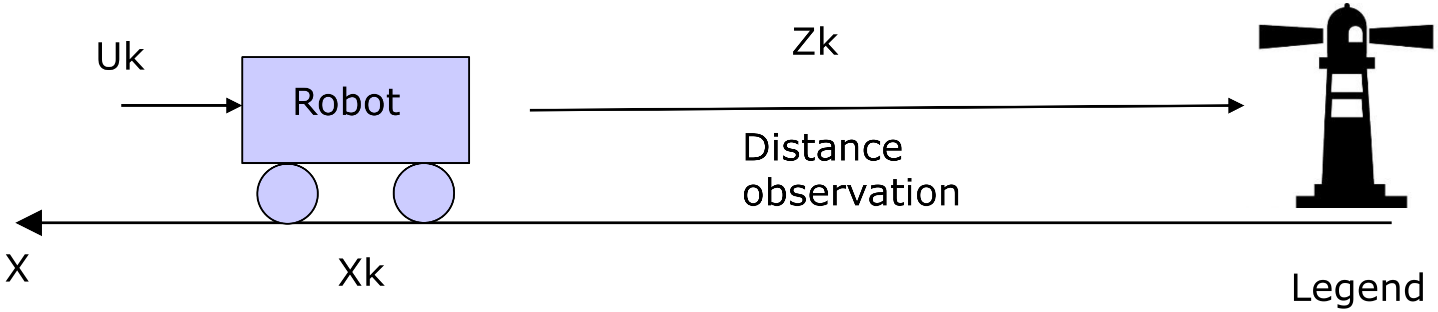
Probabilistic Robotics
From KF to PF

양정연

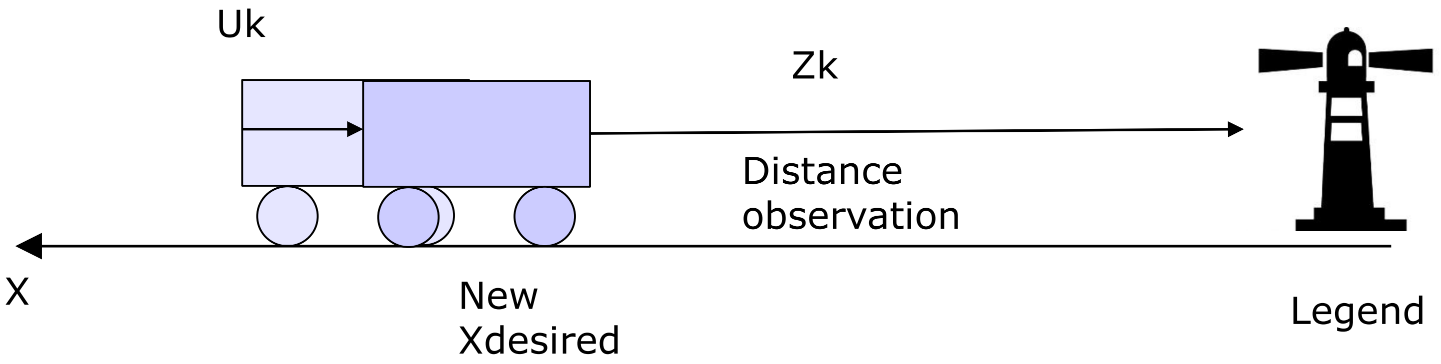
2020/12/10

Now, Localization with Kalman Filter
Where am I?(What is Xk=?)



- We Don’t Know Xk.
- We Only know Observation, Zk.
- But, Xk is noisy by Slip and so on.
- But, Zk is noisy by Matching error and sensor performance.
- What we do all is using Kalman Filter.

Now, Localization with Kalman Filter
Where am I?(What is Xk=?)



- Current Estimate is Initial Estimates $\hat{x}_{k-1|k-1}$
- Control $X_{desired}$ by Uk
- Zk changed then KF will estimate Xk by minimizing P (covariance of state variable estimate)

Remind: Kalman Filter’s Simple Concept

Actual Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

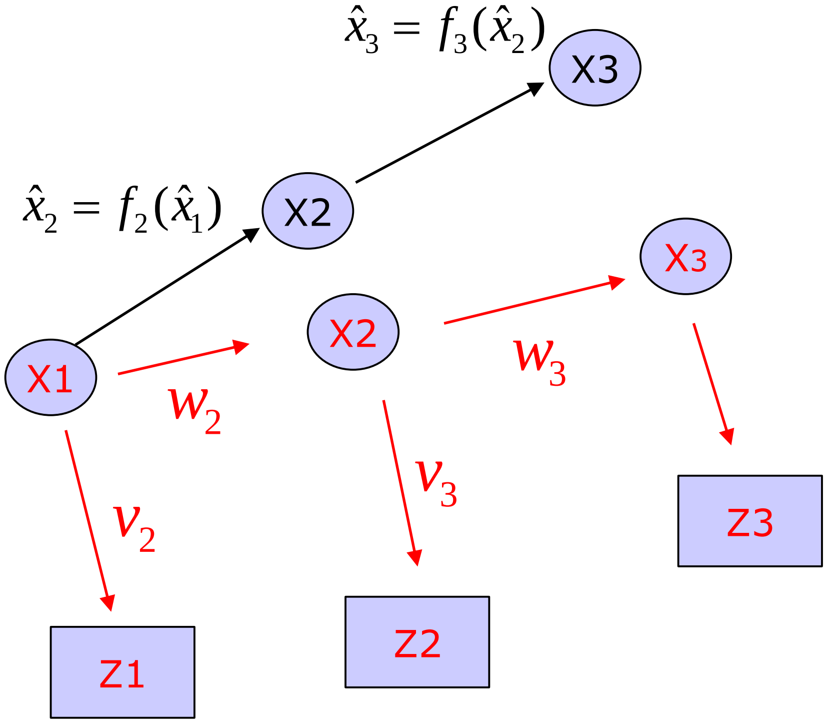
We don’t know noise

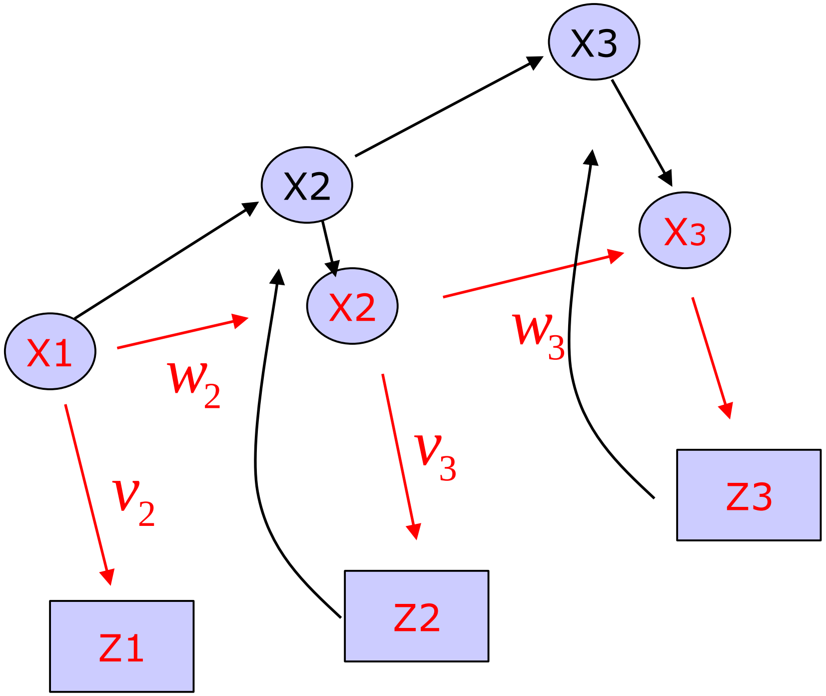
$$\xrightarrow[w_k, v_k]{\hspace{4em}}$$

Estimation

$$\hat{x}_k = f_k(\hat{x}_{k-1})$$

$$\hat{z}_k = h_k(\hat{x}_k)$$





Probabilistic Notation for KF concept

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

Causal relationship from Eqs,

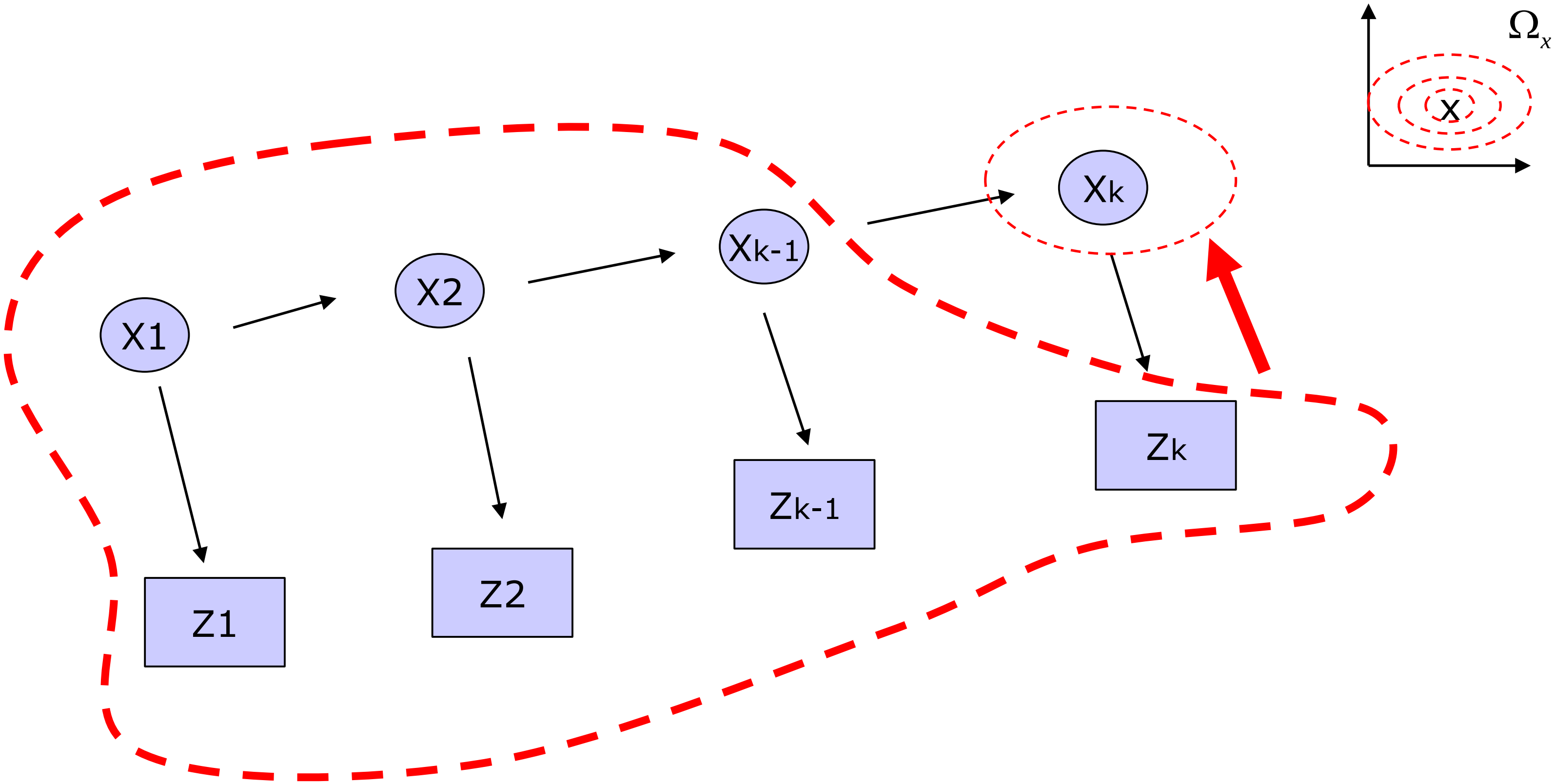
- What we want to know is,

$$Estimaing\ x_k\ from\ \ z_{1:k} = \{z_1, z_2, ..., z_k\}$$

- Reversely, we want to estimate from Z to X, “Z→X”

Probabilistic sense ⇨ $P(x_k \mid z_{1:k})$

Concept of X estimation in Probability



Basic of Probability

Definition of Engineering Concept Conditional Probability

- Prediction

$$P(x_{k+1} \mid z_{1:k})$$

- Filtering or Updating

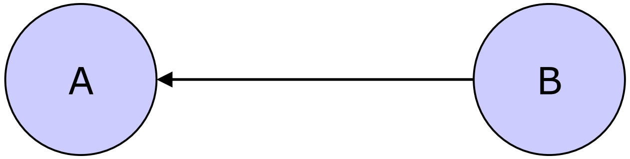
$$P(x_{k} \mid z_{1:k})$$

- Smoothing in DSP

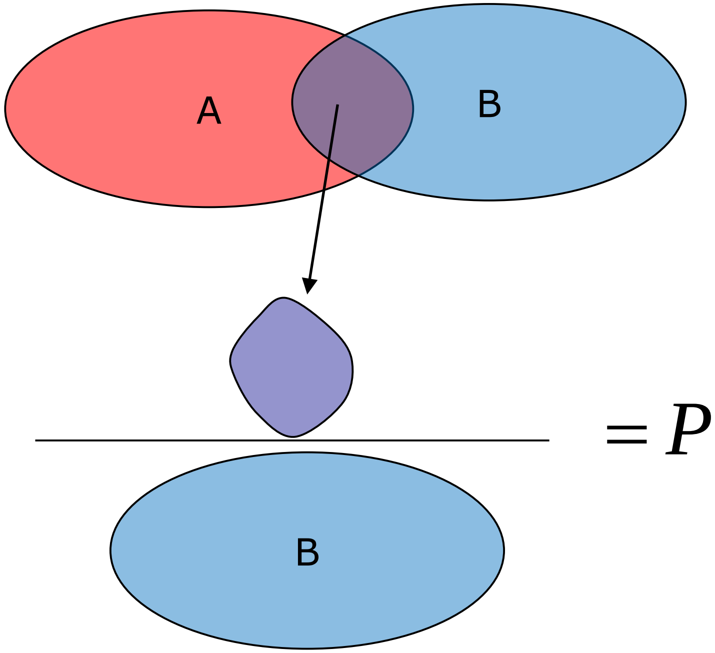
$$P(x_{k-1} \mid z_{1:k})$$

Graph Expression in Probability

- Conditional Probability



A given B



$$= P(A|\mathrm{B}) = \frac{P(\mathrm{A,B})}{P(B)} \left(= \frac{P(A \cap B)}{P(B)} \right)$$

Joint Probability

$$\sum_{x\in X} P(x) = 1$$

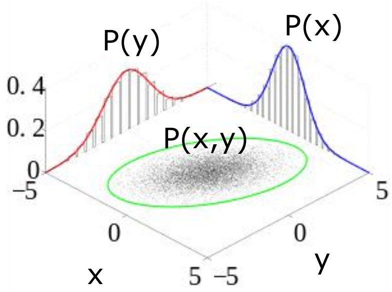
Sum of Probabilities =1

$$\sum_{y\in Y}\sum_{x\in X} P(x,y) = 1$$

Joint Probability P(x,y)

$$\sum_{y\in Y} P(x,y) = P(x) \qquad \sum_{x\in X} P(x,y) = P(y)$$

⇨



$$P(A) = \sum_{b\in B} P(A,b) = \sum_{b\in B} P(A|b)P(b) \rightarrow \int_B P(A|b)P(b)db$$

$$P(A,B) = \sum_{c\in C} P(A,B|c)P(c) = \int_C P(A,B|c)P(c)dc$$

Gaussian Distribution

$$p(z)=\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}} \quad \Rightarrow Z\sim N(0,1)$$

Gaussian Distribution
Normal Distribution

$$\int_{-\infty}^{\infty} p(z)dz=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}dz=1$$

$p(x)$: probabilistic density function

$$x=\sigma z,\quad dx=\sigma dz$$

$$\int_{-\infty}^{\infty} p(z)dz=1=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}dz=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}\frac{dx}{\sigma}=\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}dx=\int_{-\infty}^{\infty}p(x)dx$$

$$\therefore p(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \quad \Rightarrow X\sim N(0,\sigma^2)$$

Tip: Matlab
z=randn
x= sigma*randn

$$z=\frac{x-\mu}{\sigma},\quad dx=\sigma dz$$

$$\therefore p(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad \Rightarrow X\sim N(\mu,\sigma^2)$$

⇨

$$p(x\mid\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}\triangleq p(x)$$

Strict Expression

- Probability Mass function, P (Upper case)
- Probability Density function, p(Lower case)

- $$\underbrace{p(x,w)}_{\text{density}} = \underbrace{p(x|w)}_{\text{density}}\underbrace{P(w)}_{\text{mass}} = \underbrace{P(w|x)}_{\text{mass}}\underbrace{p(x)}_{\text{density}}$$

- $$\rightarrow P(w|x) = \frac{p(x|w)P(w)}{p(x)} = \frac{p(x|w)P(w)}{p(x) = \sum_{W} p(x|W)P(W)}$$

Tip:
p(x,y|a,b) or p(x,y) :
if x,y requires sum or integraton, then p(x,y|a,b) or p(x,y) is a density function

Independence and Dependence

- A and B are Independent

$$P(A,B) = P(A)P(B)$$

- A and B are NOT Independent

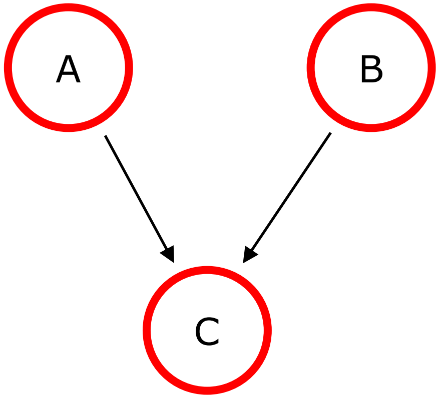
$$P(A,B) = P(A|B)P(B) = P(B|A)P(A)$$

- A and B are Conditionally Independent Given C

$$P(A,B|C) = P(A|C)P(B|C)$$

Bayesian Networks:

Directed Acyclic Graph



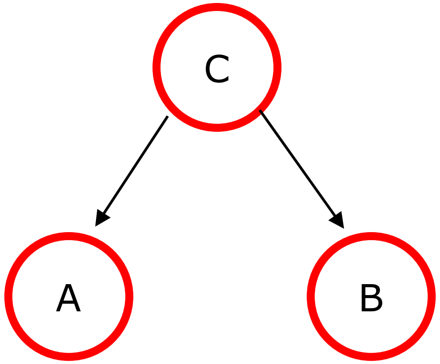
Prob. C given A **and** B
→ Prob. C given A, B

$$P(C|A,B) = \frac{P(A,B,C)}{P(A,B)}$$

$$\therefore P(A,B,C) = P(C|A,B)P(A,B)$$

$$\rightarrow P(A,B,C) = P(C|A,B)P(A)P(B)$$

However, in this case

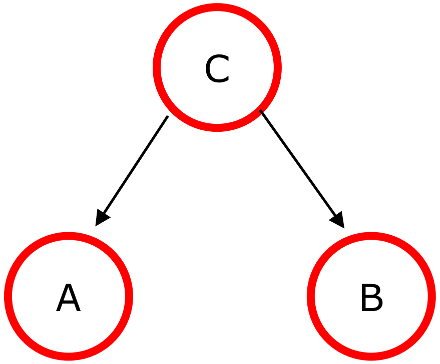


$$P(A,B,C) = \underline{P(A,B|C)}P(C)$$

Prob. A, B given C

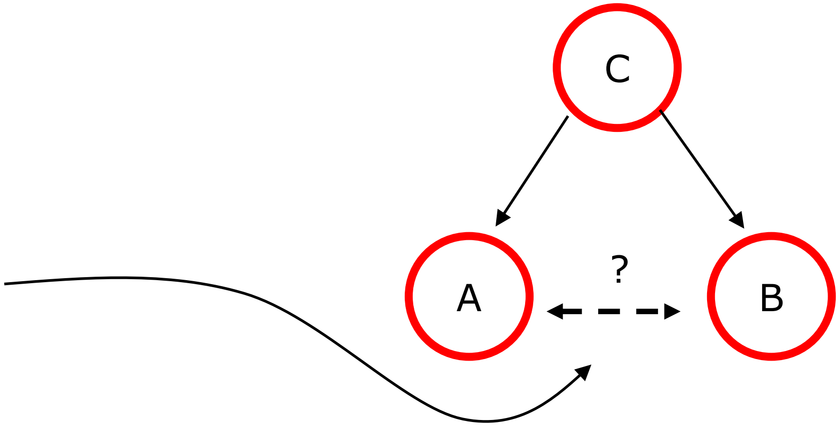
$$P(A,B|C) = ?$$

Conditional Independence



Then,

$P(A,B|C)=?$



$$P(A,B|C)=\frac{P(A,B,C)}{P(C)}=\frac{P(A,B,C)}{P(B,C)}\frac{P(B,C)}{P(C)}=P(A|B,C)P(B|C)$$

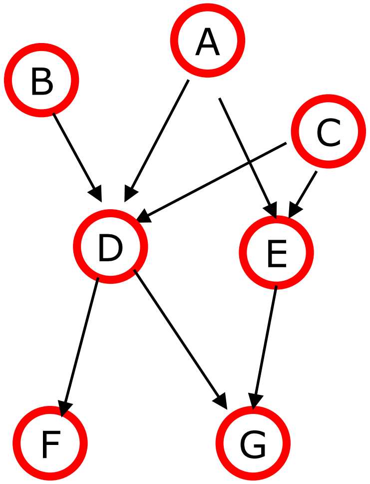
- If A is Independent of B given C, ($A \perp\!\!\!\perp B \mid C$)

$$P(A,B|C)=P(A|B,C)P(B|C)=P(A|C)P(B|C)$$

Conditional Independence

$$P(A,B|C)=P(A|C)P(B|C) \quad or \quad P(A|B,C)=P(A|C)$$

Bayesian Networks with Conditional Independence



$$P(A,B,C,D,E,F,G)$$
$$=P(A)P(B)P(C)P(D|A,B,C)P(E|A,C)P(F|D)P(G|D,E)$$

$$P(A,B,C,D,E,F,G)$$
$$=P(F,G|A,B,C,D,E)P(A,B,C,D,E)$$
$$=P(F,G|D,E)P(D,E|A,B,C)P(A,B,C)$$
$$=\frac{P(F,G,D,E)}{P(D,E)}\frac{P(D,E,A,B,C)}{P(A,B,C)}P(A,B,C)$$
$$=\frac{P(G|F,D,E)\mathrm{P}(\mathrm{F},\mathrm{D},\mathrm{E})}{P(D,E)}\frac{P(D|E,A,B,C)\mathrm{P}(\mathrm{E},\mathrm{A},\mathrm{B},\mathrm{C})}{P(A,B,C)}P(A,B,C)$$
$$=\frac{P(G|D,E)\mathrm{P}(\mathrm{F},\mathrm{D},\mathrm{E})}{P(D,E)}P(D|A,B,C)\mathrm{P}(\mathrm{E}|\mathrm{A},\mathrm{B},\mathrm{C})\mathrm{P}(\mathrm{A},\mathrm{B},\mathrm{C})$$
$$=\frac{P(G|D,E)\mathrm{P}(\mathrm{F}|\mathrm{D},\mathrm{E})\mathrm{P}(\mathrm{D},\mathrm{E})}{P(D,E)}P(D|A,B,C)\mathrm{P}(\mathrm{E}|\mathrm{A},\mathrm{C})\mathrm{P}(\mathrm{A},\mathrm{B},\mathrm{C})$$
$$=P(G|D,E)\mathrm{P}(\mathrm{F}|\mathrm{D})P(D|A,B,C)\mathrm{P}(\mathrm{E}|\mathrm{A},\mathrm{C})\mathrm{P}(\mathrm{A})\mathrm{P}(\mathrm{B})\mathrm{P}(\mathrm{C})$$

General Factorization of Bayesian Net

- Probability of Network is factorized by

$$P(x) = \prod_{k=1}^{K} P(x_k \mid a_k)$$

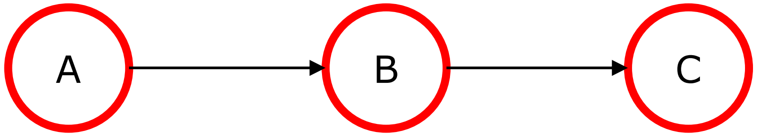
- Remind that

$$P(A, B, C, D, E, F, G)$$
$$= P(A)P(B)P(C)P(D \mid A, B, C)P(E \mid A, C)P(F \mid D)P(G \mid D, E)$$

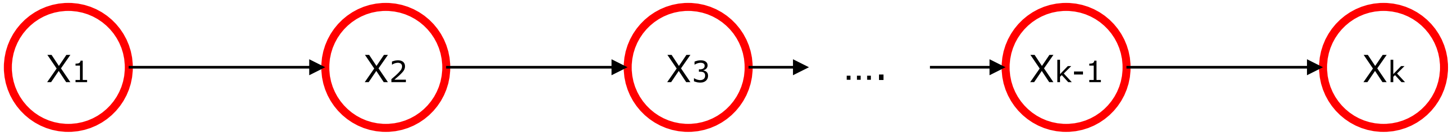
- Why Factorization is useful?
  - Use Logarithm

$$LogP(A, B, C, D, E, F, G)$$
$$= LogP(A) + \mathrm{Log}\,P(B) + \mathrm{Log}\,P(C) + \mathrm{Log}\,P(D \mid A, B, C) +$$
$$\mathrm{Log}\,P(E \mid A, C) + \mathrm{Log}\,P(F \mid D) + \mathrm{Log}\,P(G \mid D, E)$$

Markov Chain



$$P(C \mid A,B) = \frac{P(C,A,B)}{P(A,B)} = \frac{P(C \mid B)P(B \mid A)P(A)}{P(B \mid A)P(A)} = P(C \mid B)$$



$$P(x_k \mid x_1, x_2, .., x_{k-1}) = P(x_k \mid x_{1:k-1}) = P(x_k \mid x_{k-1})$$

$$\therefore P(x_k \mid x_{1:k-1}) = P(x_k \mid x_{k-1}) \quad : \text{Markov} \quad \text{Process}$$

Current state

History

Current state

Previous state

Probability in Markov Process

1. Markov Chain → Transitional Probability

$$P(x_k \mid x_1, x_2, .., x_{k-1}) = P(x_k \mid x_{1:k-1}) = P(x_k \mid x_{k-1})$$

$$\underline{\therefore P(x_k \mid x_{1:k-1}) = P(x_k \mid x_{k-1}) \quad} : \mathrm{Markov\ \ Process}$$

2. Joint Probability

Remind $P(A,B) = \sum_{c \in C} P(A|C)P(B|C)P(C) = \int_C P(A|C)P(B|C)dc$

$$\underline{P(x_k) = \sum_{x_{k-1}} P(x_k, x_{k-1}) = \int P(x_k \mid x_{k-1}) P(x_{k-1}) dx_{k-1}}$$

2. K.F. with Probabilistic Approaches

K.F. Two steps in Probability

- Our goal is,

$$P(x_k \mid z_{1:k})$$

  - From the history of Observation, $Z_{1:k}$, we want to estimate X_k

- Prediction

$$P(x_k \mid z_{1:k-1})$$

- Update (or Filter)

$$P(x_k \mid z_{1:k})$$

Prediction $P(x_k \mid z_{1:k-1})$

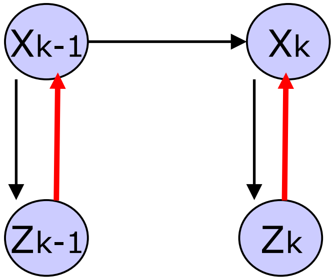
$$P(x_k \mid z_{1:k-1}) = \int_{x_{k-1}} P(x_k, x_{k-1} \mid z_{1:k-1}) dx_{k-1}$$

Remind Joint probability Definition

$$P(x_k \mid z_{1:k-1}) = \int_{x_{k-1}} P(x_k, x_{k-1} \mid z_{1:k-1}) dx_{k-1} = \int_{x_{k-1}} \frac{P(x_k, x_{k-1}, z_{1:k-1})}{P(z_{1:k-1})} dx_{k-1}$$

$$= \int_{x_{k-1}} \frac{P(x_k, x_{k-1}, z_{1:k-1})}{P(x_{k-1}, z_{1:k-1})} \frac{P(x_{k-1}, z_{1:k-1})}{P(z_{1:k-1})} dx_{k-1}$$

$$= \int_{x_{k-1}} \underline{P(x_k \mid x_{k-1}, z_{1:k-1})} P(x_{k-1} \mid z_{1:k-1}) dx_{k-1}$$



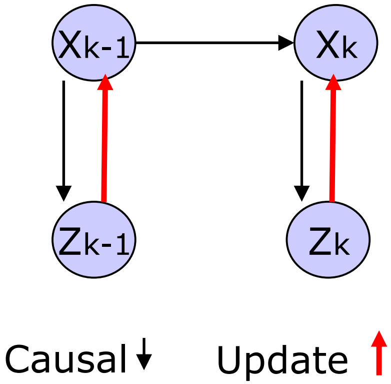
$$\therefore P(x_k \mid z_{1:k-1}) = \int_{x_{k-1}} \underline{P(x_k \mid x_{k-1})} P(x_{k-1} \mid z_{1:k-1}) dx_{k-1}$$

Causal ↓ Update ↑

Tip:(Zk-1 → Xk-1 → Xk) = (Xk-1 → Xk)

Update(Filter) $P(x_k \mid z_{1:k})$

$$
\begin{aligned}
P(x_k \mid z_{1:k}) &= P(x_k \mid z_k, z_{1:k-1}) \\
&= \frac{P(x_k, z_k, z_{1:k-1})}{P(z_k, z_{1:k-1})} = \frac{P(z_k, x_k, z_{1:k-1})}{P(x_k, z_{1:k-1})} \frac{P(x_k, z_{1:k-1})}{P(z_k, z_{1:k-1})} \\
&= P(z_k \mid x_k, z_{1:k-1}) \frac{P(x_k, z_{1:k-1}) / P(z_{1:k-1})}{P(z_k, z_{1:k-1}) / P(z_{1:k-1})} \\
&= P(z_k \mid x_k, z_{1:k-1}) \frac{P(x_k \mid z_{1:k-1})}{P(z_k \mid z_{1:k-1})} \\
&= \eta \underline{P(z_k \mid x_k, z_{1:k-1})} P(x_k \mid z_{1:k-1})
\end{aligned}
$$

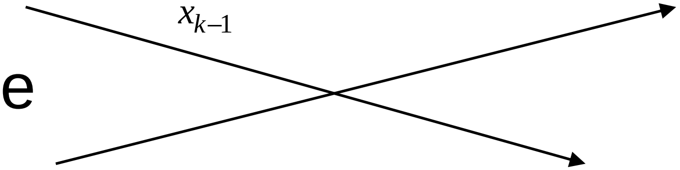


$$
\therefore P(x_k \mid z_{1:k}) = \eta P\underline{(z_k \mid x_k})P(x_k \mid z_{1:k-1})
$$

Prediction and Update with Probability

- Prediction

$$P(x_k \mid z_{1:k-1}) = \int_{x_{k-1}} P(x_k \mid x_{k-1}) P(x_{k-1} \mid z_{1:k-1}) dx_{k-1}$$



- Update

$$P(x_k \mid z_{1:k}) = \eta P(z_k \mid x_k) P(x_k \mid z_{1:k-1})$$

- Easily, with Belief function

$$\mathrm{Bel}(x_k) \triangleq P(x_k \mid z_{1:k})$$

- What is the PROBLEM?

How to Solve Integration?

$z_{1:k} = \varnothing$

Prediction k=0

Bel′(x0) = P(x0)

Update k=0

Bel(x0) = P(x0|z0)

$$= \eta P(z_0 \mid x_0) P(x_0 \mid z_{0-})$$

$$= \eta P(z_0 \mid x_0) P(x_0)$$

$Z_{1:k}$ → Prediction k=1

Bel′(x1) = P(x1|z0)

$$= \int P(x_1 \mid x_0) P(x_0 \mid z_0) dx_0$$

Update k=1

Bel(x1) = P(x1|Z0:1)

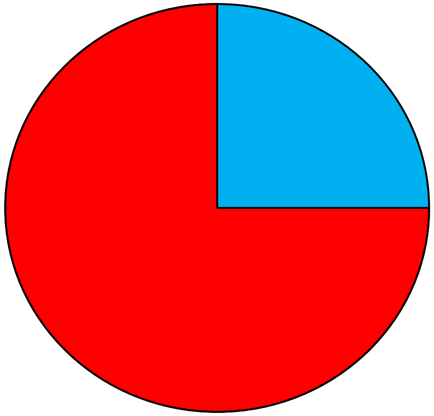
$$= \eta_1 P(z_1 \mid x_1) P(x_1 \mid z_{0:1})$$

3. Particle Filter
Estimation instead of Solving Integration

Monte Carlo Method

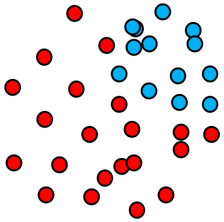
- Monte Carlo Casino
  - Named by Stanislaw Ulam,
  - Developed by John Von Neumann in Eniac.

- We Already know it.



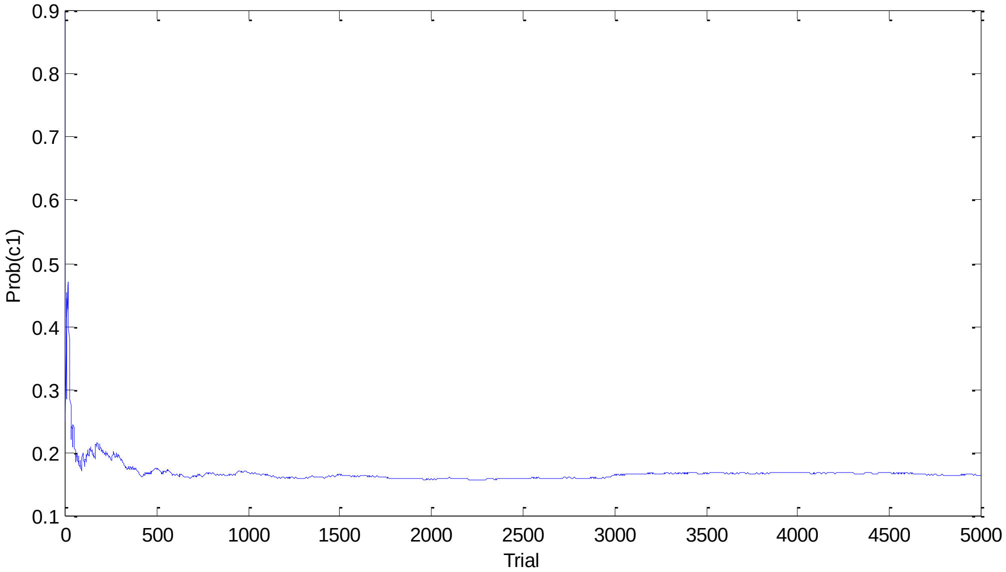
Probability of Blue = P(Blue) = ¼

→ Prob. Can be estimated by Area calculation
→ Prob. Can be estimated by Particles



Probability Can be Estimated by Monte Carlo Method

- Probability of 1 from Dice Throwing
  - P(C1)　　= 1/6
  - Test5.m



- After over 5000 trials, P(c1) converges 0.167.
- Prob. Estimation requires over than 5000~10,000 trials

Probabilistic Approach for State Estimation

- Remind that our goal is,

$$P(x_k \mid z_{1:k})$$

  - From the history of observation, $Z_{1:k}$, We want Xk estimation!

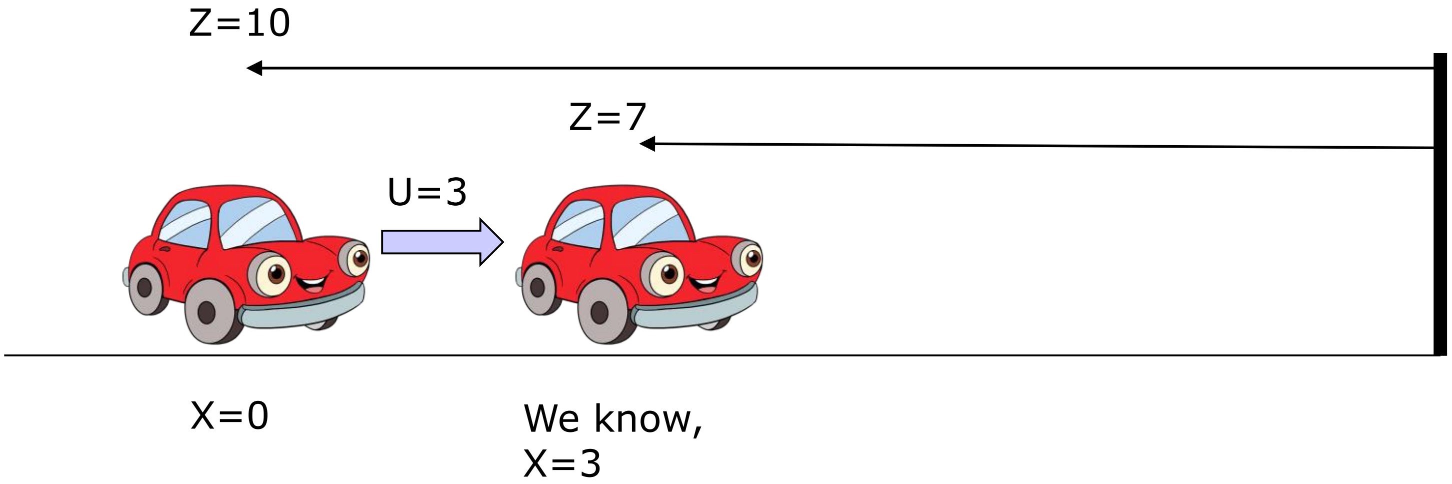
- Probability is estimated by many Trials as in Monte Carlo.

- However, we want more and more Trials.

→ How to reduce the number of trials?

→ Particle Filter with Importance Sampling

X with State Model
Z with Observation Model



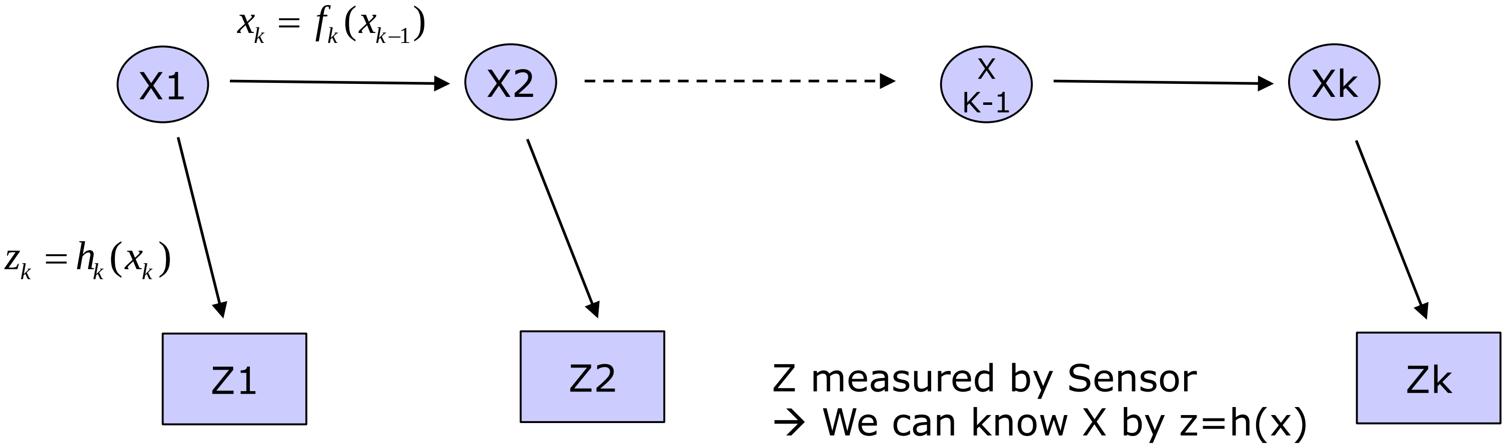
- X is a State from System Dynamics.
- Z is the measured value(from sensor)

X with State Model
Z with Observation Model

State Model

$$x_k = f_k(x_{k-1})$$

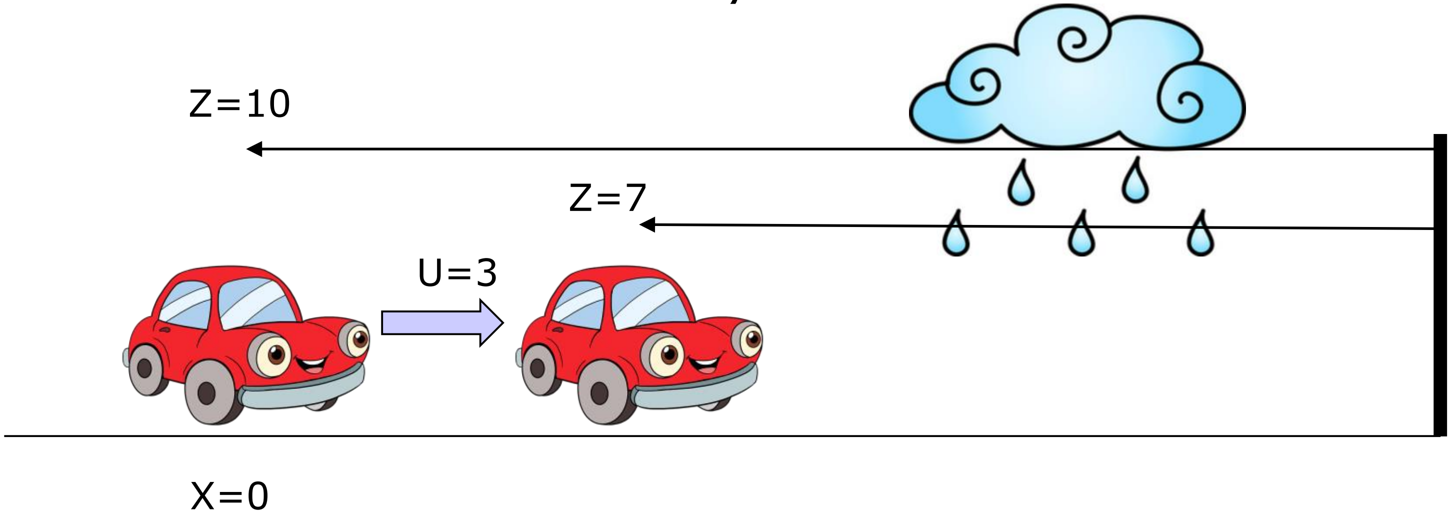
$$z_k = h_k(x_k)$$



Z measured by Sensor
→ We can know X by z=h(x)

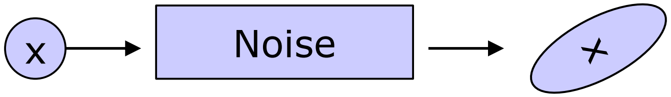
If there are noises, How the system changes?

In a NOISY World,
We DON'T Know **X**, but DO Know **Z**



1. We cannot believe Z=7 because there is NOISE
2. U=3 in a rainy day causes sliding.
3. So, how to estimate X=?

- In most cases, We DO NOT Know X
- We estimate X by measurement Z
- Also, there are noise. We model only Noise Variance.



We DON'T Know **X**, but DO Know **Z**

Red : Unknowns
Black: Knowns

Actual State Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

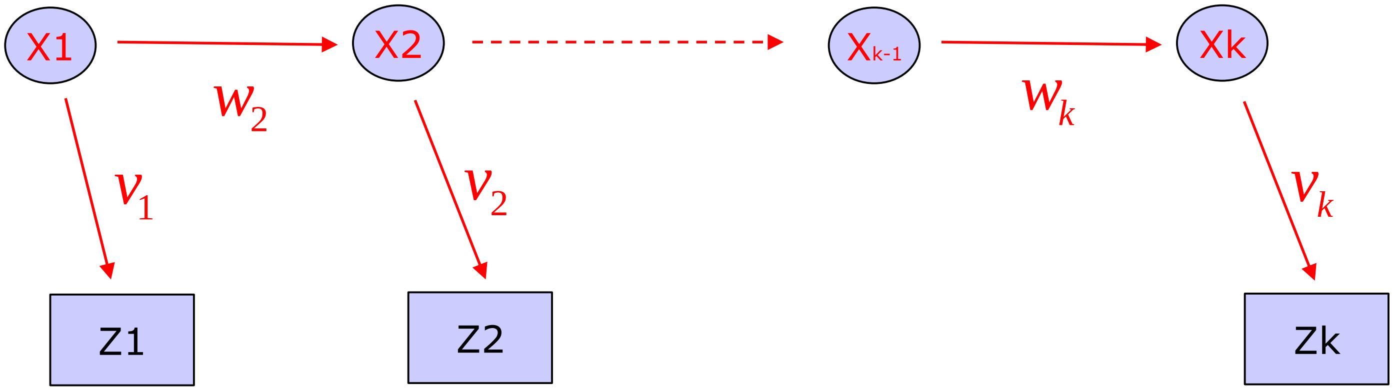
$$\xrightarrow[w_k, v_k]{\text{We don't know noise}}$$

Estimation

$$\hat{x}_k = f_k(\hat{x}_{k-1})$$

$$\hat{z}_k = h_k(\hat{x}_k)$$

X =?



Z measured by Sensor

We DON'T Know X, but DO Know Z

Red : Unknowns
Black: Knowns

Actual State Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

We don't know noise

$$w_k, v_k$$

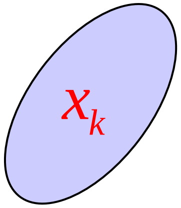
Estimation

$$\cancel{\hat{x}_k = f_k(\hat{x}_{k-1})}$$

$$\hat{z}_k = h_k(\hat{x}_k)$$

$$x_k^m = f_k(x_{k-1}^m, w_k)$$

$$x_k = f_k(x_{k-1}, w_k) =$$

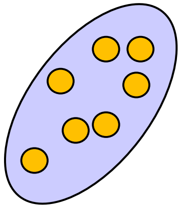


Xk is a distribution.
Our model Wk does NOT tell where Xk is.

$$x_k^m = f_k(x_{k-1}^m, w_k) = \text{●}$$

x_k^m is a particle with Random Noise Wk.

$$f_k(\text{●}\quad, w_k) =$$



Wk for a particle Generate Distribution.

Red : Unknowns
Black: Knowns

Step 1. PF State Transition

Actual State Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

We don’t know noise

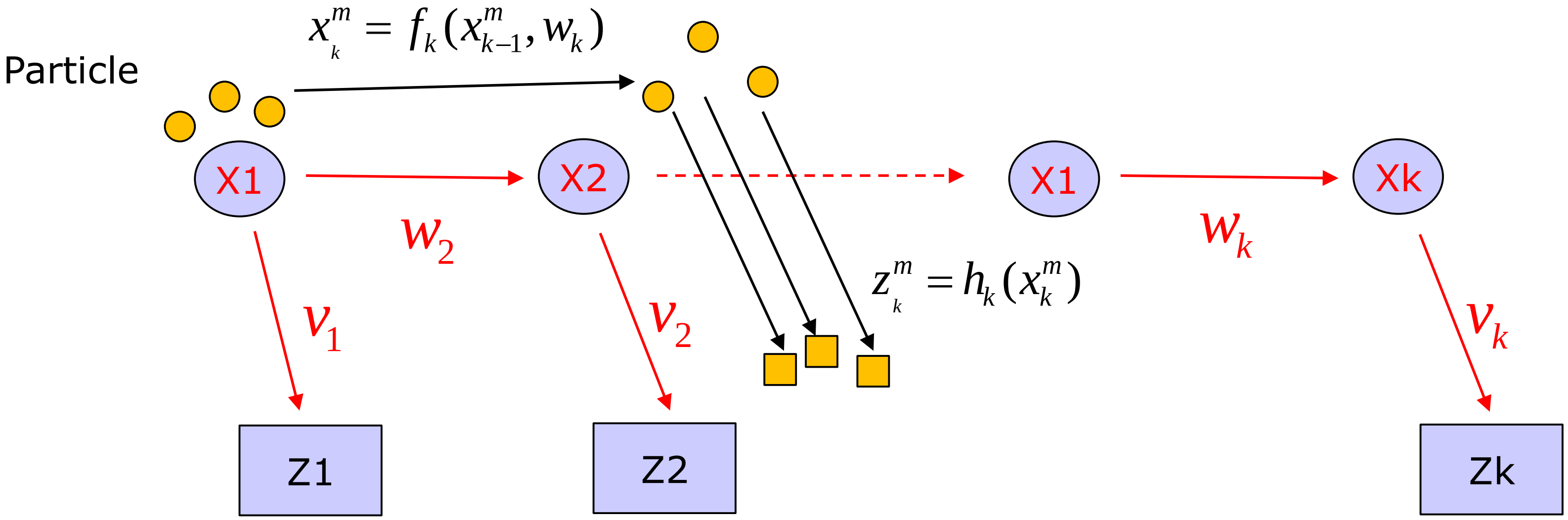
w_k, v_k

Estimation

$$\cancel{\hat{x}_k = f_k(\hat{x}_{k-1})}$$

$$\hat{z}_k = h_k(\hat{x}_k)$$

$$x_k^m = f_k(x_{k-1}^m, w_k)$$



Red : Unknowns
Black: Knowns

Step 2. PF Sampling Weight

Actual State Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

We don't know noise

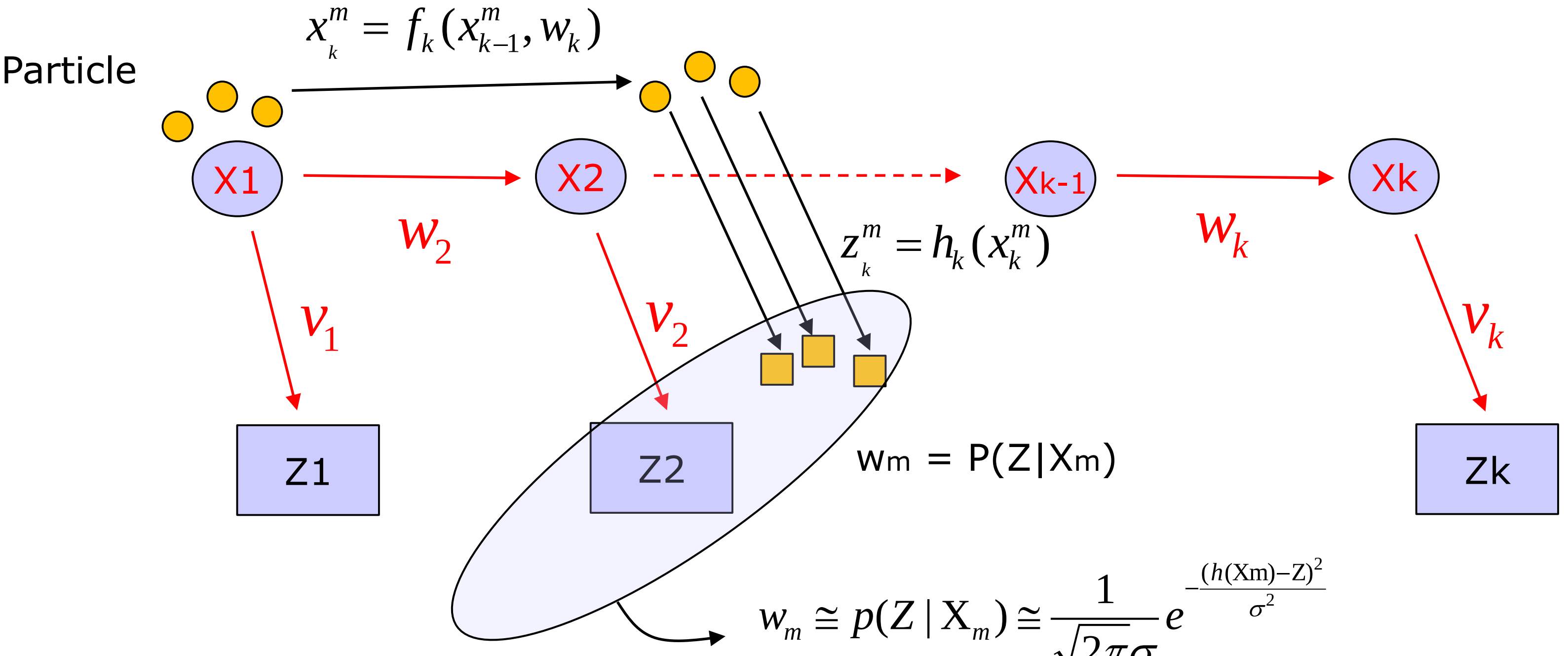
$$\xrightarrow[w_k,\, v_k]{}$$

Estimation

$$\cancel{\hat{x}_k = f_k(\hat{x}_{k-1})}$$

$$\hat{z}_k = h_k(\hat{x}_k)$$

$$x^m_k = f_k(x^m_{k-1}, w_k)$$



$$w_m \cong p(Z \mid \mathrm{X}_m) \cong \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(h(\mathrm{Xm})-\mathrm{Z})^2}{\sigma^2}}$$

Red : Unknowns
Black: Knowns

Step 3. PF Resampling

Actual State Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

We don't know noise

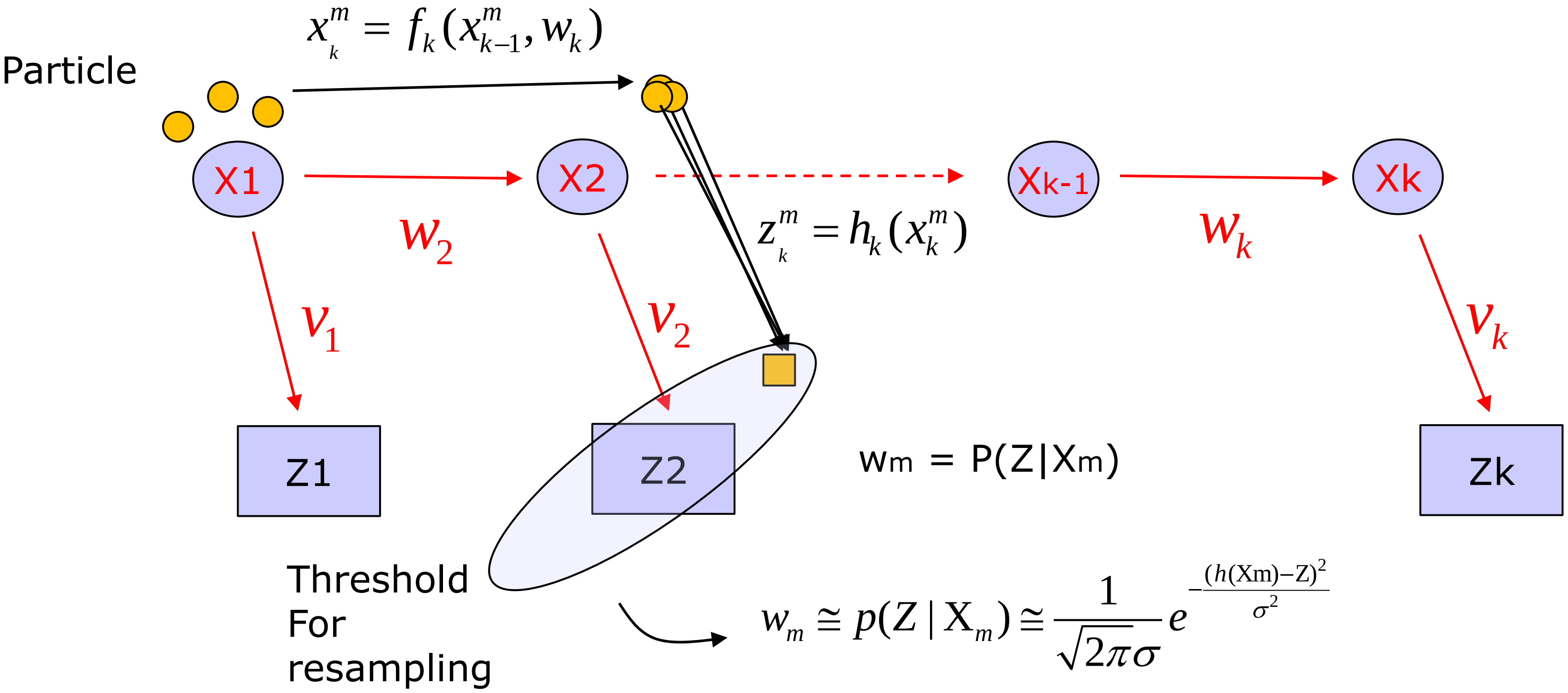
$$w_k, v_k$$

Estimation

$$\cancel{\hat{x}_k = f_k(\hat{x}_{k-1})}$$

$$\hat{z}_k = h_k(\hat{x}_k)$$

$$x^m_k = f_k(x^m_{k-1}, w_k)$$



$$x^m_k = f_k(x^m_{k-1}, w_k)$$

$$w_2 \quad v_1 \quad v_2 \quad w_k \quad v_k$$

$$z^m_k = h_k(x^m_k)$$

wm = P(Z|Xm)

Threshold
For
resampling

$$w_m \cong p(Z \mid X_m) \cong \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(h(Xm)-Z)^2}{\sigma^2}}$$

Red : Unknowns
Black: Knowns

Step 3. PF Resampling

Actual State Model

$$x_k = f_k(x_{k-1}, w_k)$$

$$z_k = h_k(x_k, v_k)$$

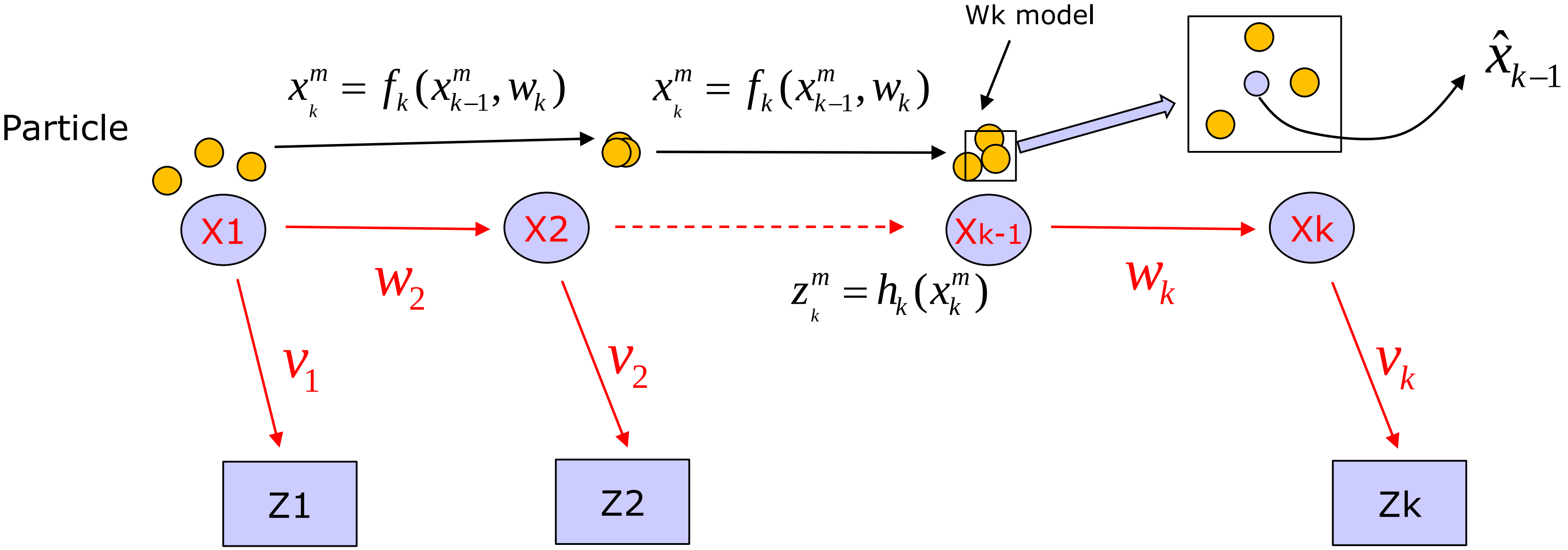
We don't know noise

w_k, v_k

Estimation

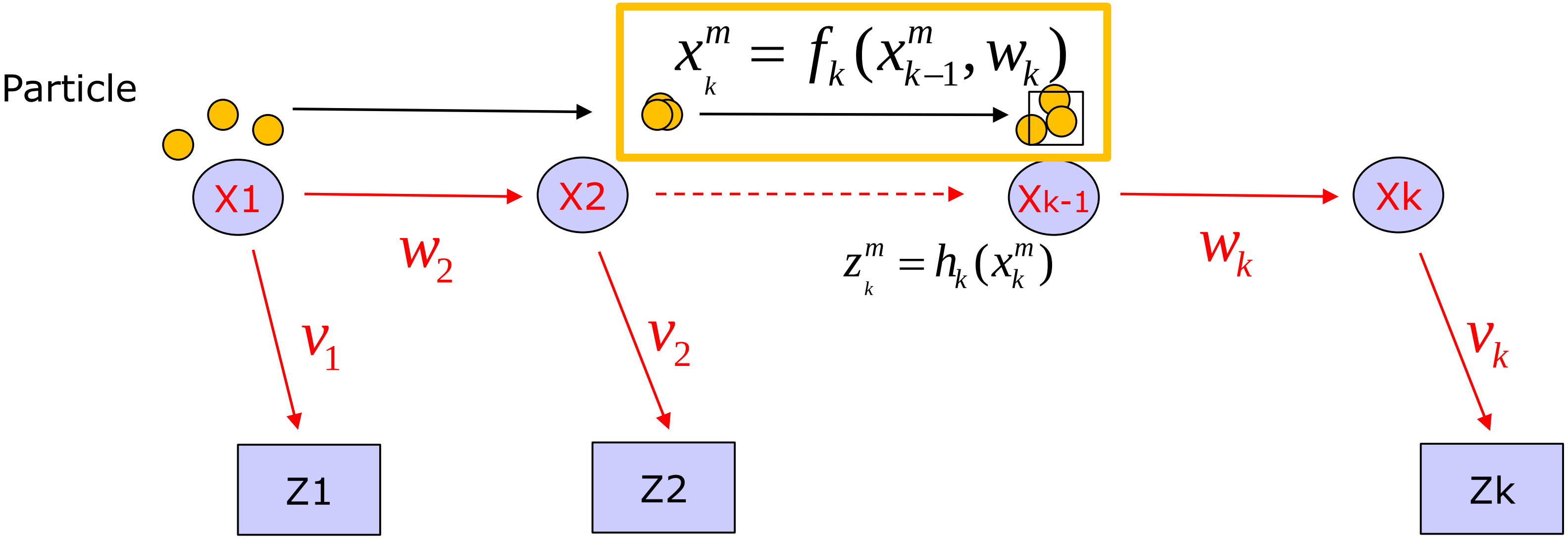
$$\cancel{\hat{x}_k = f_k(\hat{x}_{k-1})}$$

$$\hat{z}_k = h_k(\hat{x}_k)$$

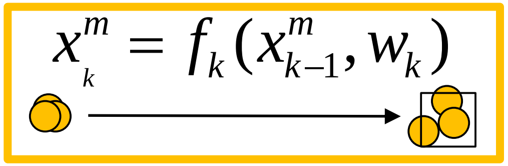


Resampling Question???

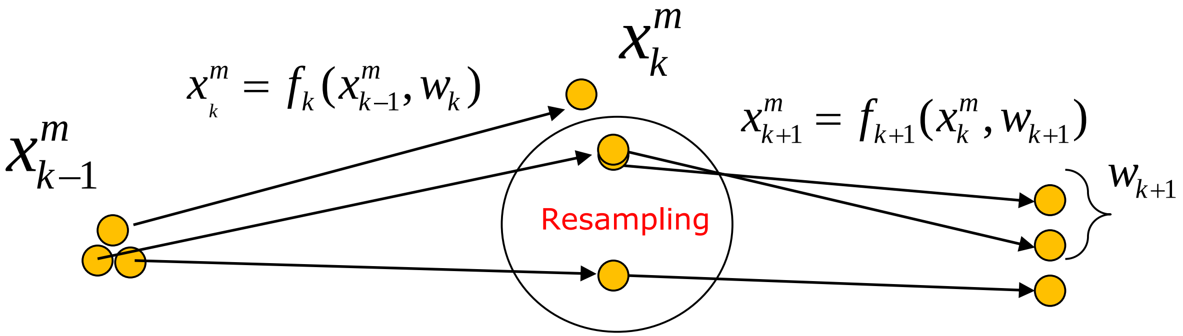
- After all, x_k^m are same?
  - In most cases, they are NOT same. Why?



Resampling : A Magical Way



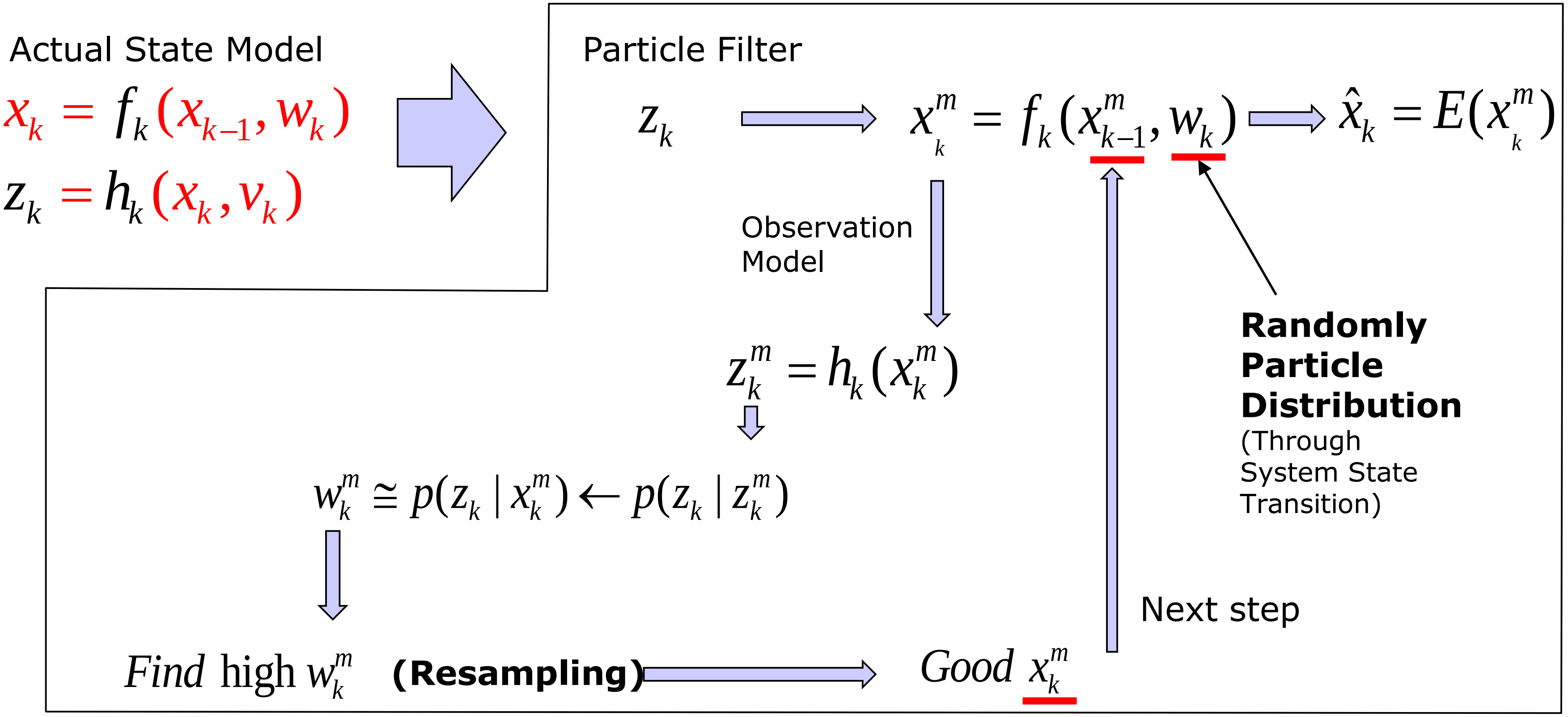
Possibly, Particles can be Scattered



System Noise wk possibly enables Particles to be scattered.

- Resampling is the Essential Point of Particle Filter
    - After system dynamics, the variance of particles distribution changes → Particles variance is not constant.

PF Summary
Random Particle Generation + Resampling



Actual State Model

$x_k = f_k(x_{k-1}, w_k)$

$z_k = h_k(x_k, v_k)$

Particle Filter

$z_k \Rightarrow x^m_k = f_k(\underline{x^m_{k-1}}, \underline{w_k}) \Rightarrow \hat{x}_k = E(x^m_k)$

Observation Model

$z^m_k = h_k(x^m_k)$

$w^m_k \cong p(z_k \mid x^m_k) \leftarrow p(z_k \mid z^m_k)$

Randomly Particle Distribution (Through System State Transition)

Next step

Find high w^m_k **(Resampling)** $\Rightarrow$ *Good* $\underline{x^m_k}$

Then, we should understand the theoretical features of Importance Sampling Weight

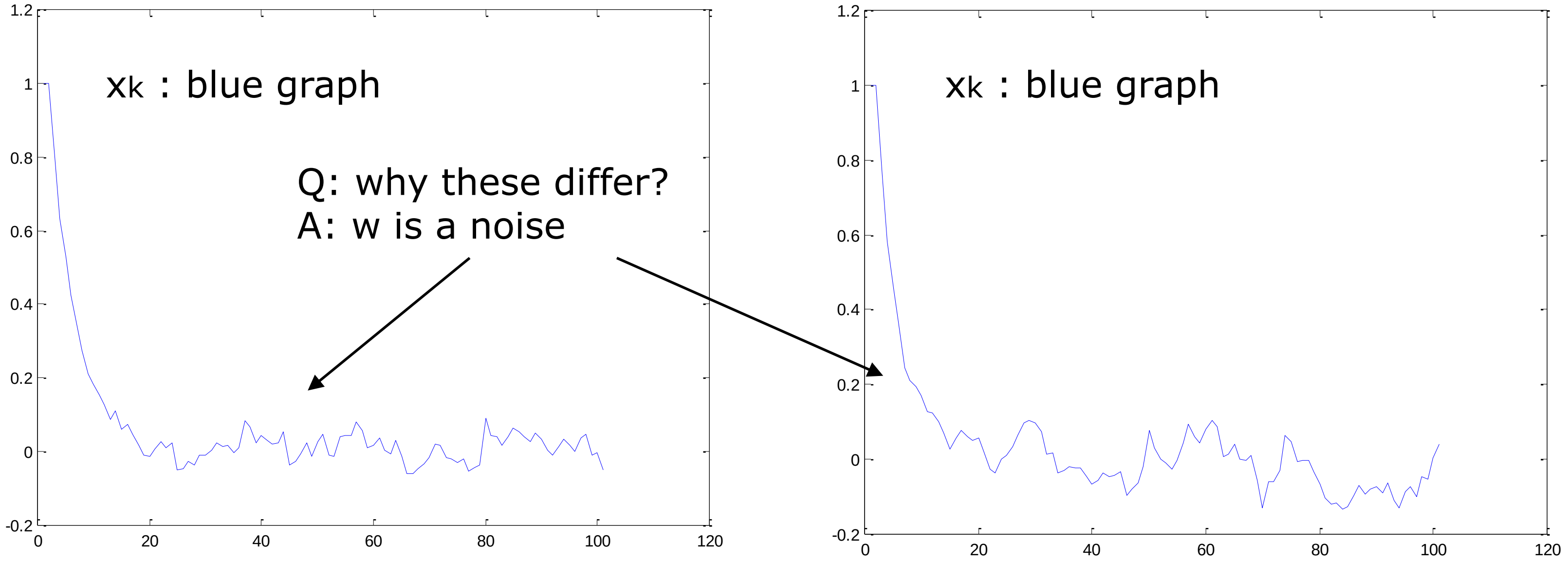
$$\text{Importance Sampling Weight} = w_m \cong p(Z \mid \mathrm{X}_m) \cong \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(h(\mathrm{Xm})-\mathrm{Z})^2}{2\sigma^2}}$$

PF Example (TestPF.m)

- System dynamics

$$\dot{x} = -20x - \sin(px) + w$$

$$w \sim N(\mu,\sigma) = N(0,\sqrt{10})$$



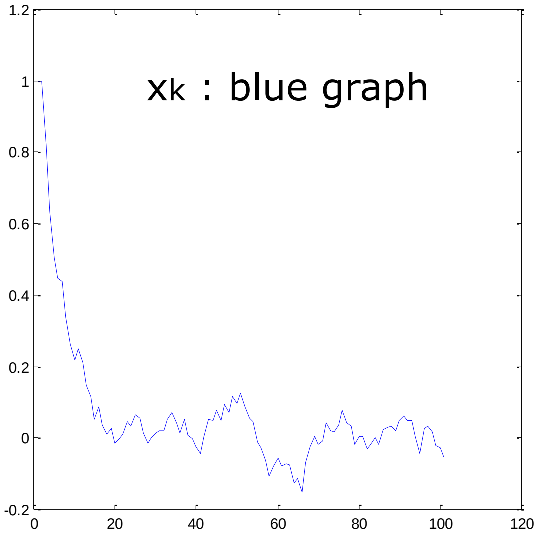
Q: why these differ?
A: w is a noise

PF Example (TestPF.m)

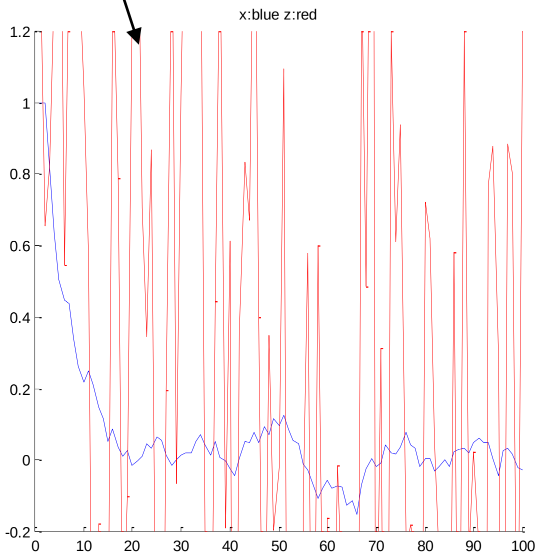
- System dynamics and Measurement

$$\dot{x} = -20x - \sin(px) + w \qquad w \sim N(\mu,\sigma) = N(0,\sqrt{10}^{2})$$

$$z = x + v \qquad v \sim N(\mu,\sigma) = N(0,1)$$



x_k : blue graph



z_k : red graph

Simulation for TestPF.m

- Discretization

$$\dot{x} = \frac{x_{k+1} - x_k}{\Delta t} = -20x_k - \sin(px_k) + w_k$$

$$\rightarrow x_{k+1} = x_k - 20\Delta t x_k - \Delta t \sin(px_k) + \Delta t w_k$$

$$\rightarrow z_k = x_k + v_k$$

- W,V simulation(Normal distribution=“randn” in Matlab)

$$p(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \quad \Rightarrow Z \sim N(0,1) = randn$$

$$x = \sigma z$$

$$\Rightarrow X \sim N(0, \sigma^2) = \sigma \bullet randn$$

$$w_k = \sqrt{W} randn$$

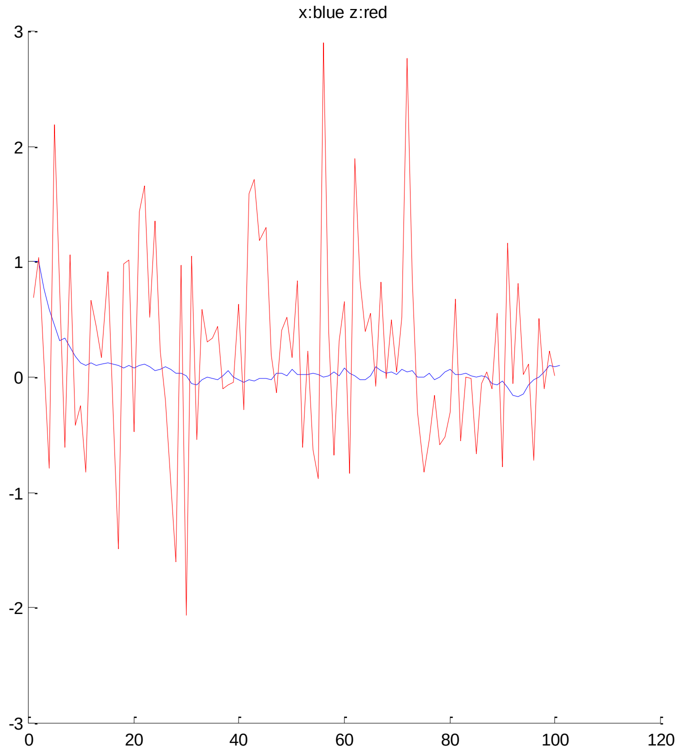
$$v_k = \sqrt{V} randn$$

Simulation Code for TestPF.m

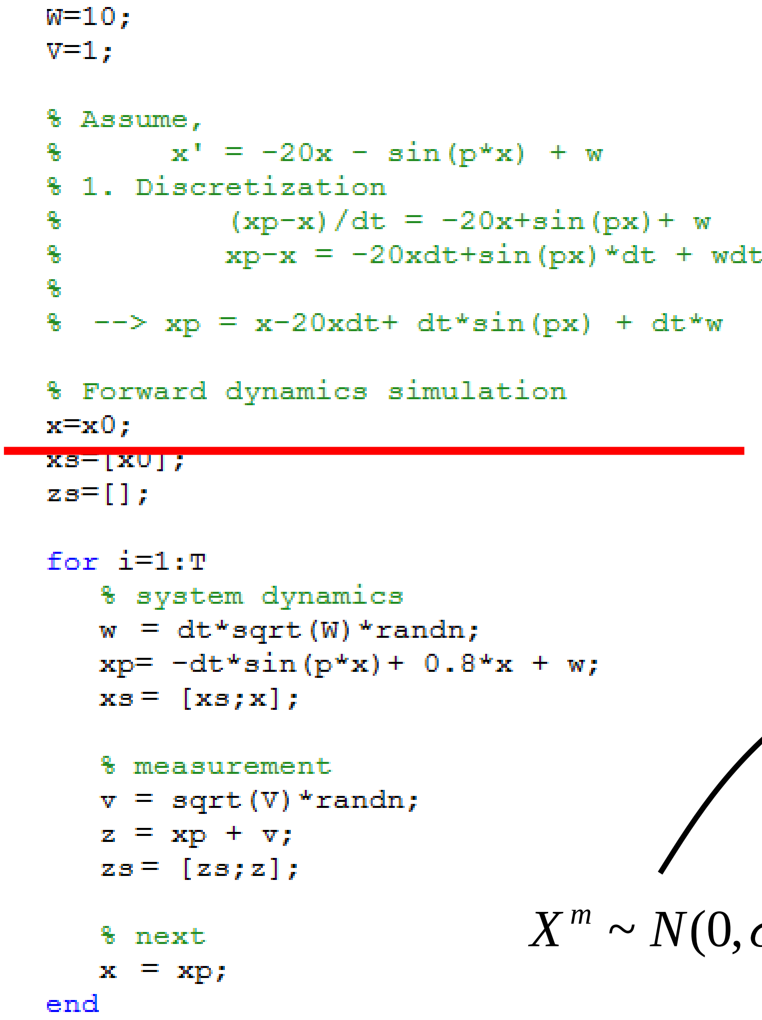
```
W=10;
V=1;

% Assume,
%       x' = -20x - sin(p*x)  + w
% 1. Discretization
%           (xp-x)/dt = -20x+sin(px)+ w
%           xp-x = -20xdt+sin(px)*dt + wdt
%
%  --> xp = x-20xdt+ dt*sin(px)  + dt*w

% Forward dynamics simulation
x=x0;
xs=[x0];
zs=[];

for i=1:T
   % system dynamics
   w  = dt*sqrt(W)*randn;
   xp= -dt*sin(p*x)+ 0.8*x + w;
   xs= [xs;x];

   % measurement
   v = sqrt(V)*randn;
   z = xp + v;
   zs= [zs;z];

   % next
   x = xp;
end
```

$$x_{k+1} = (1 - 20\Delta t)x_k - \Delta t \sin(px_k) + \Delta t w_k$$

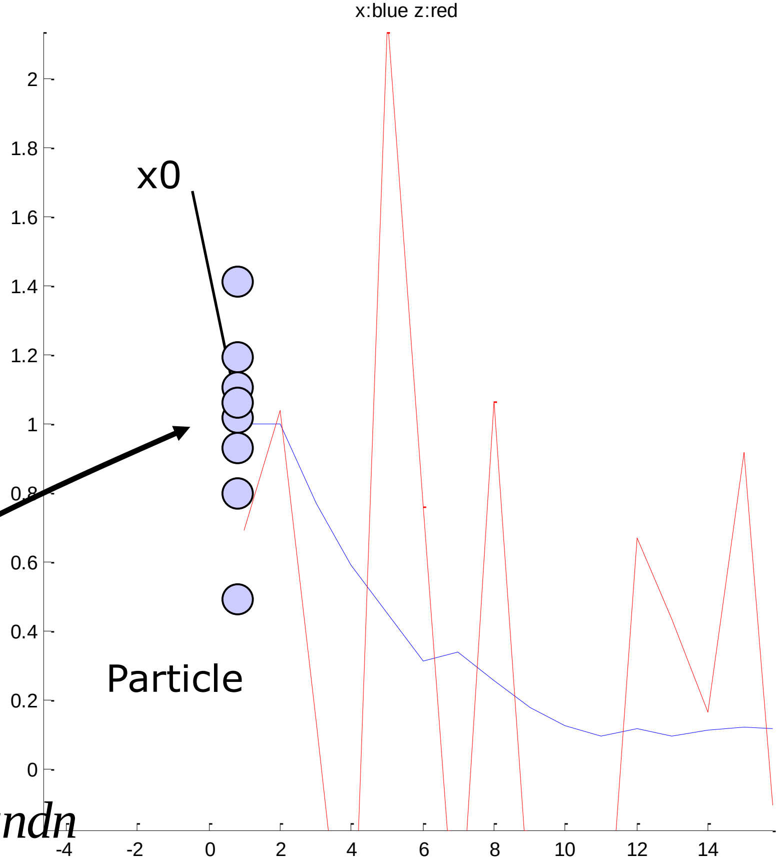
$$z_k = x_k + v_k$$



0. Particle Generation at an Initial State, x_0



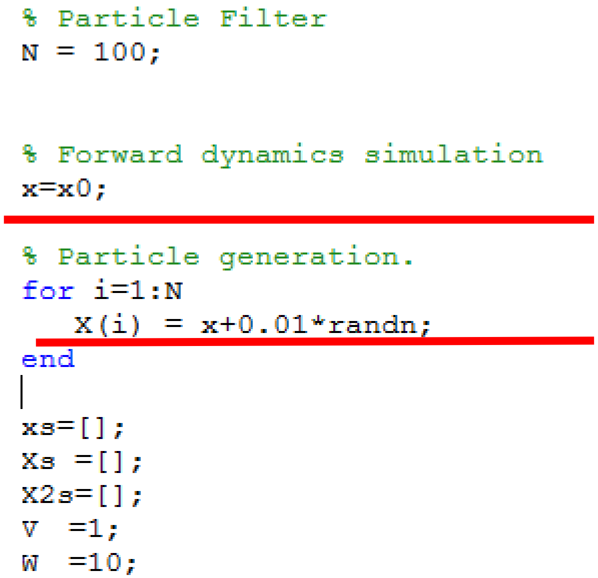
```
W=10;
V=1;

% Assume,
%        x' = -20x - sin(p*x) + w
% 1. Discretization
%            (xp-x)/dt = -20x+sin(px)+ w
%            xp-x = -20xdt+sin(px)*dt + wdt
%
%  --> xp = x-20xdt+ dt*sin(px) + dt*w

% Forward dynamics simulation
x=x0;
xs=[x0];
zs=[];

for i=1:T
    % system dynamics
    w = dt*sqrt(W)*randn;
    xp= -dt*sin(p*x)+ 0.8*x + w;
    xs= [xs;x];

    % measurement
    v = sqrt(V)*randn;
    z = xp + v;
    zs= [zs;z];

    % next
    x = xp;
end
```



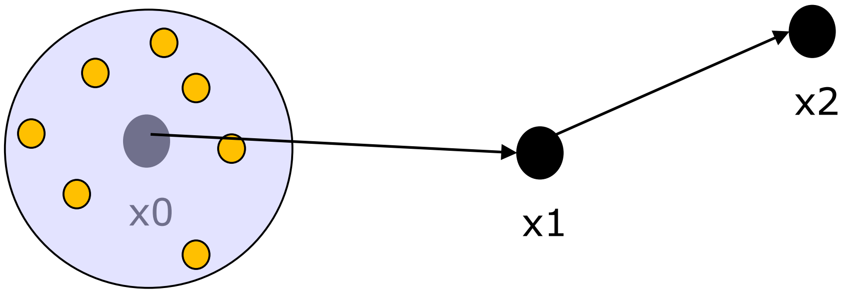
$$X^m \sim N(0,\sigma) = \sigma \bullet randn$$

0. Particle Generation at an Initial State, x_0

(testPF2.m)



```
% Particle Filter
N = 100;


% Forward dynamics simulation
x=x0;

% Particle generation.
for i=1:N
   X(i) = x+0.01*randn;
end
|
xs=[];
Xs =[];
X2s=[];
V  =1;
W  =10;
```



● Particle X in Matlab → Xm in Eq.

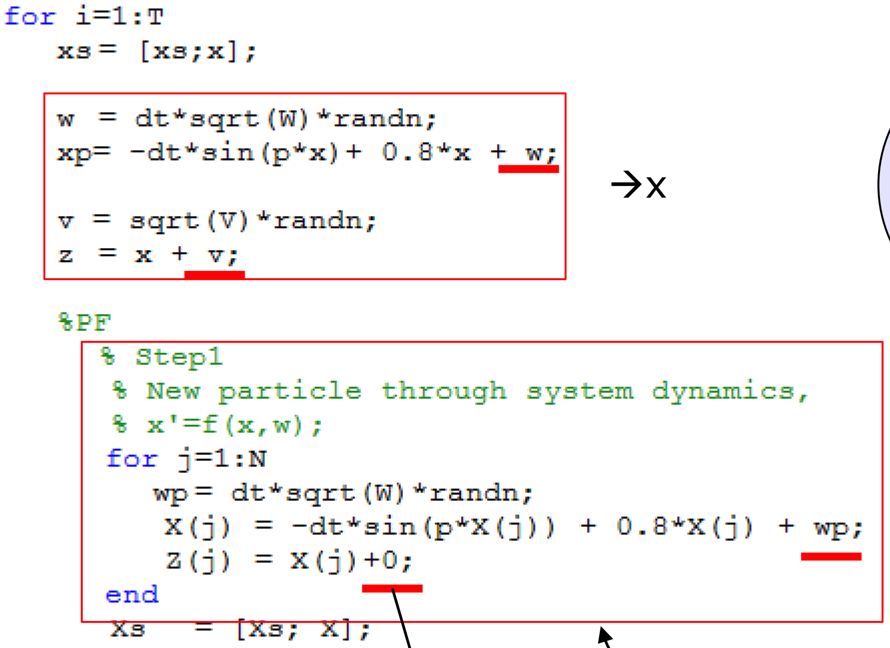
$$x_{k+1} = (1 - 20\Delta t) x_k - \Delta t \sin(p x_k) + \Delta t w_k$$

$$z_k = x_k + v_k$$

x: actual state
X: particle

X generation with sigma=0.01

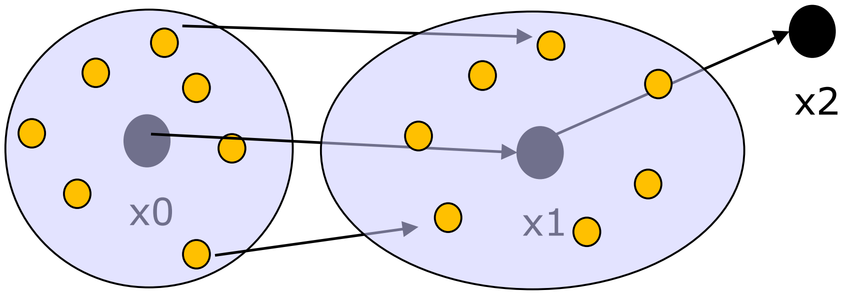
1. Particle Virtually Moves by System Dynamics



```
for i=1:T
   xs= [xs;x];

   w  = dt*sqrt(W)*randn;
   xp= -dt*sin(p*x)+ 0.8*x + w;

   v  = sqrt(V)*randn;
   z  = x + v;

  %PF
    % Step1
     % New particle through system dynamics,
     % x'=f(x,w);
    for j=1:N
       wp= dt*sqrt(W)*randn;
        X(j) = -dt*sin(p*X(j)) + 0.8*X(j) + wp;
        Z(j) = X(j)+0;
    end
     Xs   = [Xs; X];
```

→x

→X



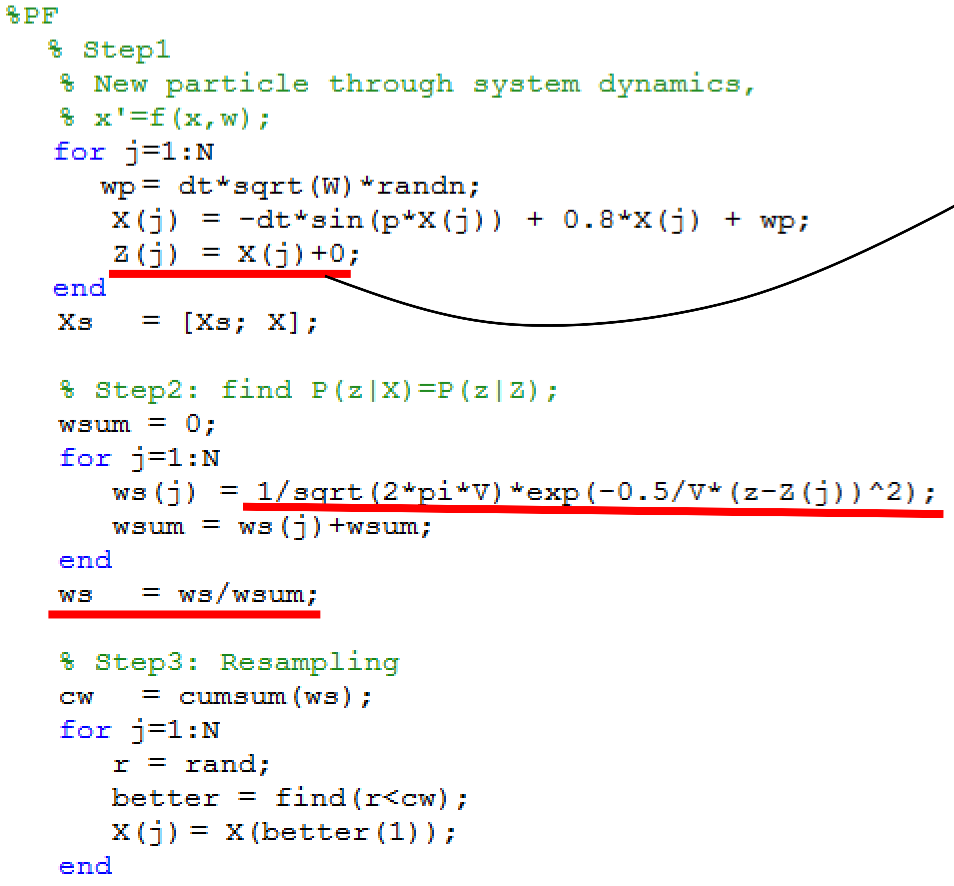
● Particle X in Matlab → Xm in Eq.

$$x^m{}_{k+1} = (1-20\Delta t)x^m{}_k - \Delta t\sin(px^m{}_k) + \Delta t w^m{}_k$$

$$z^m{}_k = x^m{}_k$$

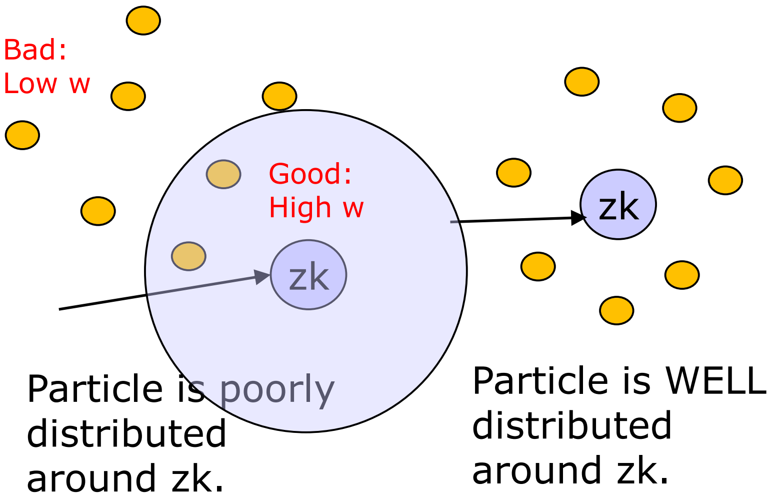
wp: particle has noisy system dynamics
vp=0: Particle state is a virtual one.
We measure it WITHOUT noise.

2. Weight Calculation



```
%PF
  % Step1
   % New particle through system dynamics,
   % x'=f(x,w);
  for j=1:N
     wp= dt*sqrt(W)*randn;
      X(j) = -dt*sin(p*X(j)) + 0.8*X(j) + wp;
      Z(j) = X(j)+0;
  end
   Xs   = [Xs; X];

   % Step2: find P(z|X)=P(z|Z);
   wsum = 0;
   for j=1:N
      ws(j) = 1/sqrt(2*pi*V)*exp(-0.5/V*(z-Z(j))^2);
      wsum = ws(j)+wsum;
   end
   ws   = ws/wsum;

   % Step3: Resampling
   cw   = cumsum(ws);
   for j=1:N
      r = rand;
      better = find(r<cw);
      X(j) = X(better(1));
   end
```

- $P(z|x_m)=P(z|z_m)$



Particle is poorly distributed around zk.

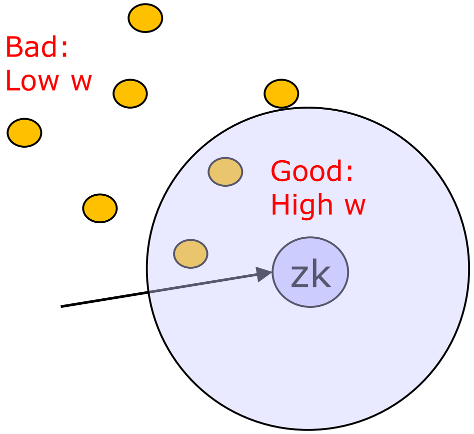
Particle is WELL distributed around zk.

Importance Sampling Weight=

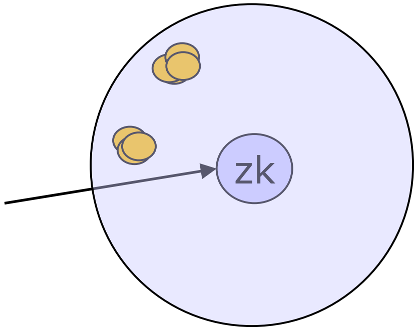
$$= w_m \cong p(Z \mid X_m) \cong \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(Z m-Z)^2}{2\sigma^2}}$$

3. Resampling

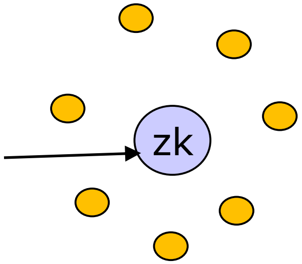
- Choose Good Particle(with high weight)



Particle is poorly distributed around zk.



Particle is well Resampled around zk.



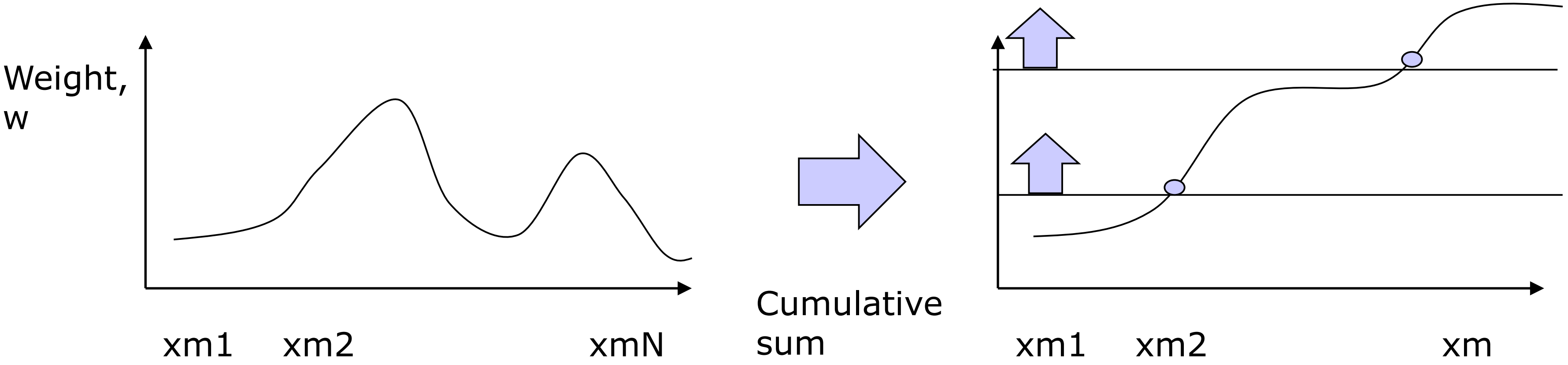
Particle is WELL distributed around zk.

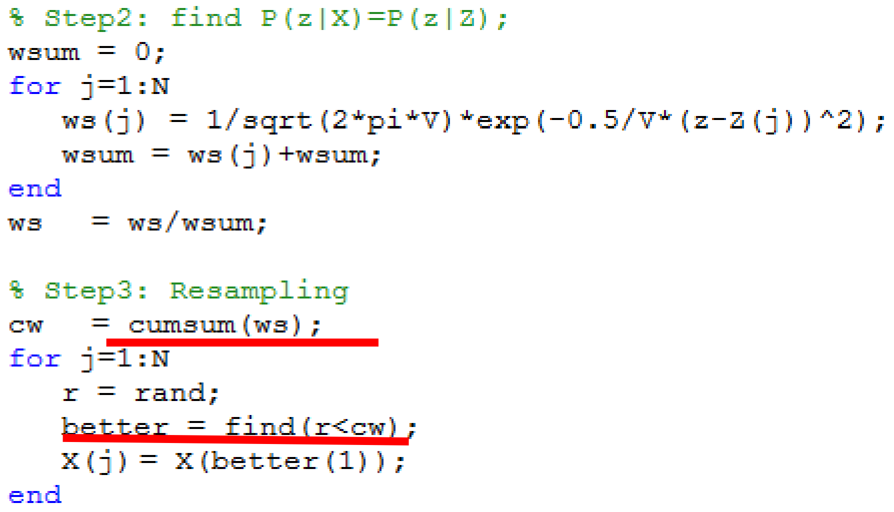
→ **No need for Resampling**

1. Resampling with Constant Number, N
2. Choose High Weight.
3. Most Particles converges into One particle, probably.

4. Goal : every has same or similar weight.

3. Resampling in Matlab

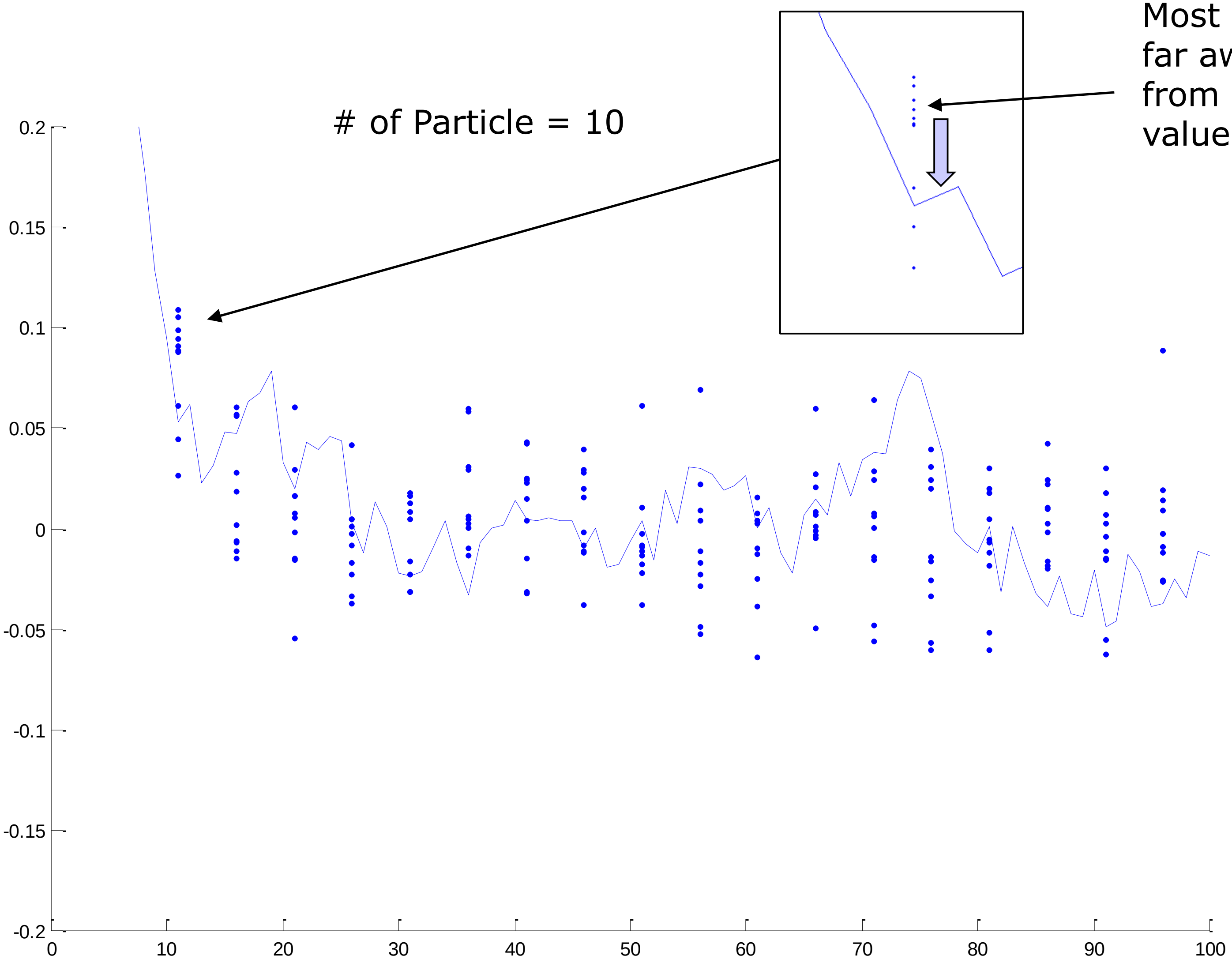




```
% Step2: find P(z|X)=P(z|Z);
wsum = 0;
for j=1:N
   ws(j) = 1/sqrt(2*pi*V)*exp(-0.5/V*(z-Z(j))^2);
   wsum = ws(j)+wsum;
end
ws   = ws/wsum;

% Step3: Resampling
cw   = cumsum(ws);
for j=1:N
   r = rand;
   better = find(r<cw);
   X(j) = X(better(1));
end
```

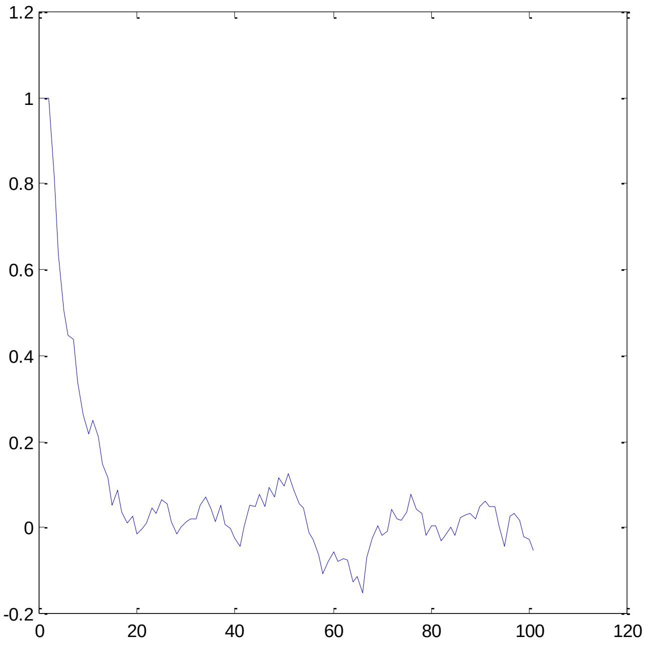
- Randomly choose higher Weight

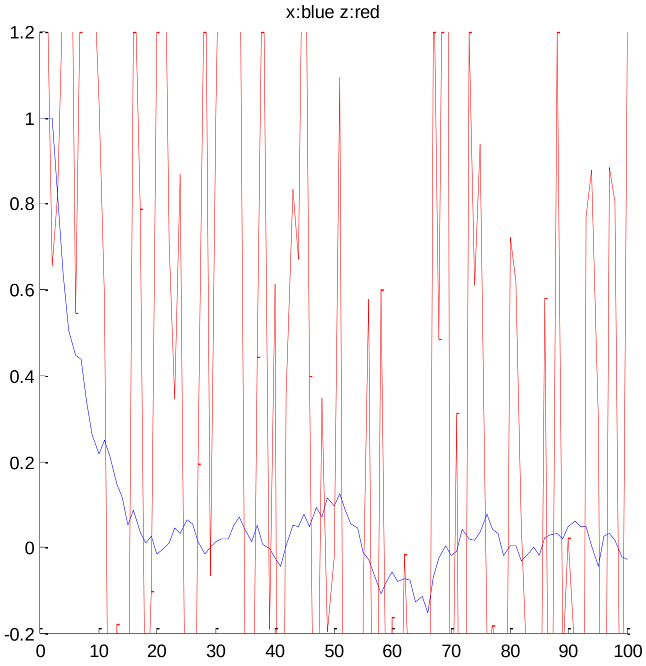
Result: Without Resampling



Most particles are far away from an actual value

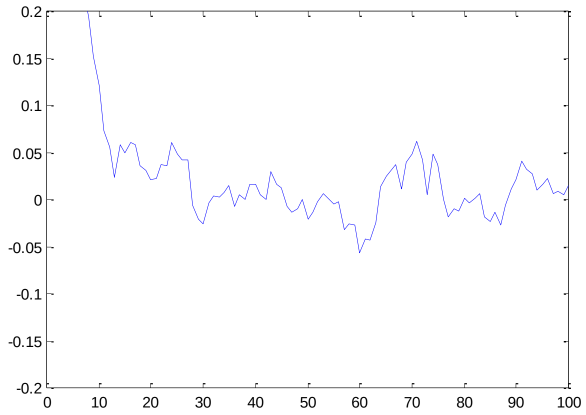
Remind the Result of Measurement

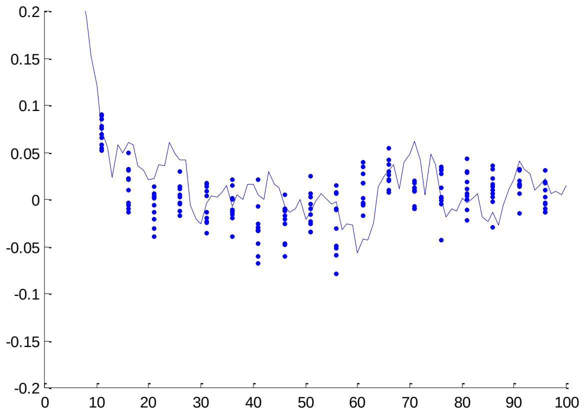


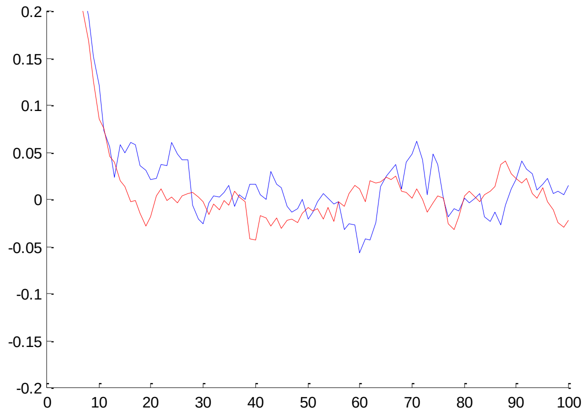


Much Noisy..

TestPF2 with N=10

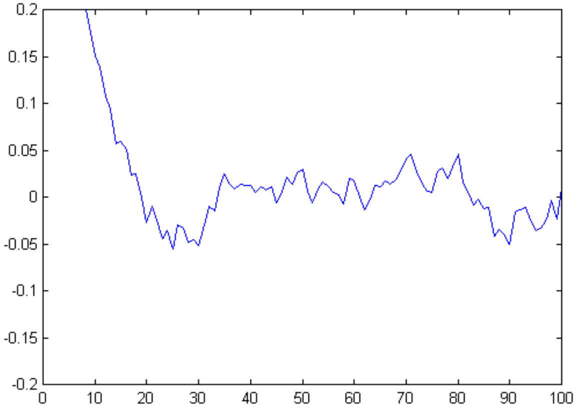


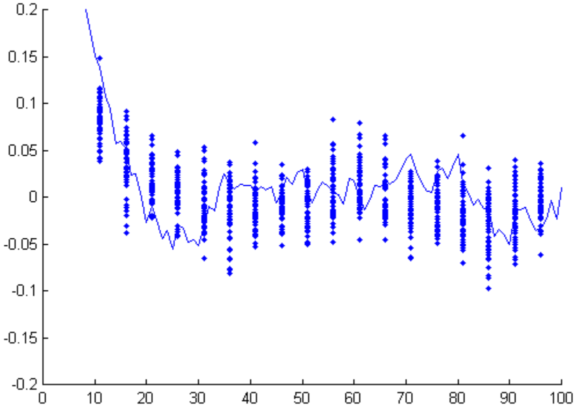


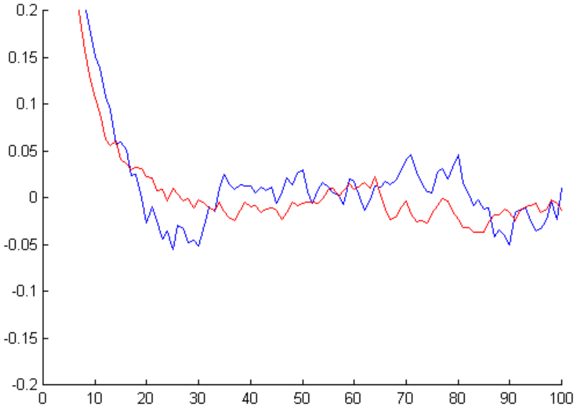


Blue: Actual
Red: Estimation

TestPF2 with N=50

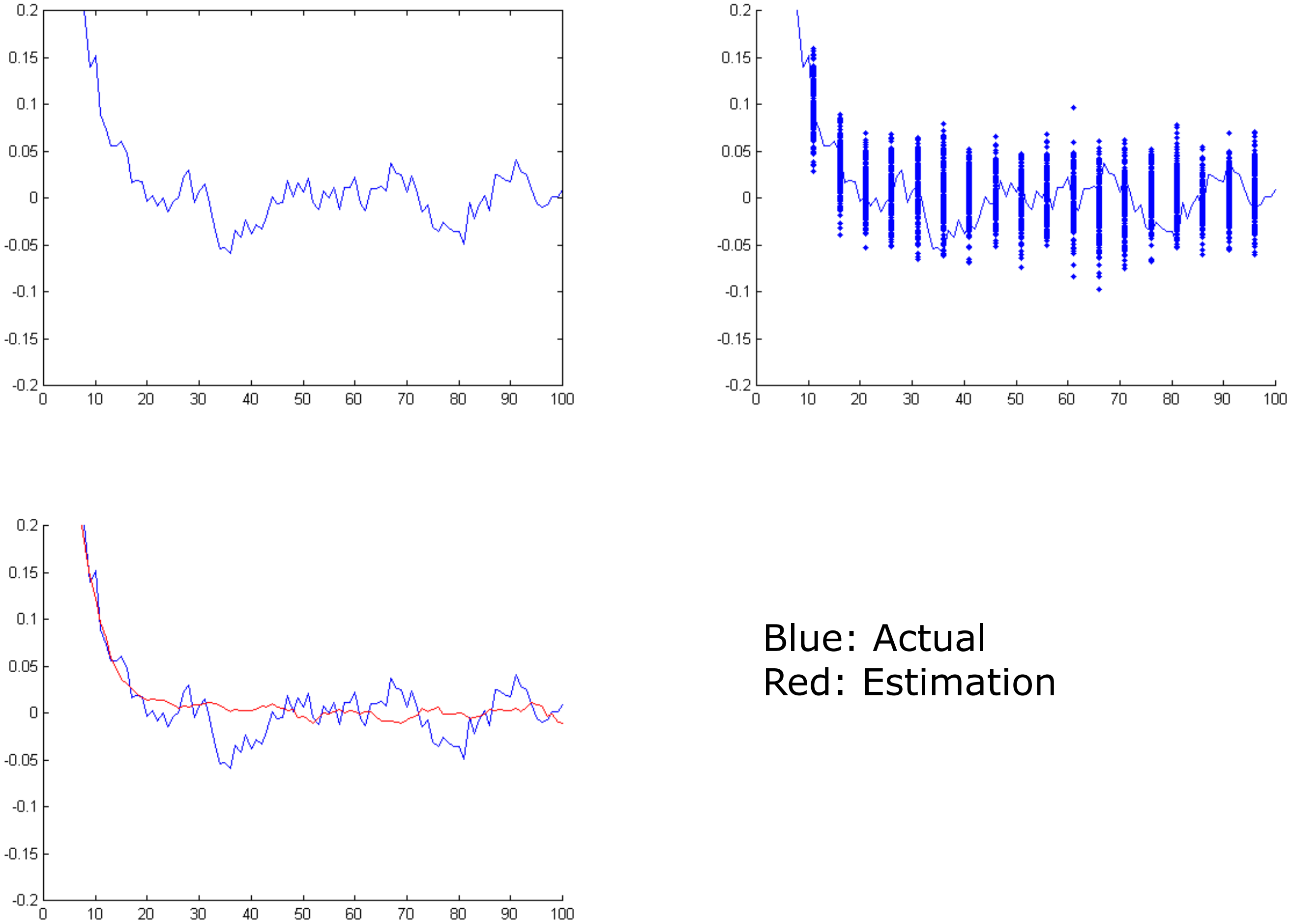






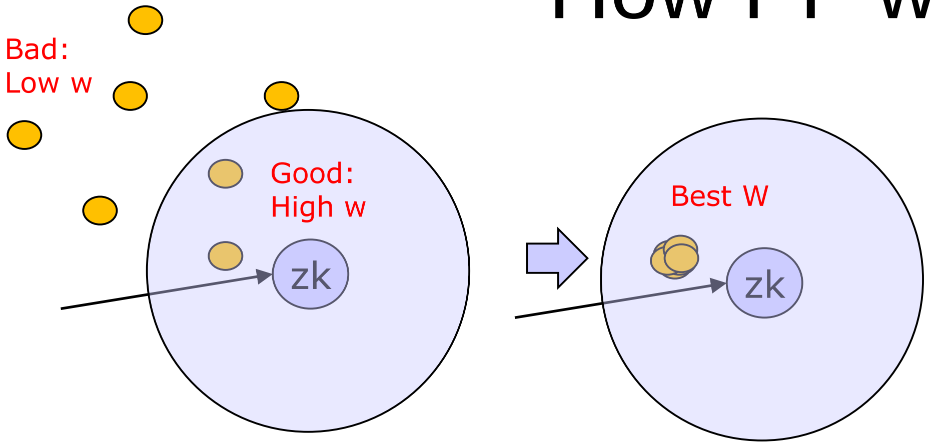
Blue: Actual
Red: Estimation

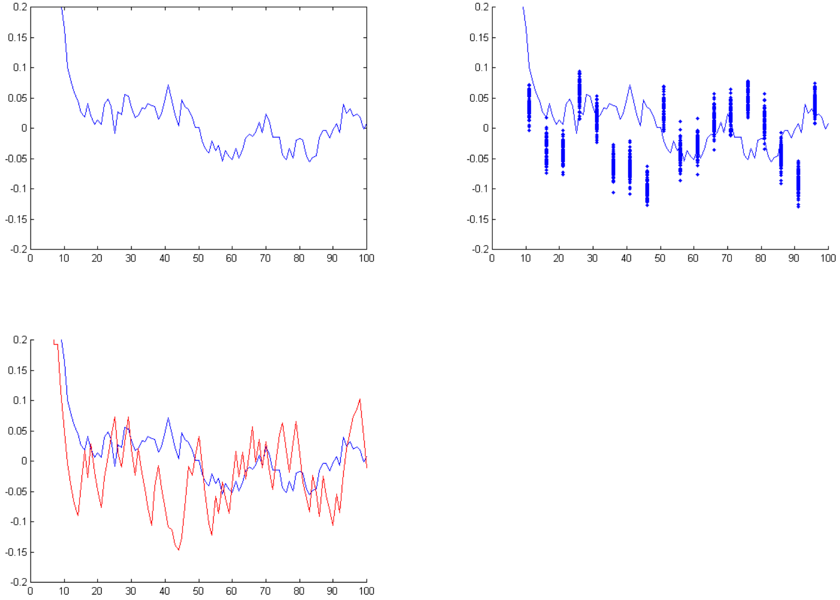
TestPF2 with N=200



Blue: Actual
Red: Estimation

Question:
When we Choose the Best Weight, How PF works?



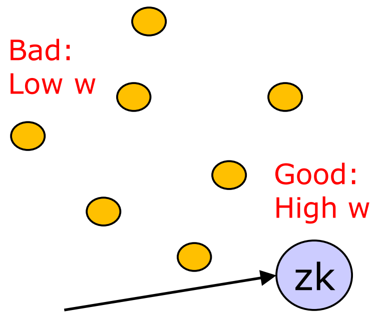


TestPF3.m

Q: Why the results show Poor Performance?

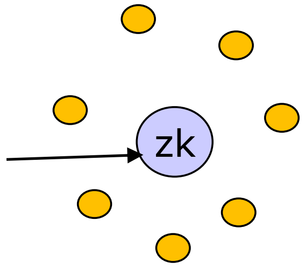
→ Best Weight can be biased.
→ Best Weight May be NOT the best Distribution.

Extra Concept: Weight Measurement of Importance Sampling



Not Good
→ Weight(based Resampling) is needed

W=[0.1, 0.3, 0.01]



Good

W=[0.1, 0.1, 0.1]

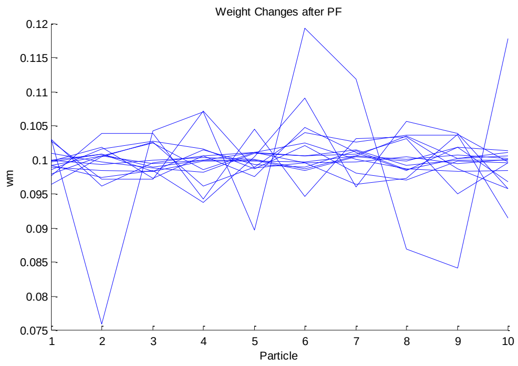
- Variance of Weight

$$N_{eff} = \frac{1}{\sum_m {w_m}^2}$$

if (Neff<N_{th}) Resampling

otherwise, No resampling

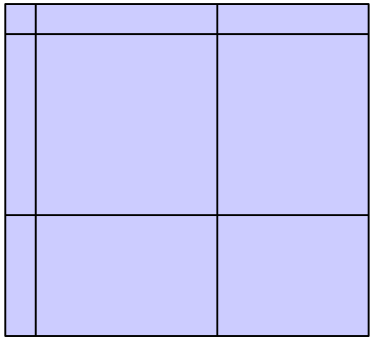
TestPF4.m



Weight variance becomes small.

Why PF becomes More Dominant than KF

- KF requires O(N2) operation (10 Legends → 22x22 matrix)



- Particle Filter O(MLogN) operation
  - Most SLAM methods are based on PF concept.
  - All cleaning robot uses PF-based SLAM.

- Low Dimension : KF>PF
- Higher Dimension : PF>>>KF