k^{(i)})^2}$$

3) skip for this part/problem

want only that $\sum w_k^{(i)} = 1$, to prepare for re-sampling (to standardize weights) $\sqrt{5}$

Questions assigned to the following page: [4](#) and [5](#)

Q4 contd

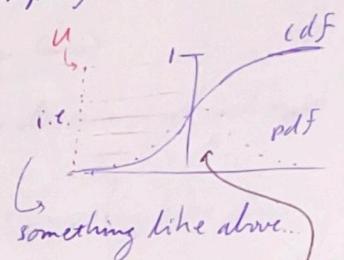
$$\text{weighted mean: } \mu_w = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

$$\text{biased weighted variance: } \hat{\sigma}_w^2 = \frac{\sum_{i=1}^n w_i (x_i - \mu_w)^2}{\sum_{i=1}^n w_i}$$

Q5 Let X be a random variable with 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 . s.t. we can sample from

$$\sum_{i=1}^n w_n^{(i)} \delta(x - x_n^{(i)}) \quad (\text{CDF})$$

$$\text{Let } F = \sum_{i=1}^n w_n^{(i)} \delta(x - x_n^{(i)})$$



$$P(F^{-1}(U) \leq x) = F(x)$$

$F^{-1}(x)$ will give the value x s.t. $F(x) = x$ ($= P(X \leq x)$)

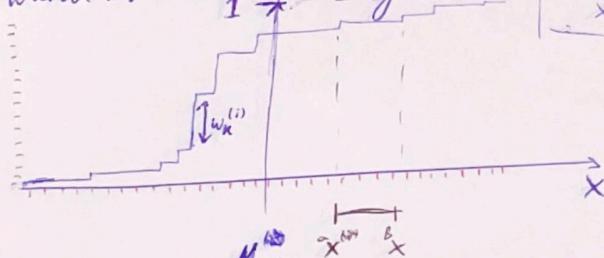
so, the probability that $F^{-1}(U) \leq x$ is $P(F^{-1}(U) \leq x) =$

$$F^{-1}(U) \leq x \Rightarrow U \leq F(x) \Rightarrow P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$$

$$\text{So, } F(x) = \sum_{i=1}^n w_n^{(i)} \delta(x - x_n^{(i)})$$

(distributed according to p_x)

which will look something like:



$$\text{So, } F^{-1}(0.9) = [x^\alpha, x^\beta]$$

higher concentration of points around mean for $F^{-1}(U)$

since U is a uniform distribution, any value in $[0, 1]$ is equally possible, s.t. the probability that $U < F(x)$ is simply the value of $F(x)$.

so have to figure out how to split and sample. if $F(U_i)$ is between two values, choose the lower one. Then within the range of values that it is $[x^\alpha, x^\beta]$, randomly select a position in between. (u/uniform distr.)

Question assigned to the following page: [6](#)

Q6 Plot the PF with re-sampling if $n_{eff} < 20$ (10% of samples provides)

method for resampling:

at step k , if $n_{eff}[k-1] < 20$:

draw new samples according to:

$$F(x) = \sum_{i=1}^n w_n^{(i)} \delta(x - x_n^{(i)}) \rightarrow$$

~~bias~~ = iterative sum of $w^{(i)}$

$V = \text{random}(\text{uniform}, 200)$

foreach $U^{(i)}$:

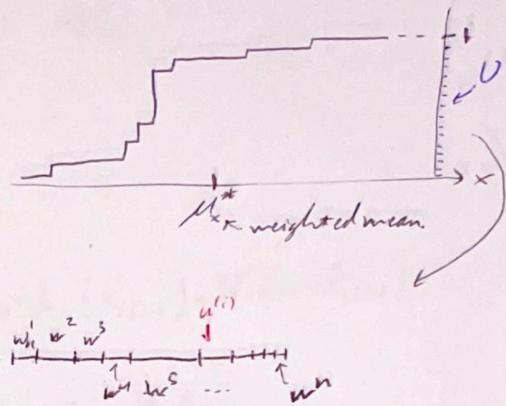
foreach bin: (in reverse)

if $U^{(i)} \geq \text{bin}$:

$$x_{\text{new}}^{(i)} = x_K^{(i)} + (x_n^{(i)} + x_n^{(i+1)}) * \text{random(uniform)}$$

~~break/continue~~

and continue algorithm...



do this so not the same point each time

[See python code for implemented PF and algorithm]

(Q7) →

PF

Question assigned to the following page: [7](#)

Q7 Derive a linearized version of the non-linear system:

System:

$$y_n = \frac{1}{20} x_n^2 + v_n$$

$$x_n = \frac{1}{2} x_{n-1} + \frac{25 x_{n-1}}{1 + x_{n-1}^2} + 8 \cos(1.2(n-1)) + u_n$$

per Lec 17.

$$x_i = f_i(x_{i-1}) + u_{i-1}, \quad y_i = h_i(x_i) + v_i$$

$$f_i(x_i) \approx f_i(\hat{x}_{i|i-1}) + F_i(x_i - \hat{x}_{i|i-1}), \quad h_i(x_i) \approx h_i(\hat{x}_{i|i-1}) + H_i(x_i - \hat{x}_{i|i-1})$$

where $F_i = \begin{bmatrix} \frac{\partial f_{i1}}{\partial x_{i1}} & \dots & \frac{\partial f_{in}}{\partial x_{in}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{in}}{\partial x_{i1}} & \dots & \frac{\partial f_{in}}{\partial x_{in}} \end{bmatrix}$ and similar for H_i

$$\text{let } \tilde{y}_n = x_n - 8 \cos(1.2(n-1))$$

$$\Rightarrow \tilde{y}_n = \frac{1}{2} x_{n-1} + \frac{25 x_{n-1}}{1 + x_{n-1}^2} + u_n \rightarrow \text{let } \boxed{f_h(x_{n-1}) = \frac{1}{2} x_{n-1} + \frac{25 x_{n-1}}{1 + x_{n-1}^2}} \quad (7.1)$$

$$\Rightarrow \boxed{F_h = \frac{\partial f_h}{\partial x_{n-1}} = \frac{1}{2} + 25(1+x_n^2)^{-1} + 25x_n(-1)(2x_{n-1})^{-2}} \quad (7.2)$$

s.t. $\tilde{y}_{n+1} \approx f_h(\hat{x}_{n|n}) + F_h(\hat{x}_{n|n})(x_n - \hat{x}_{n|n}) + u_n$

$$\Rightarrow \boxed{x_{n+1} \approx f_h(\hat{x}_{n|n}) + F_h(\hat{x}_{n|n})(x_n - \hat{x}_{n|n}) + 8 \cos(1.2(n-1)) + u_n} \quad (7.3)$$

$$\text{let } \boxed{h_n(x_n) = \frac{1}{20} x_n^2} \Rightarrow h_n(x_n) \approx h_n(\hat{x}_{n|n-1}) + H_n(\hat{x}_{n|n-1})(x_n - \hat{x}_{n|n-1}) \quad (7.4)$$

$$\text{s.t. } \boxed{y_n = h_n(\hat{x}_{n|n-1}) + H_n(\hat{x}_{n|n-1})(x_n - \hat{x}_{n|n-1}) + v_n} \quad (7.5)$$

$$\boxed{H_n(x_n) = \frac{\partial h_n}{\partial x_n} = \frac{1}{10} x_n} \quad (7.6)$$

Questions assigned to the following page: [7](#) and [8](#)

EKF equations:

$$\left[\begin{array}{l} \hat{x}_{i+1|i} = f_i(\hat{x}_{i|i}) \\ \hat{x}_{i|i} = \hat{x}_{i|i-1} + K_{f,i}(y_i - h_i(\hat{x}_{i|i-1})) \\ K_{f,i} = P_{i|i-1} H_{i|i}^T (H_{i|i} P_{i|i-1} H_{i|i}^T + R_i)^{-1} \\ P_{i|i} = (I - K_{f,i} H_{i|i}) P_{i|i-1} \\ P_{i+1|i} = F_{ii} P_{i|i} F_{ii}^T + G_i Q_i G_i^T \quad , \quad G_i = \text{Identity} \end{array} \right]$$

Question assigned to the following page: [1](#)

Table of Contents

- [1 \[Q1\] Make a pot of the trajectory. This will serve as a reference throughout the problem](#)
- ▼ [2 \[Q4\] Implement the Particle Filter without the resampling step](#)
 - [2.1 \[Q4.1\] Provide a graph with at least the mean and variance of the filter superposed to the data.](#)
 - [2.2 \[Q4.2\] Plot the \$n_{eff}\$ as a function of time index k.](#)
- ▼ [3 \[Q6\] Implement the Particle Filter with the resampling step](#)
 - [3.1 Illustration of sorted particles and weights at a single step](#)
 - [3.2 \[Q6.1\] The Particle Filter algorithm \(with resampling step\)](#)
 - [3.3 \[Q6.2\] Plot mean and variance superposed to trajectory](#)
 - [3.4 \[Q6.3\] Plot the \$n_{eff}\$ as a function of time step k](#)
 - [3.5 \[Q6.4\] Discussion of Results](#)

ECE6555 HW5

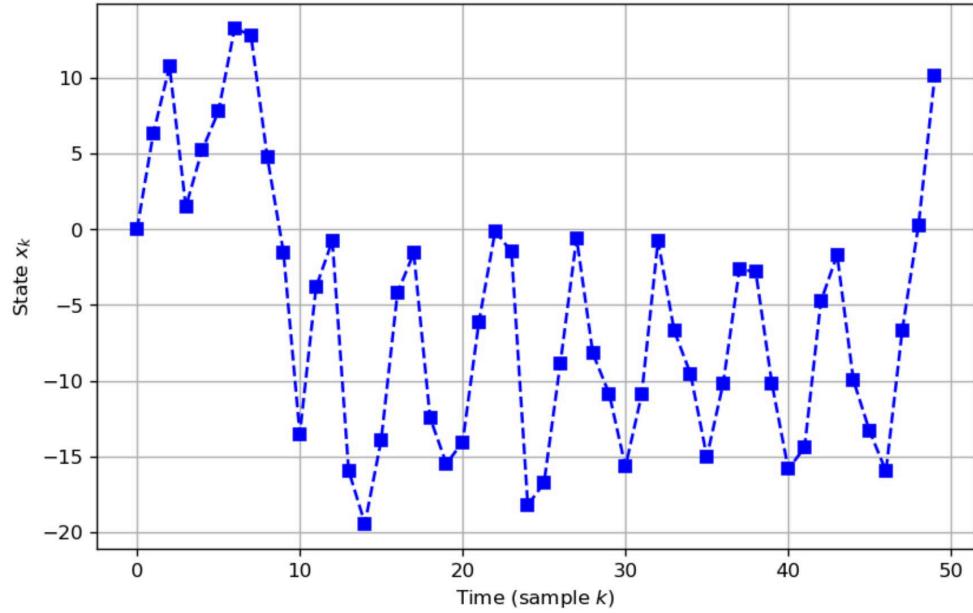
Author: Teo Wilkening Due Date: 2022-12-16

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

```
In [1]: ❶ import numpy as np
❷ import matplotlib.pyplot as plt
❸
❹ # Create the trajectory
❺ np.random.seed(202212)
❻ NumSteps = 50
❼ TimeScale = np.arange(1,NumSteps,1)
❽ x0=0
❾ sigma=1
❿
❿ x = [x0]
❿ y = [0]
❿ for k in TimeScale:
❽   xk = 0.5*x[-1]+25*x[-1]/(1+x[-1]**2)+8*np.cos(1.2*(k-1))+np.random.randn()
❽   yk = 1/20*xk**2+np.random.randn()
❽   x.append(xk)
❽   y.append(yk)
```

Question assigned to the following page: [1](#)

```
In [2]: # Plot the trajectory
1 fig, ax = plt.subplots(figsize=(8,5), dpi=120)
2
3
4 ax.plot(np.insert(TimeScale,0,0),x,'b--')
5 ax.grid(True)
6 ax.plot(np.insert(TimeScale,0,0),x,'bs',markersize=6)
7 #plt.legend(['line','markers'])
8 ax.set_ylabel(r'State $x_k$')
9 ax.set_xlabel(r'Time (sample $k$)')
10
11 plt.show()
```



2 [Q4] Implement the Particle Filter without the resampling step

Use 200 particles.

Question assigned to the following page: [4](#)

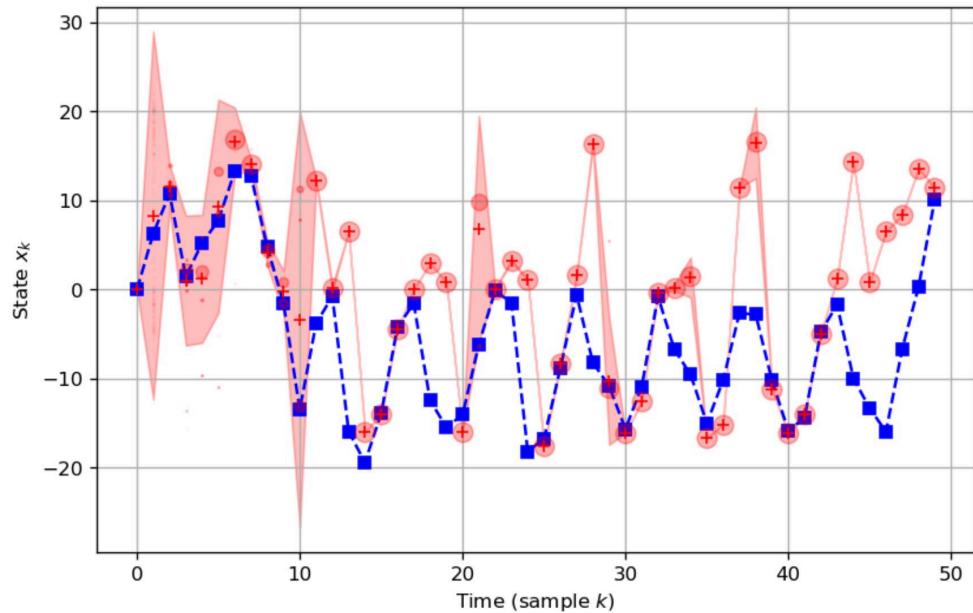
```
In [12]: # 1) draw n samples from the prior
# 2) for each k = 1...T
#     a) draw samples x_k(i) from the importance distribution
#     b) compute the new weights
#     c) normalize the new weights
#
# initialize x^i_k and w^i_k matrices to keep track of state estimation distributions and weights
n = 200 # number of particles
xki = np.zeros((NumSteps,n),dtype=float)
wki = np.zeros((NumSteps,n),dtype=float)
11
12 # 1) draw n samples from the prior
13 x0_mu, x0_sigma = 0, np.sqrt(2)
14 x0 = np.random.normal(x0_mu, x0_sigma, n)
15 w0 = 1/n*np.ones(n)
16
17 # insert the samples from the prior into our matrices for keeping track of things
18 xki[0,:] = x0
19 wki[0,:] = w0
20
21 # initialize noise Gaussian parameters
22 u_mu, u_sigma = 0, 1
23 v_mu, v_sigma = 0, 1
24
25 # 2) for each k = 1...T
26 mean = np.zeros(NumSteps) # keep track of the mean of the particles
27 var = np.zeros(NumSteps) # keep track of the variance of the particles at each step
28 neff = np.zeros(NumSteps)
29
30 for k in np.arange(1,NumSteps,1):
    # a) draw samples x_k(i) from the importance distribution
31     xki[k,:] = 1/2*xki[k-1,:] + 25*xki[k-1,:](1 + xki[k-1,:]**2) + 8*np.cos(1.2*(k-1)) +
    np.random.normal(u_mu, u_sigma,n)
32     # print(sum(xki[k,:]))
33     # b) compute the new weights
34     wki[k,:] = wki[k-1,:]*1/np.sqrt(2*np.pi)*np.exp(-0.5*(y[k] - 1/20*(xki[k-1,:]**2))**2)
35     # c) normalize the new weights
36     wki[k,:] = wki[k,:]/sum(wki[k,:])
37     mean[k] = np.average(xki[k,:],weights=wki[k,:])
38     var[k] = np.average((xki[k,:] - mean[k])**2,weights=wki[k,:])
39     #var[k] = np.average((xki[k,:])**2,weights=wki[k,:]) - (mean[k])**2
40     neff[k] = 1/sum(wki[k,:]**2)
41
42 # track mean for later analysis
43 mean_pf = mean
44
```

Question assigned to the following page: [4](#)

2.1 [Q4.1] Provide a graph with at least the mean and variance of the filter superposed to the data.

```
In [4]: # Plot the trajectory
1 fig, ax = plt.subplots(figsize=(8,5), dpi=120)
2
3
4 ax.plot(np.insert(TimeScale,0,0),x,'b--')
5 ax.grid(True)
6 ax.plot(np.insert(TimeScale,0,0),x,'bs',markersize=6)
7 #plt.legend(['Line','markers'])
8 ax.set_ylabel(r'State $x_k$')
9 ax.set_xlabel(r'Time (sample $k$)')
10 for k in np.arange(1,NumSteps,1):
11     for i in np.arange(n):
12         if wki[k,i] > 1e-3:
13             ax.plot(k,xki[k,i],'ro',markersize=10*wki[k,i],alpha=0.3)
14 ax.plot(mean, 'r+')
15 ax.fill_between(np.arange(NumSteps), mean-2*np.sqrt(var), mean+2*np.sqrt(var), alpha=0.25, color='r')
16 fig.suptitle('Particle Filter with NO re-sampling')
17
18 plt.show()
```

Particle Filter with NO re-sampling

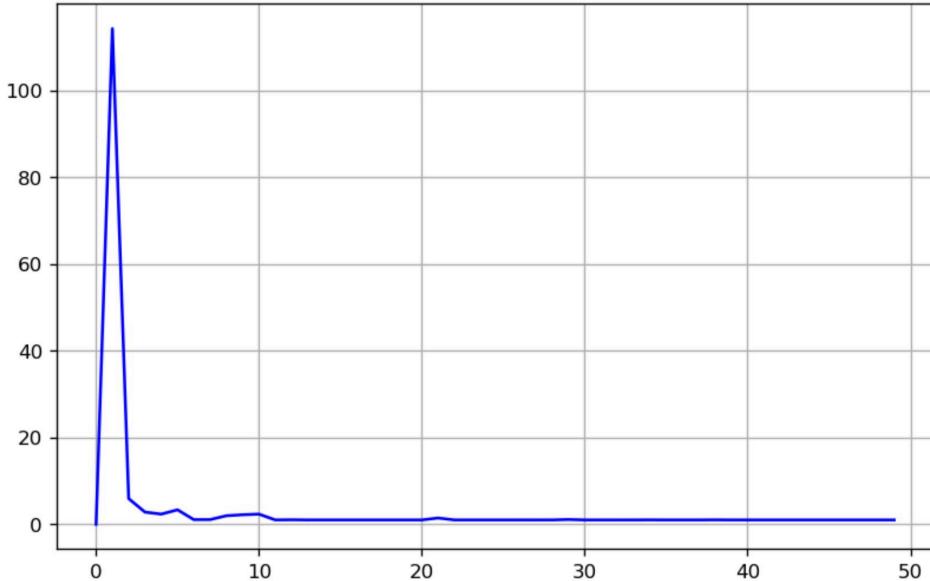


Question assigned to the following page: [4](#)

2.2 [Q4.2] Plot the n_{eff} as a function of time index k.

In [5]:

```
1 # Plot the neff
2 fig, ax = plt.subplots(figsize=(8,5), dpi=120)
3
4 ax.plot(np.insert(TimeScale,0,0),neff,'b')
5 ax.grid(True)
6 plt.show()
```



In []:

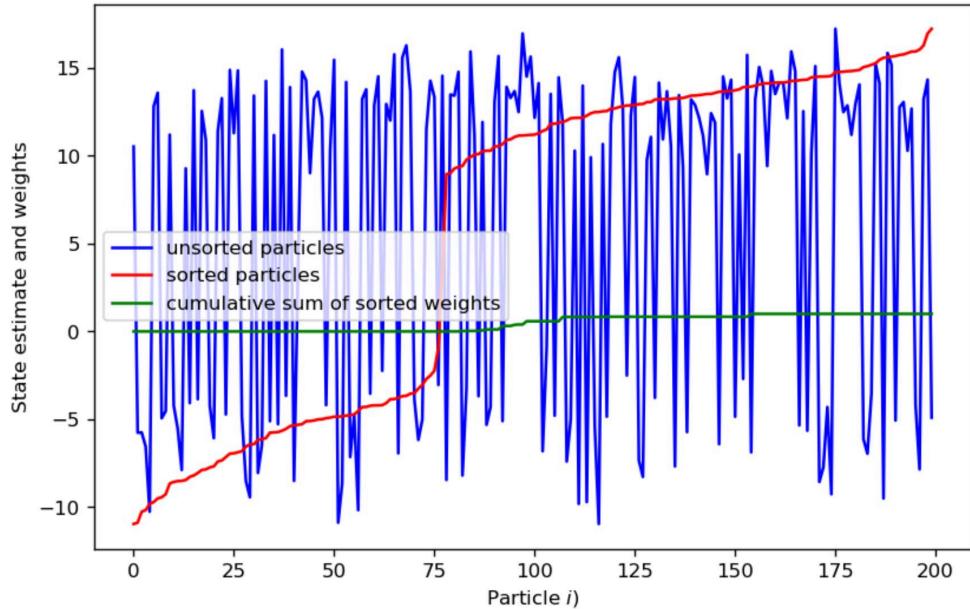
1

Question assigned to the following page: [6](#)

3 [Q6] Implement the Particle Filter with the resampling step

3.1 Illustration of sorted particles and weights at a single step

```
In [6]: fig, ax = plt.subplots(figsize=(8,5), dpi=120)
2
3
4 ax.plot(xki[2,:], 'b')
5 ax.plot(np.sort(xki[2,:]), 'r')
6 ax.plot(np.cumsum(np.take_along_axis(wki[2,:], np.argsort(xki[2,:], axis=0)), 'g')
7 #plt.Legend(['Line', 'markers'])
8 ax.set_ylabel(r'State estimate and weights')
9 ax.set_xlabel(r'Particle $i$')
10 ax.legend(['unsorted particles', 'sorted particles', 'cumulative sum of sorted weights'])
11
12 plt.show()
```



Question assigned to the following page: [6](#)

3.2 [Q6.1] The Particle Filter algorithm (with resampling step)

```
In [13]: # 1) draw n samples from the prior
# 2) for each k = 1...T
#     a) draw samples x_k(i) from the importance distribution
#     b) compute the new weights
#     c) normalize the new weights
#
# initialize x^i_k and w^i_k matrices to keep track of state estimation distributions and weights
n = 200 # number of particles
xki = np.zeros((NumSteps,n),dtype=float)
wki = np.zeros((NumSteps,n),dtype=float)
#
# 1) draw n samples from the prior
x0_mu, x0_sigma = 0, np.sqrt(2)
x0 = np.random.normal(x0_mu, x0_sigma, n)
w0 = 1/n*np.ones(n)
#
# insert the samples from the prior into our matrices for keeping track of things
xki[0,:] = x0
wki[0,:] = w0
#
# initialize noise Gaussian parameters
u_mu, u_sigma = 0, 1
v_mu, v_sigma = 0, 1
#
# 2) for each k = 1...T
mean = np.zeros(NumSteps) # keep track of the mean of the particles
var = np.zeros(NumSteps) # keep track of the variance of the particles at each step
neff = np.zeros(NumSteps)
for k in np.arange(1,NumSteps,1):
    # a) draw samples x_k(i) from the importance distribution
    xki[k,:] = 1/2*xki[k-1,:]+25*xki[k-1,:]/(1+xki[k-1,:]**2)+8*np.cos(1.2*(k-1))+np.random.normal(u_mu, u_sigma,n)
    # print(sum(xki[k,:]))
    # b) compute the new weights
    wki[k,:] = wki[k-1,:]*1/np.sqrt(2*np.pi)*np.exp(-0.5*(y[k]-1/20*(xki[k-1,:]**2))**2)
    # c) normalize the new weights
    wki[k,:] = wki[k,:]/sum(wki[k,:])
    mean[k] = np.average(xki[k,:],weights=wki[k,:])
    var[k] = np.average((xki[k,:]-mean[k])**2,weights=wki[k,:])
    neff[k] = 1/sum(wki[k,:]**2)
    # draw new samples if the number of effective weights is < 20
    if neff[k] < 20:
        print(f"""Effective particles < 20 for step {k}""")
        ind = np.argsort(xki[k,:]) # index sort of the particles
        xki[k,:] = np.take_along_axis(xki[k,:],ind, axis=0)
        wki[k,:] = np.take_along_axis(wki[k,:],ind, axis=0) # sort the weights according to the particles
        bins = np.cumsum(wki[k,:]) # bins from which we are going to sample; cumulative sum of the weights
        uni = np.random.uniform(0,1,n) # uniform distribution used for re-sampling
        uni2 = np.random.uniform(0,1,n) # secondary random sampling for within bins
        for i in np.arange(0,n):
            for j in np.arange(n-1,-1,-1):
                if uni[i] >= bins[j]:
                    xki[k,i] = xki[k,j] + (xki[k,j+1]-xki[k,j])*uni2[i]
        # and reset the weights:
        wki[k,:] = w0
#
# track mean for later analysis
mean_pf_resamp = mean
```

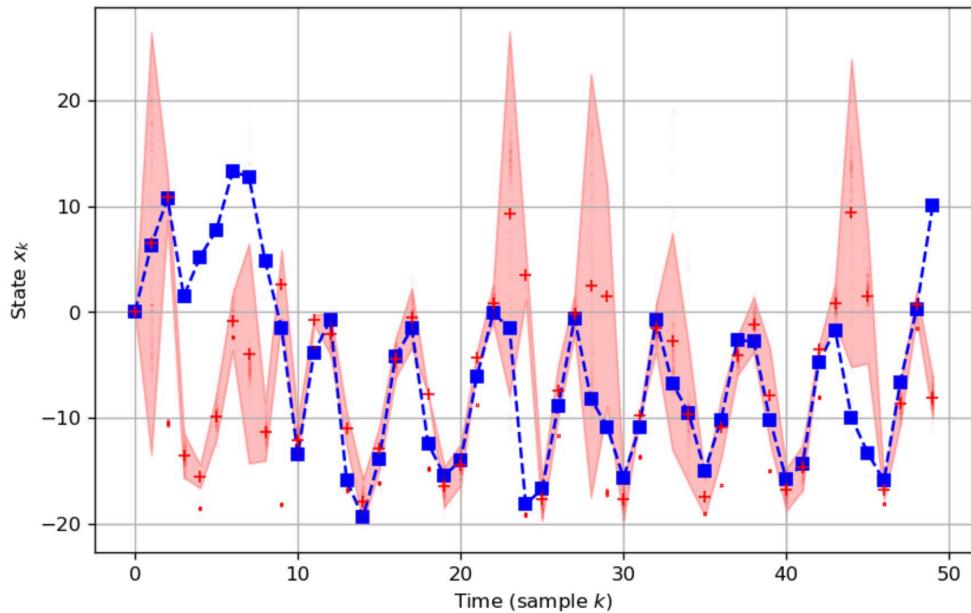
```
Effective particles < 20 for step 2
Effective particles < 20 for step 4
Effective particles < 20 for step 8
Effective particles < 20 for step 9
Effective particles < 20 for step 11
Effective particles < 20 for step 13
Effective particles < 20 for step 16
Effective particles < 20 for step 19
Effective particles < 20 for step 21
Effective particles < 20 for step 24
Effective particles < 20 for step 26
Effective particles < 20 for step 29
Effective particles < 20 for step 31
Effective particles < 20 for step 35
Effective particles < 20 for step 36
Effective particles < 20 for step 39
Effective particles < 20 for step 42
Effective particles < 20 for step 46
Effective particles < 20 for step 48
```

Question assigned to the following page: [6](#)

3.3 [Q6.2] Plot mean and variance superposed to trajectory

```
In [8]: # Plot the trajectory
1 fig, ax = plt.subplots(figsize=(8,5), dpi=120)
2
3
4 ax.plot(np.insert(TimeScale,0,0),x,'b--')
5 ax.grid(True)
6 ax.plot(np.insert(TimeScale,0,0),x,'bs',markersize=6)
7 #plt.legend(['line','markers'])
8 ax.set_ylabel(r'State $x_k$')
9 ax.set_xlabel(r'Time (sample $k$)')
10 for k in np.arange(1,NumSteps,1):
11     for i in np.arange(n):
12         if wki[k,i] > 1e-3:
13             ax.plot(k,xki[k,i],'ro',markersize=10*wki[k,i],alpha=0.3)
14 ax.plot(mean, r'+')
15 ax.fill_between(np.arange(NumSteps), mean-2*np.sqrt(var), mean+2*np.sqrt(var), alpha=0.25, color='r')
16 fig.suptitle('Particle Filter with re-sampling')
17
18 plt.show()
```

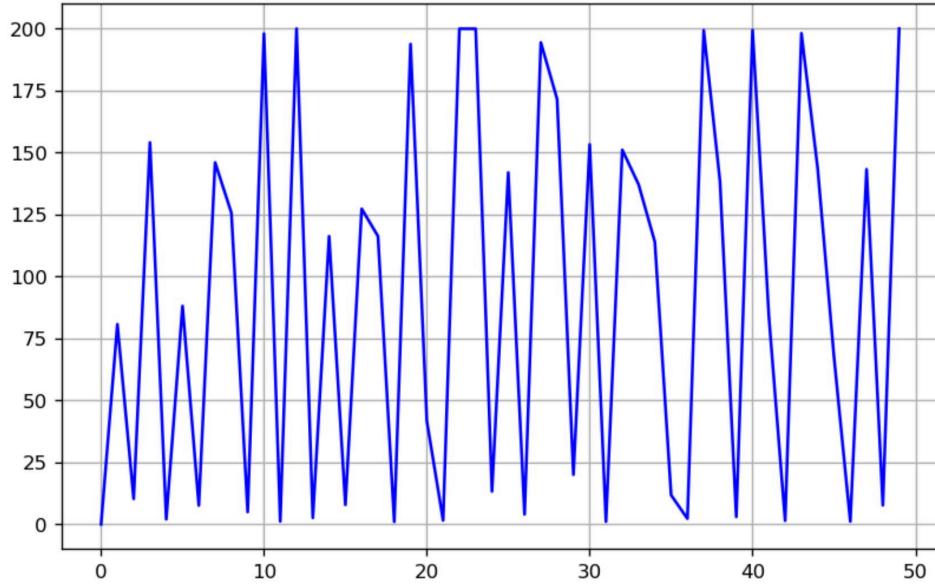
Particle Filter with re-sampling



Question assigned to the following page: [6](#)

3.4 [Q6.3] Plot the n_{eff} as a function of time step k

```
In [9]: # Plot the neff
fig, ax = plt.subplots(figsize=(8,5), dpi=120)
ax.plot(np.insert(TimeScale,0,0),neff,'b')
ax.grid(True)
plt.show()
```



3.5 [Q6.4] Discussion of Results

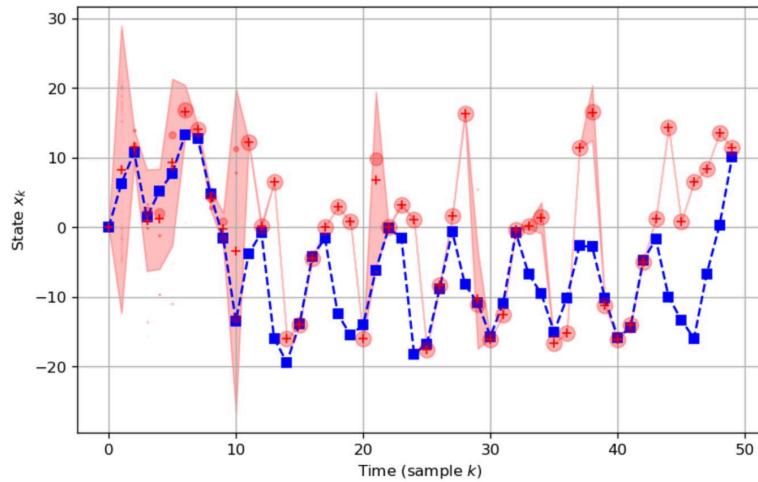
As we can see from the below plots (re-captured from the above code), the particle filter with re-sampling maintains a better tracking on the variance of the estimation, and a slightly better tracking per the MSE calculation below.

```
In [18]: mse_pf = np.sum((mean_pf - x)**2)/NumSteps
mse_pf_resamp = np.sum((mean_pf_resamp - x)**2)/NumSteps
print(f'''MSE of the PF without re-sampling: {mse_pf}''')
print(f'''MSE of the PF with re-sampling: {mse_pf_resamp}'''')
```

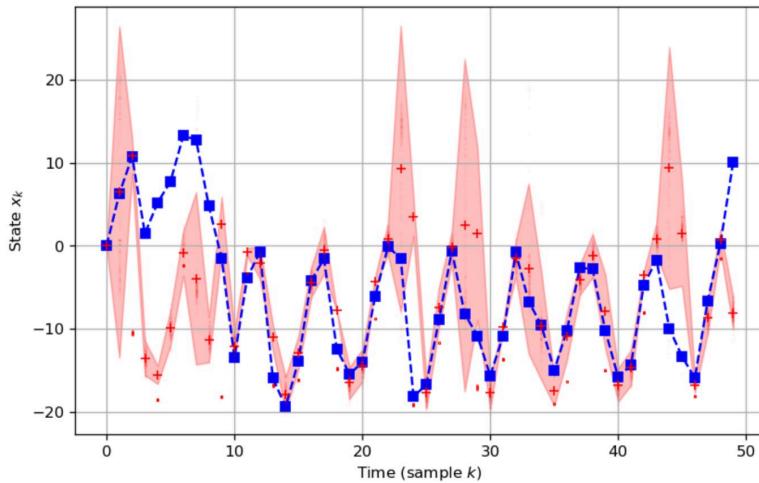
MSE of the PF without re-sampling: 107.36649647973412
MSE of the PF with re-sampling: 76.75540576179377

Question assigned to the following page: [6](#)

Particle Filter with NO re-sampling



Particle Filter with re-sampling



NOTES:

```
In [10]: 1 a = np.array([1, 2, 3, 4])
2 b = np.ones(4) + 1
3 a - b
4 a * b
5 j = np.arange(5)
6 2** (j + 1) - j
```

```
Out[10]: array([ 2,  3,  6, 13, 28])
```

```
In [11]: 1 list = [0, 1, 2, 3, 4]
2 display([0, list])
3 len(x)
```

```
[0, [0, 1, 2, 3, 4]]
```

```
Out[11]: 50
```