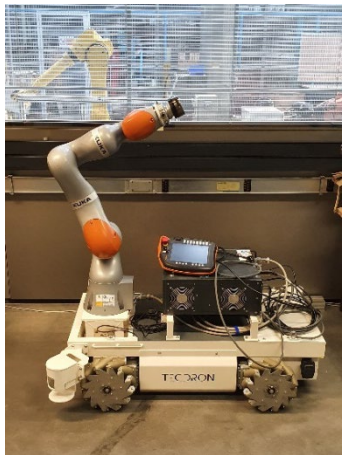
ESTIMATION THEORY AND PARAMETRIC IDENTIFICATION

NON-LINEAR ESTIMATION THEORY



Mechbal N. / Guskov M. / Rebillat M.

*Laboratoire **PIMM**, UMR-CNRS*
Arts et Métiers ParisTech (ENSAM), Paris

OUTLINE TOPICS

There are 3 main Parts in this course:

- **Part 1** Estimation Theory – N. Mechbal – 6 lectures
  - Review of probability theory
  - Linear Estimation Theory – Deterministic and Bayesian methods
  - **Nonlinear estimation – EKF, UKF and PF**
- **Part 2** Parametric identification theory – N. Mechbal & M. Rebillat
  - Mathematical foundations of system identification – **NM/1 Lecture**
  - Parameter estimation - **NM /2 Lectures**
  - Sensors and Signal processing for identification – discrete time processing, frequency analysis, denoising - **MR/3 Lectures**
- **Part 3** Model identification of flexible manipulators – M. Guskov
  - Modeling of flexible manipulators - **MG/1Lectures**
  - Modeling for vibration analysis - **MG/1Lectures**
  - Application: Robotic machining - **MG/1Lectures**

SYLLABUS PART 1

- **Lecture 1:**
  - Course organization
  - Motivations
  - Introductive Examples
  - Review of Probability
- **Lecture 2:**
  - Estimation theory
  - Exercises
- **Lecture 3:**
  - Linear estimators: Least Square, Maximum Likelihood
  - Exercise
- **Lecture 4:**
  - Linear estimator: Kalman Filter
  - Exercise & Matlab class
- **Lecture 5:**
  - Nonlinear estimation: EKF and UKF
  - Exercises & Matlab class
- **Lecture 6:**
  - Nonlinear estimation: PF and RBPF
  - Exercise & Matlab class

"The probability of any event is the ratio between the value at which an expectation depending on the happening of the event thought to be computed, and the value of the thing expected upon its happening."

Thomas Bayes (1702-1761)

"Statistics is the art of never having to say you're wrong. Variance is what any two statisticians are at."

C. J. Bradeld.

NONLINEAR ESTIMATION: BAYESIAN FILTERING

OUTLINE

I. Introduction

II. Recursive Bayesian Estimation

III. Extend Kalman Filtering (EKF)

IV. Unscented Kalman Filtering (UKF)

V. Particle Filter (PF)

VI. Application

Nonlinear identification towards Nonlinear estimation

- In parameter estimation approach, we use a **random walk** variation for the parameters to rewrite a new state representation:

$$\underline{\theta}_{k+1} = \underline{\theta}_k + \underline{q}_k$$

where the parameter θ correspond to a stationary process with identity state transition matrix, driven by "*artificial*" process noise $\boldsymbol{q}$.

- *The system identification problem can now be cast as a nonlinear estimation problem.*

- Two estimation problems can be addressed:
  - ***Dual Estimation* problem**
  - ***Joint estimation problem***

Nonlinear identification towards Nonlinear estimation

- ***Dual Estimation* problem:** Separate state-space representation is used for the state and the parameters. Two estimators are run simultaneously for $\underline{x}$ and $\underline{\boldsymbol{\theta}}$.

  Iteratively, *the current estimate of $\underline{\theta}$ is used in the state filter as a given input and likewise the current estimate of the state is used in the parameter filter. The two coupled state representation used re given by:*

$$x_{k+1} = f(x_k, u_k, v_k, \hat{\theta}_k)$$
$$y_k = h(x_k, u_k, \omega_k, \hat{\theta}_k)$$

Estimated parameters

$$\theta_{k+1} = \theta_k + q_k$$
$$y_k = h(\hat{x}_k, u_k, \omega_k, \theta_k)$$

Estimated state

- ***Joint estimation problem:*** The strategy employed is rather well-known. The idea is ***to augment the states with the parameters into a new state vector.*** The new state to be estimated and model are:

$$\underline{x}^a(t) = \begin{bmatrix} \underline{x}(t) \\ \underline{\theta}(t) \end{bmatrix}$$

$$\underline{x}^a_{k+1} = \begin{bmatrix} \underline{x}_{k+1} \\ \underline{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} F(\underline{x}_k, \underline{\theta}_k, \underline{u}_k) \\ \underline{\theta}_k \end{bmatrix} + \begin{bmatrix} 0 \\ \underline{q}_k \end{bmatrix}$$

Markov Process

- **Definition:** *Markov property*. *A discrete time stochastic process* $\boldsymbol{X}(t)$ *is said to posses the Markov property if for* $t_1 < t_2 < t_3 < \cdots t_k < t_{k+1}$

$$p_{\underline{x}}\left(\underline{x}(t_{k+1})/\underline{x}(t_1), \underline{x}(t_2), \ldots, \underline{x}(t_k)\right) = p_{\underline{x}}\left(\underline{x}(t_{k+1})/\underline{x}(t_k)\right)$$

- This means that the realization of the process at time t_k **contains all information** about the past, which is necessary in order to calculate the future behavior of the process. Therefore, *if the present realization of the process is known, the future is independent of the past*.
- Since the system changes randomly, it is generally impossible to predict the exact state of the system in the future. However, the statistical properties of the system's future can be predicted. The changes of state are called *transitions*, and the probabilities associated with various state-changes are called *transition probabilities*.
- A Markov chain is a sequence of random variable that verify this properties. Markov chain is characterized by the properties of its states, *e.g. transiency, periodicity, and ergodicity*.

Markov Process

- ***Definition***: *HMM. A Hidden Markov Model is defined by*

$$\underline{x}(t_{k+1}) \sim p_{\underline{x}}\left(\underline{x}(t_{k+1})/\underline{x}(t_k)\right)$$
$$y(t_k) \sim p_{\underline{x}}\left(\underline{y}(t_k)/\underline{x}(t_k)\right)$$

N.B. for simplicity we will note : $x(t_k) = x(k)$

- The pdf $p(x(t_{k+1})/x(tk))$ describes the evolution of the state variable over time. In general, it can *be non-Gaussian and include nonlinearities*.
- The state process $x(k)$ is an unobserved (hidden) Markov process. Information about this process is indirectly obtained from measurements (observations) $y(k)$ according to the measurement model.
- The observation process $y(k)$ is assumed to be **conditionally independent** of the state process, *i.e.*,

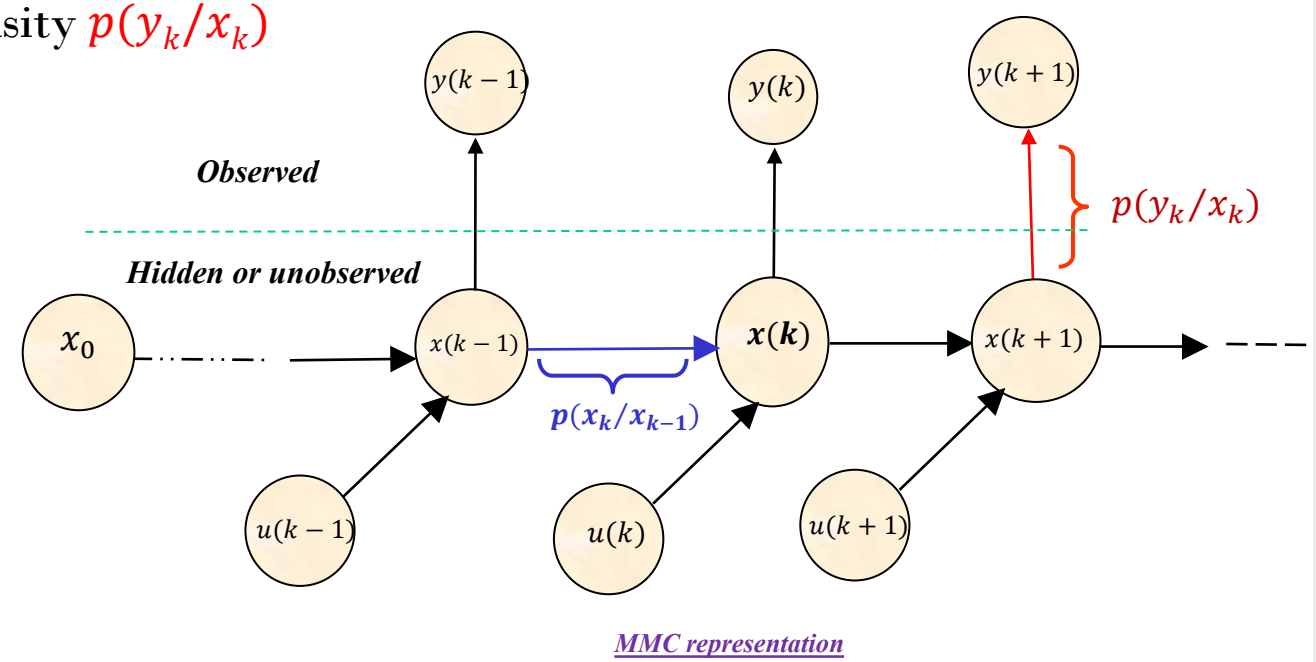
$$p\left(\underline{y}(t_k)/\underline{x}(t_1), \underline{x}(t_2), \ldots, \underline{x}(t_N)\right) = p\left(\underline{y}(t_k)/\underline{x}(t_k)\right) \quad \forall\, 1 \le k \le N$$

- and **mutually independent**, *i.e.*,

$$p\left(\underline{y}(t_1), \underline{y}(t_2), \ldots, \underline{y}(t_N)/\underline{x}(t_1), \underline{x}(t_2), \ldots, \underline{x}(t_N)\right) = \prod_{n=1}^{N} p\left(\underline{y}(t_n)/\underline{x}(t_1), \underline{x}(t_2), \ldots, \underline{x}(t_N)\right)$$
$$= \prod_{n=1}^{N} p\left(\underline{y}(t_n)/\underline{x}(t_n)\right)$$

Markov Process

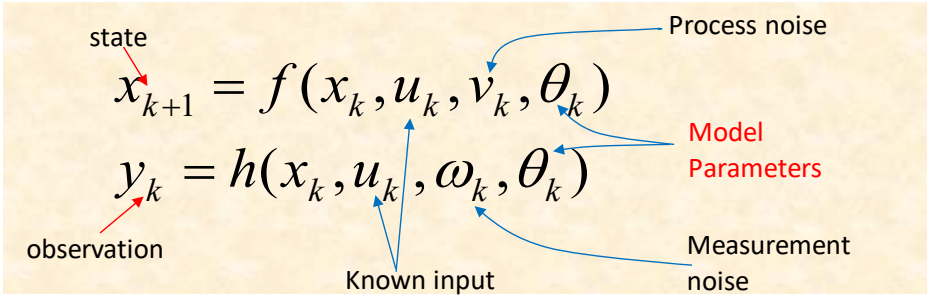
- The hidden Markov models have a great modeling power and can easily describe the structures of complex temporal dependencies.
- If we assume explicit expressions for both the system model and the measurement model, we result in the ***state-space model***.
- We need to specify:
  - Transition density $p(x_k/x_{k-1})$
  - Emission density $p(y_k/x_k)$



MMC representation

BAYESIAN ESTIMATION

- **Model:**
  - **Discrete nonlinear, non-Gaussian multivariable state space model**



- **Assumptions:**
  - Process, v_k, and measurement, ω_k, noises are supposed mutually white and independent random variables with zeros means.
  - Observations are mutually independent over time.
  - If now the noises are additive ones (usually the case) then

$$x_{k+1} = f(x_k, u_k, \theta_k) + v_k$$
$$y_k = h(x_k, u_k, \theta_k) + \omega_k$$

The mutually independence leads then to

$$p(\underline{y}_1, \underline{y}_2, \dots, \underline{y}_N / \underline{x}_1, \underline{x}_2, \dots, \underline{x}_N) = \prod_{k=1}^{N} p_x\left(\underline{y}_k / \underline{x}_k\right) = \prod_{k=1}^{N} p_\omega\left(\underline{y}_k - h(x_k, u_k, \theta_k)\right)$$

and using Markov property we have

$$p\left(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N\right) = \prod_{k=1}^{N-1} p_x\left(\underline{x}_{k+1} / \underline{x}_k\right) = \prod_{k=1}^{N-1} p_v\left(\underline{x}_{k+1} - f(x_k, u_k, \theta_k)\right)$$

BAYESIAN ESTIMATION

Bayesian Inference

- **Inverse Problem**: *Stochastic filtering is an inverse problem.* Given collected date y_k, and known functions $f(\cdot)$ and $h(\cdot)$, one needs to find the optimal or suboptimal estimate $\hat{x}_k$. *This mapping from output to input space is generally non-unique.*

- **Bayesian inference** is the problem of estimating the hidden variables (states or parameters) of a system in an optimal and consistent fashion as a set of noisy or incomplete observations of the system becomes available online.

- In Bayesian inference, all of *uncertainties* (including states, parameters) are treated as random variables. The objective is to use *priors and causal knowledge*, quantitatively and qualitatively, to infer the conditional probability, given observations.

Bayesian statistics

- *Sufficiency* and Likelihood principles are two axiomatic principles in the Bayesian inference. Three problems are related to Bayesian statistics:

- **Normalization**: Given the prior $p(x)$ and likelihood $p(y/x)$, the posterior $p(x/y)$ is obtained by the product of prior and likelihood divided by a normalizing factor as

$$p(x/y) = \frac{p(x,y)}{p(y)} = \frac{p(y/x)p(x)}{\int p(y/x)p(x)dx}$$

- **Marginalization**: Given the joint posterior $p(x,z)$, the marginal posterior is

$$p(x/y) = \int p(x,z/y)\,dz$$

- **Expectation**: Given the conditional *pdf*, some averaged statistics of interest can be calculated

$$\mathbb{E}_{p(x/y)}[f(x)] = \int f(x)\,p(x)dx$$

BAYESIAN ESTIMATION

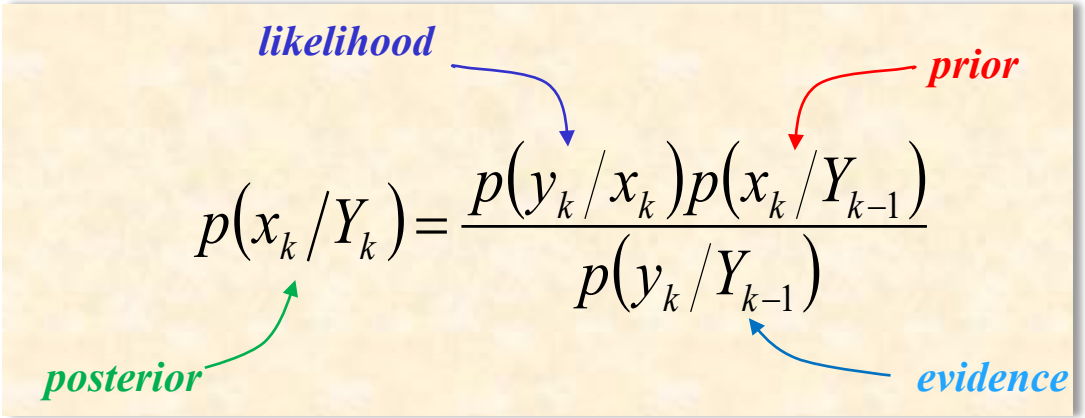
Markov Process

- Bayesian filtering is aimed to apply the Bayesian statistics and Bayes rule to probabilistic inference problems, and specially the stochastic filtering problem.

- The optimal solution to this problem is given by the *recursive Bayesian* estimation algorithm *which recursively updates the posterior density of the system state as new observations arrive.*

- This **posterior density** constitutes the complete solution to the probabilistic inference problem, and allows us to calculate any "*optimal*" estimate of the state (*see Part II of this course*)

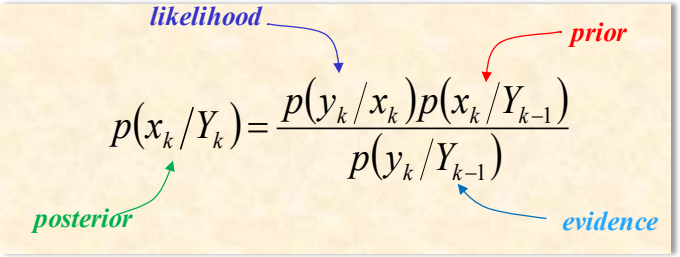
Recursive Bayesian estimation

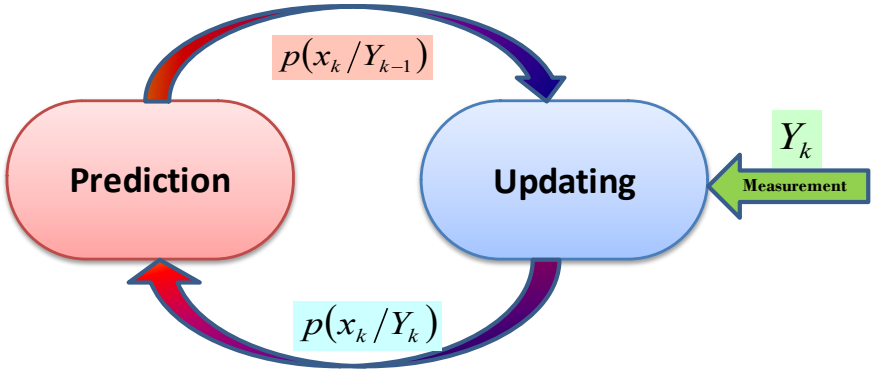


- The recursive Bayesian estimation suppose that:
  - The state is a first order Markov process
  - The observations are independents of the given states
- To simplified the notation, we use

$$p(x_k/Y_k) = p(x_k/y_{1:k})$$

Recursive Bayesian estimation





- It is a recursive estimation loop where the calculation or approximation of the following three terms is involved:

1. **Prior:** Define the knowledge of the model and it is given by the propagation of past state into future before new observation is made

$$p(x_k/y_{1:k-1}) = \int p(x_k/x_{k-1})\,p(x_{k-1}/y_{1:k-1})\,dx_{k-1}$$

$p(x_k/x_{k-1})$: it is the transition density of the state given by the process model

2. **Likelihood**: Expresses the belief in the occurrence of an event. It is defined in terms of observation model and it is determined using the measurement noise model.
3. **Evidence**: Involves from marginalization the following integral

$$p(y_k/y_{1:k-1}) = \int p(y_k/x_k)\,p(x_k/y_{1:k-1})\,dx_k$$

- **Recursive Bayesian estimation**
  - The "proof " is trivial and only involves rewriting using Bayes rules :

$$p(x_k/Y_k) = \frac{p(\boldsymbol{Y_k}/x_k)p(x_k)}{p(\boldsymbol{Y_k})}$$

$$= \frac{p(y_k, \boldsymbol{Y_{k-1}}/x_k)p(x_k)}{p(y_k, \boldsymbol{Y_{k-1}})}$$

$$= \frac{p(y_k/\boldsymbol{Y_{k-1}}, x_k)p(\boldsymbol{Y_{k-1}}/x_k)p(x_k)}{p(y_k/\boldsymbol{Y_{k-1}})p(Y_{k-1})}$$

$$= \frac{p(y_k/\boldsymbol{Y_{k-1}}, x_k)p(x_k/\boldsymbol{Y_{k-1}})p(\boldsymbol{Y_{k-1}})p(x_k)}{p(y_k/\boldsymbol{Y_{k-1}})p(Y_{k-1})p(x_k)}$$

$$= \frac{p(y_k/x_k)p(x_k/\boldsymbol{Y_{k-1}})}{p(y_k/\boldsymbol{Y_{k-1}})}$$

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

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

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

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

Optimal Nonlinear estimation

- Problem statement

$$x_{k+1} = f(x_k, u_k, v_k, \theta_k)$$
$$y_k = h(x_k, u_k, \omega_k, \theta_k)$$

- The criterion of optimality used for Bayesian filtering is *the Bayes risk of BMSE* (see Part II). it is ***optimal*** in a sense that it seeks the *posterior distribution* which integrates and uses all of *available information* expressed by probabilities.

- The optimal Bayesian estimation is given by (see Part II) by

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = E(x_k/Y_k) = \int x_k \underbrace{\boldsymbol{p(x_k/Y_k)}}_{\text{Recusively}} dx_k$$

- **Different practical solutions:**

BAYESIAN ESTIMATION

- **Optimal Nonlinear estimation**
  - **Gaussian approximations**:
    - Extended Kalman filter (**EKF**): non-linear model (low nonlinearities, approximation of order 1)
    - Unscented Kalman filter (**UKF**): non-linear model (linearization of order 2)

⇒ *These estimators make an approximation of the posterior density by a Gaussian distribution. To overcome this limitation, particle filters and their variants have been developed*

  - **Sequential Monte Carlo Methods** (Particle filters): non-linear model , non-Gaussian noises, ….
    - Bayesian Importance Sampling (**IS**)
    - Sampling-Importance Resampling (**SIR**)
    - …

Gaussian Approximation

- Kalman estimation is the most pervasive and universal tools of engineering

- It is based on the propagation of the *two first moments* of the distribution of $x(k)$ and then uses prediction and correction procedure.

⇨ ***What happens then, when propagating means and covariances through nonlinear transformations ?***

- *To ease the understanding and introduce the core of the Kalman approaches to nonlinear system, we will consider in detail the calculation of mean and covariance of a random variable through nonlinear function.*

Propagating a Gaussian random variable

- Consider a random variable x with mean m_x and covariance P_{xx}. And suppose that a second random variable z is related to x through a nonlinear function $g(.)$:

$$z = g(x)$$

- The goal is to calculate the mean m_z and covariance P_{zz} in order to have a consistent estimate of z.

- This problem can be examined by considering *Taylor series expansion*:

$$g(x) = g(m_x + \delta x) = g(m_x) + \left.\frac{\partial g(x)}{\partial x}\right|_{x=m_x} \delta x +$$

$$\frac{1}{2}\sum_{i=1}^{n_x}\sum_{j=1}^{n_x} \delta x_j \delta x_i \left.\frac{\partial^2 g(x)}{\partial x_i \partial x_j}\right|_{x=m_x} + \ldots + \frac{1}{n!}\left(\sum_{i=1}^{n_x} \delta x_i \frac{\partial}{\partial x_i \partial x_j}\right)^n g(x)\Big|_{x=m_x}$$

  - considering small variations, $\delta x = x - m_x$, with zero mean and covariance P_{xx}
  - δx_i is the i^{th} component of δx

Propagating a Gaussian random variable

- Taking expectation leads to:

$$E(z) = g(m_x) + \left.\frac{\partial g(x)}{\partial x}\right|_{x=m_x} E(\delta x) + \frac{1}{2}\sum_{i=1}^{n_x}\sum_{j=1}^{n_x} E\left(\delta x_j \delta x_i\right) \left.\frac{\partial^2 g(x)}{\partial x_i \partial x_j}\right|_{x=m_x} + \ldots.$$

- Considering that:
  - $g(m_x)$ is <u>deterministic</u> and the mean of $E(\delta x) = 0$
  - the covariance $E\left(\delta x_j \delta x_i\right) = P_{xx}(i,j) = P_{xx}(j,i)$
- Then the transformed mean and covariance are given by:

$$\boldsymbol{E(z)} = g(m_x) + \frac{1}{2}\sum_{i=1}^{n_x}\sum_{j=1}^{n_x} P_{xx}(i,j) \left.\frac{\partial^2 g(x)}{\partial x_i \partial x_j}\right|_{x=m_x} + \ldots$$

$$\boldsymbol{P_{zz}} = \frac{\partial g(x)}{\partial x} P_{xx} \left(\frac{\partial g(x)}{\partial x}\right)^T + \cdots$$

<u>Correct evaluation:</u>

Up to which order we want to evaluate moments ?

Order 1 : EKF

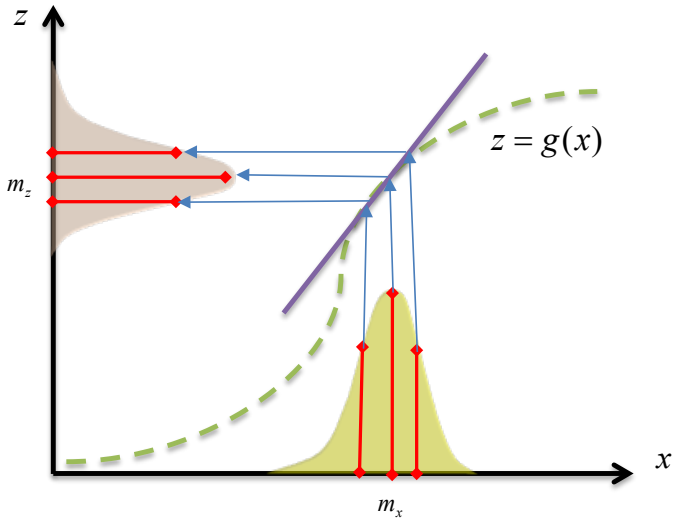
Order 2: UKF

Problem statement

- We consider nonlinear system represented by:

$$\begin{cases} \underline{x}_{k+1} = f(\underline{x}_k) + g(\underline{u}_k) + H_k \underline{\zeta}_k \\ \underline{y}_k = \varphi(\underline{x}_k) + \underline{\omega}_k \end{cases}$$

- $f(\cdot), g(\cdot)$ and $\varphi(\cdot)$ are nonlinear functions. H is constant and the noises enter additively.
- Assumptions: The same hypothesis on noises and initial values as for the Kalman filter
- Approach: *The EKF is **Sub-optimal**. It is based on a **first order** (Taylor series) linearization of nonlinear state space model at current state estimate. The EKF calculates the **Jacobian** of $f(\cdot), g(\cdot)$ and $\varphi(\cdot)$ around the estimated state, which in turn, yields a trajectory of model function centered around this state.*



EXTENDED KALMAN FILTER

- **EKF derivation**
  - **Prediction**: Expectation gives

$$\hat{\underline{x}}_{k+1|k} = E\left\{\underline{x}_{k+1}\middle|y_1^k\right\} = E\left\{f(\underline{\mathbf{x}}_k)\middle|y_1^k\right\} + E\left\{g(\underline{u}_k)\middle|y_1^k\right\} + E\left\{H_k\underline{\zeta}_k\middle|y_1^k\right\}$$

  - Using assumptions and first order approximation we have:

$$E\left\{g(\underline{u}_k)\middle|y_1^k\right\} = g(\underline{u}_k)$$ ***We suppose here that the input is deterministic***

$$E\left\{H_k\underline{\zeta}_k\middle|y_1^k\right\} = H_k E\left\{\underline{\zeta}_k\middle|y_1^k\right\} = 0$$

$$E\left\{f(\underline{\mathbf{x}}_k)\middle|y_1^k\right\} \approx f\left(E\left\{\underline{\mathbf{x}}_k\middle|y_1^k\right\}\right) = f\left(\hat{\underline{x}}_{k|k}\right)$$

1st order Approximation

$$\hat{\underline{x}}_{k+1|k} = f\left(\hat{\underline{x}}_{k|k}\right) + g(\underline{u}_k)$$

EKF derivation

- **Prediction:** Calculus of the prediction covariance $P_{k+1|k}$

Linearization of $\underline{\mathbf{x}}_{k+1} = f(\underline{\mathbf{x}}_k) + g(\underline{u}_k) + H_k \underline{\zeta}_k$ $= \underline{\hat{x}}_{k+1|k}$

around $\underline{\hat{x}}_{k|k}$ $\Rightarrow \underline{\mathrm{x}}_{k+1} \approx f(\underline{\hat{x}}_{k|k}) + F_k(\underline{\mathrm{x}}_k - \underline{\hat{x}}_{k|k}) + g(\underline{u}_k) + H_k \underline{\zeta}_k$

$$\Rightarrow \quad \underline{\mathbf{x}}_{k+1} - \underline{\hat{x}}_{k+1|k} \approx F_k \left(\underline{\mathbf{x}}_k - \underline{\hat{x}}_{k|k}\right) + H_k \underline{\zeta}_k$$

with $$F_k = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\underline{\hat{x}}_{k|k}}$$

By following then the same procedure and methodology than the Kalman filter, we obtain the variance of the prediction:

$$P_{k+1|k} = F_k P_{k|k} F_k^T + H_k Q_k H_k^T$$

EKF derivation

- **Correction:** we use the observation equation

Linearization of $\underline{\mathbf{y}}_{k+1} = \varphi(\underline{\mathbf{x}}_{k+1}) + \underline{\omega}_k$

around $\hat{\underline{x}}_{k+1|k} \Rightarrow \quad \underline{\mathbf{y}}_{k+1} \approx \varphi\left(\hat{\underline{x}}_{k+1|k}\right) + C_{k+1}\left(\underline{\mathbf{x}}_{k+1} - \hat{\underline{x}}_{k+1|k}\right) + \underline{\omega}_k$

$$\Rightarrow \quad \underline{\mathbf{y}}_{k+1} - \hat{\underline{y}}_{k+1|k} \approx C_{k+1}\left(\underline{\mathbf{x}}_{k+1} - \hat{\underline{x}}_{k+1|k}\right) + \underline{\omega}_k$$

$$\text{with} \quad C_{k+1} = \left.\frac{\partial\, \varphi(x)}{\partial\, x}\right|_{x=\hat{\underline{x}}_{k+1|k}}$$

By following the same procedure and methodology than the Kalman filter, we obtain:

$$K_{k+1} = P_{k+1|k} C_{k+1}^T \left(C_{k+1} P_{k+1|k} C_{k+1}^T + R_{k+1}\right)^{-1}$$

$$P_{k+1|k+1} = \left(I - K_{k+1} C_{k+1}\right) P_{k+1|k}$$

EKF derivation

Correction:

Estimator : $\hat{\underline{x}}_{k+1|k+1} = \hat{\underline{x}}_{k+1|k} + K_{k+1}\left(\underline{y}_{k+1} - \hat{\underline{y}}_{k+1|k}\right)$

Output prediction:

$$\hat{\underline{y}}_{k+1|k} = E\left\{\underline{\mathbf{y}}_{k+1}\middle|y_1^k\right\} = \underbrace{E\left\{\varphi(\underline{\mathbf{x}}_{k+1})\middle|y_1^k\right\}}_{\approx \varphi\left(E\left\{\underline{\mathbf{x}}_{k+1}\middle|y_1^k\right\}\right) = \varphi\left(\hat{\underline{x}}_{k+1|k}\right)} + \underbrace{E\left\{\underline{\omega}_k\middle|y_1^k\right\}}_{=0}$$

1st Approximation

$$\hat{\underline{x}}_{k+1|k+1} = \hat{\underline{x}}_{k+1|k} + K_{k+1}\left(\underline{y}_{k+1} - \varphi\left(\hat{\underline{x}}_{k+1|k}\right)\right)$$

EXTENDED KALMAN FILTER

Kalman Algorithm

Initialization:

Choose $\hat{\underline{x}}_{0|0}$ et $P_{0|0}$.

$k = 0$

Prediction:

$$\hat{\underline{x}}_{k+1|k} = F_k\, \hat{\underline{x}}_{k|k} + G_k\, \underline{u}_k$$

$$P_{k+1|k} = F_k\, P_{k|k}\, F_k^T + H_k\, Q_k\, H_k^T$$

Measure $\underline{y}_{k+1}$

Correction:

$$K_{k+1} = P_{k+1|k}\, C_{k+1}^T \left(C_{k+1}\, P_{k+1|k}\, C_{k+1}^T + R_k\right)^{-1}$$

$$\hat{\underline{x}}_{k+1|k+1} = \hat{\underline{x}}_{k+1|k} + K_{k+1}\left(\underline{y}_{k+1} - C_{k+1}\, \hat{\underline{x}}_{k+1|k}\right)$$

$$P_{k+1|k+1} = \left(I - K_{k+1} C_{k+1}\right) P_{k+1|k}$$

$k = k + 1$

EKF Modifications :

$$\hat{\underline{x}}_{k+1|k} = f\left(\hat{\underline{x}}_{k|k}\right) + g\left(\underline{u}_k\right)$$

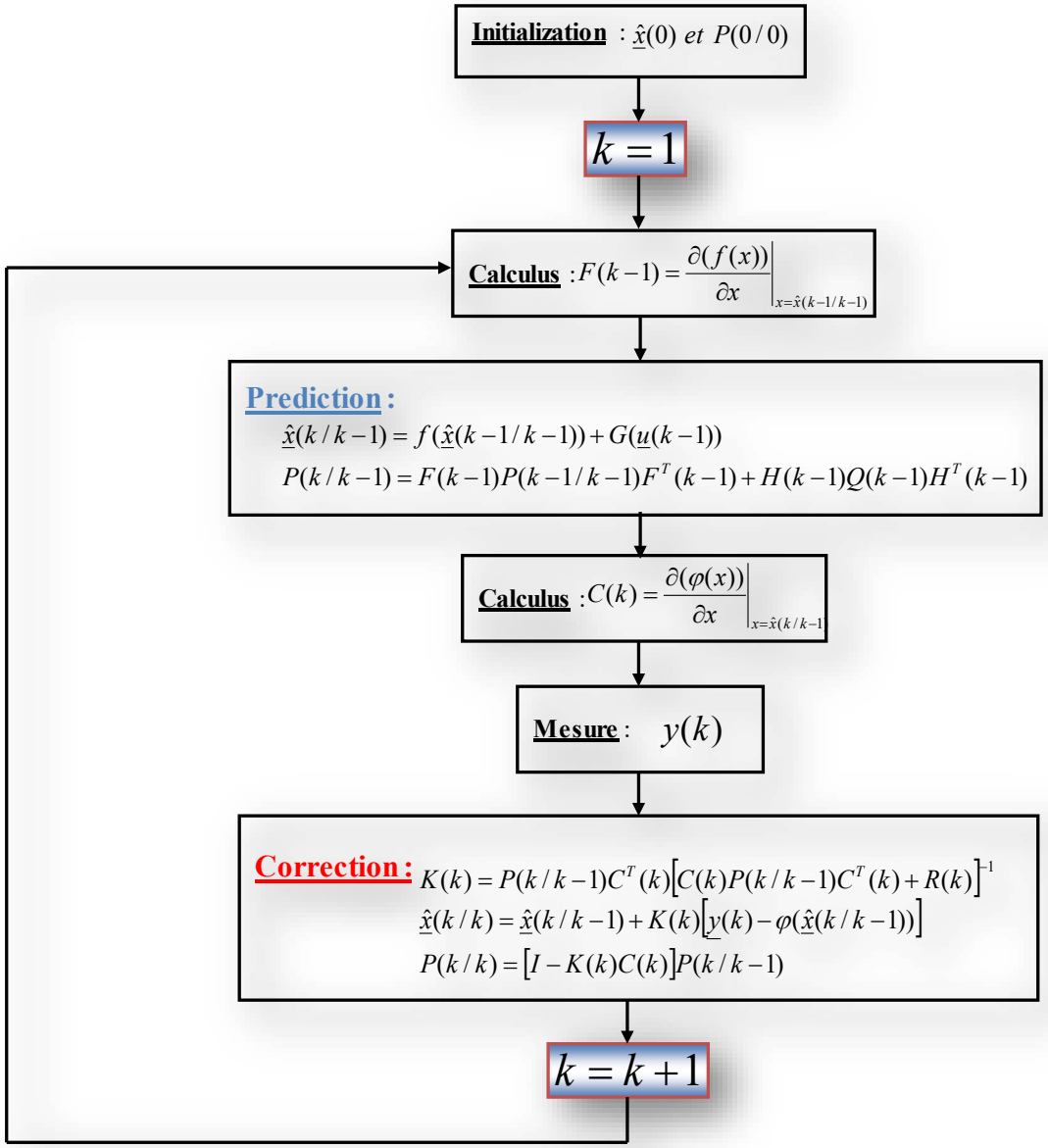
$$F_k = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{\underline{x}}_{k|k}}$$

$$C_{k+1} = \left.\frac{\partial \varphi(x)}{\partial x}\right|_{x=\hat{\underline{x}}_{k+1|k}}$$

$$\hat{\underline{x}}_{k+1|k+1} = \hat{\underline{x}}_{k+1|k} + K_{k+1}\left(\underline{y}_{k+1} - \varphi\left(\hat{\underline{x}}_{k+1|k}\right)\right)$$

EXTENDED KALMAN FILTER

EKF Algorithm



Remarks:

- Matrices $P_{k+1|k}$, $P_{k+1|k+1}$, K_{k+1}, cannot be no more calculated off-line because they depend on the measurements

- The convergence of filter is *not straightforward guarantied*. It depends on
  - *Initialization values* $P_{0|0}$ *and* $x_{0|0}$
  - *The nonlinearities acting on the system*

- Suppose now that the state equation is given by: $\underline{\mathbf{x}}_{k+1} = f(\underline{\mathbf{x}}_k, \underline{u}_k) + H_k \underline{\zeta}_k$
  then two cases:
  1. if $\underline{u}_k$ is deterministic then, it is the same calculus.
  2. if $\underline{u}_k$ is not perfectly known (random variable) , *i.e.*, we measure: $\underline{v}_k = \underline{u}_k + \underline{\gamma}_k$
     where γ_k is a Gaussian vector with zero mean and covariance R_γ.

  ⇨ *Then, we will have the following modifications on the former EKF algorithm:*

$$\underline{\hat{x}}_{k+1|k} = f\left(\underline{\hat{x}}_{k|k}, \underline{v}_k\right) \qquad P_{k+1|k} = F_k P_{k+1|k} F_k^T + J_k R_\gamma J_k^T + H_k Q_k H_k^T$$

$$\text{with :} \quad F_k = \left.\frac{\partial f(x,u)}{\partial x}\right|_{(x=\underline{\hat{x}}_{k|k}, u=\underline{v}_k)} \qquad J_k = \left.\frac{\partial f(x,u)}{\partial u}\right|_{(x=\underline{\hat{x}}_{k|k}, u=\underline{v}_k)}$$

Exercise

- Consider the following non-linear state space system:

$$\begin{cases} \underline{x}_{k+1} = \begin{bmatrix} \underline{x}_{k+1}(1) \\ \underline{x}_{k+1}(2) \end{bmatrix} = \begin{bmatrix} \underline{x}_k^2(1) + \underline{x}_k^2(2) \\ \underline{x}_k(1) \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \varsigma_k \\ \underline{y}_k = \begin{bmatrix} \underline{y}_k(1) \\ \underline{y}_k(2) \end{bmatrix} = \begin{bmatrix} \underline{x}_k(1) \\ \underline{x}_k^3(2) \end{bmatrix} + \begin{bmatrix} \omega_k(1) \\ \omega_k(2) \end{bmatrix} \end{cases}$$

- The noises are supposed Gaussians with zero means, mutually independent and with covariances:

$$\begin{cases} E(\varsigma_k) = 0 \\ E(\underline{\omega}_k) = 0 \end{cases} \quad \begin{cases} E(\varsigma_k^2) = Q = 1 & \textit{scalar} \\ E(\omega_k \omega_k^T) = R = I & \textit{vector} \end{cases}$$

- Calculate for k=1 and k=2, the estimation and its covariance, *i.e.*, $\underline{\hat{x}}_{k/k}$ and $P_{k/k}$

We give the following initial and output values:

$$\underline{\hat{x}}_{0/0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, P_{0/0} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$For\ k = 1 \Rightarrow \underline{y}_1 = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} and\ \ for\ k = 2 \Rightarrow \underline{y}_2 = \begin{bmatrix} 0.0001 \\ 0 \end{bmatrix}$$

UNSCENTED KALMAN FILTER

Introduction

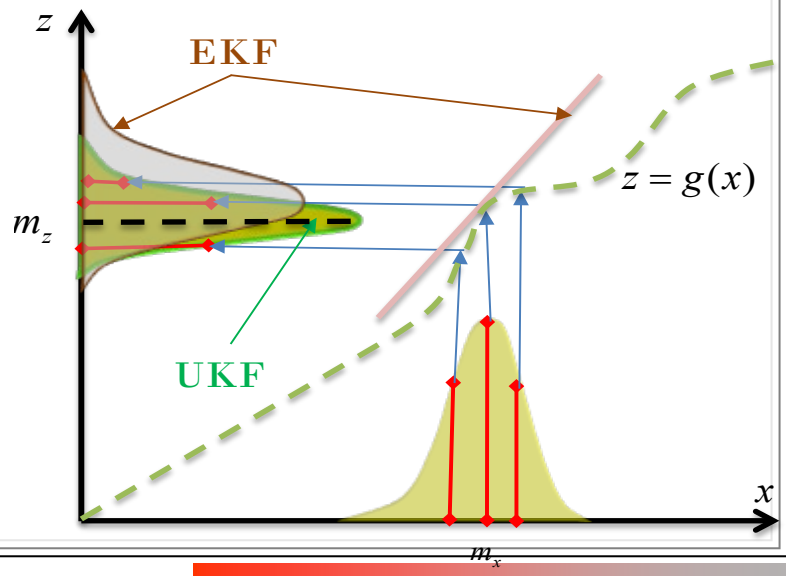
- EKF only uses the *first order terms* of the Taylor series expansion of the nonlinear functions, it often introduces large errors in the estimated statistics of the posterior distributions of the states.
- First order approximation, becomes useless when models are highly nonlinear and then local linearity assumption breaks down.
- The problem of propagating Gaussian random variables through a nonlinear function can also be approached using another technique, namely the ***Unscented Transformation*** **(UT)**.
- UT is founded on the intuition that *it is easier to approximate a Gaussian distribution than to approximate an arbitrary nonlinear function or transformation.*
- Instead of linearizing, the UKF specifies the Gaussian state distribution using *a set of points and propagates* them through the *actual nonlinear* function where they completely captures the *posteriori mean and covariance* accurately to the 2^{nd} order.

Unscented Transformation

- It is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation. It is based on:
  - *Generating a set of points whose mean, covariance match those of the random variable*
  - *The NL function is applied to each of these points to yield a transformed sample.*
  - *The predicted mean and covariance can be recalculated from the propagated points, yielding more accurate results compared to ordinary function linearization.*
- It is not a Monte-Carlo method !! Although this method bares a resemblance.
- It can be seen as a "*deterministic*" Monte Carlo approach !! *The samples are not drawn at random but rather according to a deterministic algorithm.*

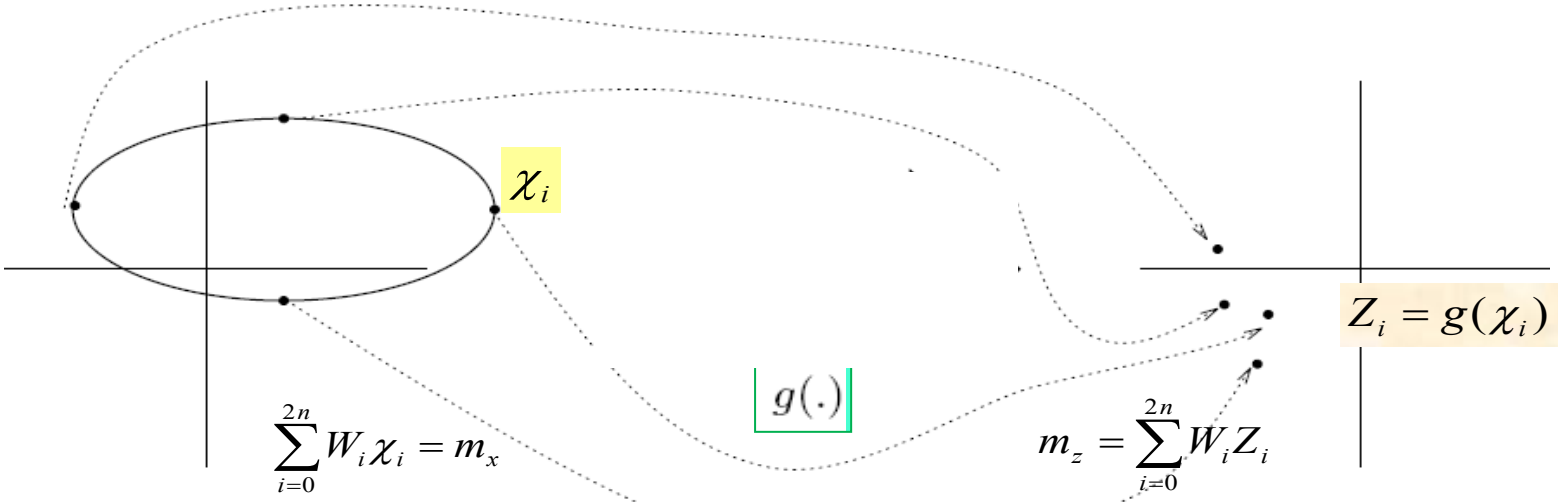
Propagation of a random variable through a nonlinear function:

- *EKF first order linearization*
- *UKF Unscented transformation*



Unscented Transformation

- Resume the propagation problem of a random variable x ($\dim(x) = n_x$) through an arbitrary nonlinear function, $z = g(x)$
- To calculate the first two moments of z we proceed as follows:
  - *Determine a set of* $2n_x + 1$ *weighted samples called* ***Sigma-Points (SP):*** $\{\chi_i, Wi\}$. *The SP means and covariances are equal to* m_x *and* P_{xx}.
  - *Propagate each sigma point through the nonlinear function to yield the set of transformed points.*
  - *Approximate mean and covariance by weighted average of the transformed points.*



UNSCENTED KALMAN FILTER

Unscented Transformation

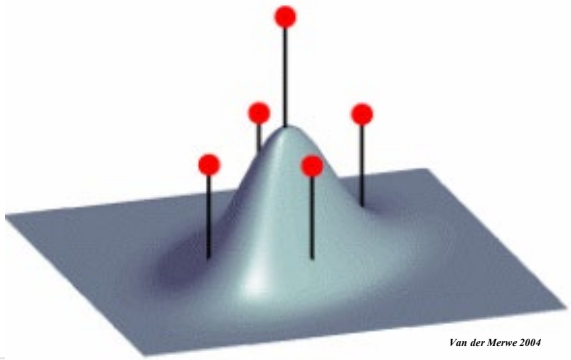
- A set of sigma points is given by:

$$\begin{aligned} \chi_0 &= m_x & W_0 &= \kappa/(n+\kappa) \\ \chi_i &= m_x + \sqrt{n+\kappa}\left(\sqrt{P_{xx}}\right)_i & W_i &= 1/2(n+\kappa) \qquad i = 1,\ldots,n \\ \chi_{i+n_x} &= m_x - \sqrt{n+\kappa}\left(\sqrt{P_{xx}}\right)_i & W_{i+n_x} &= 1/2(n+\kappa) \end{aligned}$$

Exercise: *Using this choice of SP, show that:*

$$m_z = \sum_{i=0}^{2n} W_i Z_i \quad \text{and} \quad P_{zz} = \sum_{i=0}^{2n} W_i (Z_i - m_z)(Z_i - m_z)^T$$

- W_i is the weight associated with the i^{th} sigma-point such that: $\sum_{i=0}^{2n} W_i = 1$
- The parameter κ is a **real** scaling factor. It provides an extra degree of freedom to "*fine tune*" the higher order moments of the approximation and can be used to reduce the overall prediction errors.
- $\left(\sqrt{P_{xx}}\right)_i$ is the the i^{th} column (or row) of the matrix square roots of the covariance P_{xx}. We use the numerically efficient ***Cholesky factorization*** method to drive it (*see* ***chol.m*** *in Matlab help*).
- *Example:*
  - ***Weighted sigma-points for a 2-dimensional Gaussian random variable:***
    - *The sigma-points lie along the major eigen-axes of the covariance matrix and complete captures the first and second order statistics.*
    - *The height of each sigma-point indicates its relative weight.*



Scale Unscented Transformation (SUT)

- To correctly manage the severity of specific nonlinearity we use an appropriate choice of the parameter κ to scale the SP towards or away from the mean of the prior distribution.
- In some cases, the choice of κ can lead to a negative weight and then the calculated covariance can become *non-positive semi-definite*.
- *Scaled Unscented Transformation (SUT)*: The idea is to introduce an extra degree of freedom to control the scaling of the SP and avoiding non-negative covariance through the choice of a new parameter α.
- The SUT replaces the original set of sigma-points with a transformed set where α is the new sigma-point *scaling parameter:*

$$\chi_i' = \chi_0 + \alpha(\chi_i - \chi_0) \qquad W_i' = \begin{cases} W_i/\alpha^2 & i = 1,\ldots,2n \\ W_0/\alpha^2 + (1+1/\alpha^2) & i = 0 \end{cases}$$

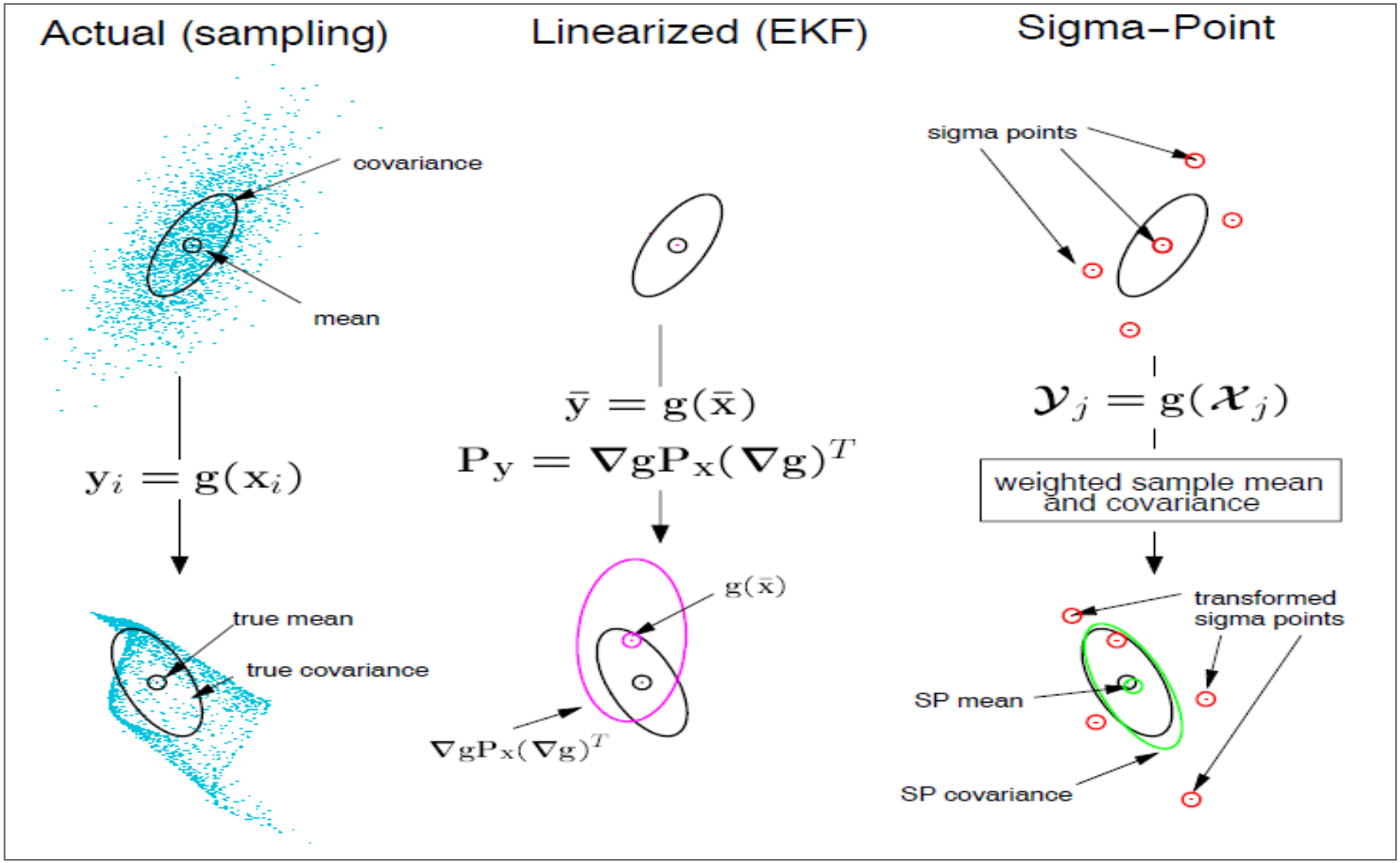
- Another parameter, β, is also introduced which affects the weighting of the zeroth sigma-point for the calculation of the covariance. We will differentiate between the *initial mean weight*, W_0^m and the *covariance weight*, W_0^P . A single unified choice is given by:

$$\begin{cases} \chi_0 = m_x \\ \chi_i = m_x + \sqrt{n+\lambda}\left(\sqrt{P_{xx}}\right)_i & i = 1,\ldots,n \\ \chi_{i+n_x} = m_x - \sqrt{n+\lambda}\left(\sqrt{P_{xx}}\right)_i & i = n+1,\ldots,2n \end{cases} \qquad \begin{cases} W_0^m = \lambda/(n+\lambda) & i = 0 \\ W_0^P = \lambda/(n+\lambda) + (1-\alpha^2+\beta) & i = 0 \\ W_i^m = W_i^P = 1/2(n+\lambda) & i = 1,\ldots,2n \end{cases} \quad \text{and } \lambda = \alpha^2(n+\kappa) - n$$

UNSCENTED KALMAN FILTER

- **Example:**

Van der Merwe 2004



Implementation

- The UKF is obtained by *slightly restructuring the Kalman recursive* algorithm by applying the unscented transformation.
- The nonlinear system is

$$x_{k+1} = f(x_k, u_k, v_k, \theta_k)$$
$$y_k = h(x_k, u_k, \omega_k, \theta_k)$$

- We need first to define an **augmented** new state vector, where the state is redefined as **the concatenation of the original state and the process and observation noise random variables**

$$x_k^a = \begin{bmatrix} x_k \\ v_k \\ \omega_k \end{bmatrix} \qquad \dim(x_k^a) = n = n_x + n_v + n_\omega$$

- We also define the covariance matrix of this augmented vector

$$P_{xx}^a = \begin{bmatrix} P_{xx} & 0 & 0 \\ 0 & P_{vv} & 0 \\ 0 & 0 & P_{\omega\omega} \end{bmatrix} = \begin{bmatrix} P_{xx} & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{bmatrix}$$

- *The procedure is to apply the Unscented Transformation or SUT to this new vector to drive prediction estimation of the state and the output*

Algorithm

Initialization:

$$x_k^a = \begin{bmatrix} x_k \\ v_k \\ \omega_k \end{bmatrix} \Rightarrow \begin{cases} \hat{x}_0 = E(x_0) & P_{x_0} = E\big((x_0 - \hat{x}_0)(x_0 - \hat{x}_0)\big)^T \\ \hat{x}_0^a = E(\hat{x}_0^a) = \begin{bmatrix} \hat{x}_0 \\ 0 \\ 0 \end{bmatrix} & P_{x_0} = \begin{bmatrix} P_{x_0} & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{bmatrix} \end{cases}$$

For k =1 *to* N (data size)

1. Sigma-points calculus

$$\boldsymbol{\chi}_{k-1}^a = \begin{bmatrix} \hat{x}_{k-1}^a & \hat{x}_{k-1}^a + \eta\sqrt{P_{k-1}^a} & \hat{x}_{k-1}^a - \eta\sqrt{P_{k-1}^a} \end{bmatrix} \quad with \quad \begin{matrix} \eta = \sqrt{n+\kappa} & UT \\ \eta = \sqrt{n+\lambda} & SUT \end{matrix} \quad and \quad \boldsymbol{\chi}^a = \begin{bmatrix} \boldsymbol{\chi}^x \\ \boldsymbol{\chi}^v \\ \boldsymbol{\chi}^\omega \end{bmatrix}$$

2. Prediction equations

$$\boldsymbol{\chi}_{k/k-1}^x = f(\boldsymbol{\chi}_{k-1/k-1}^x, \boldsymbol{\chi}_{k-1/k-1}^v, u_{k-1})$$

$$\hat{x}_{k/k-1} = \sum_{i=0}^{2n} W_i^m \boldsymbol{\chi}_{i,k/k-1}^x \qquad (or\ W_0^m)$$

$$P_{k/k-1}^{xx} = \sum_{i=0}^{2n} W_i^P \left(\boldsymbol{\chi}_{i,k/k-1}^x - \hat{x}_{k/k-1}\right)\left(\boldsymbol{\chi}_{i,k/k-1}^x - \hat{x}_{k/k-1}\right)^T \quad (or\ W_0^P)$$

Algorithm

3. Correction equations

$$\mathbf{Y}_{k/k-1} = h(\boldsymbol{\chi}^{x}_{i,k/k-1}, \boldsymbol{\chi}^{\omega}_{i,k-1/k-1}, u_{k-1})$$

$$\hat{y}_{k/k-1} = \sum_{i=0}^{2n} W_i^m \mathbf{Y}^{x}_{i,k/k-1}$$

$$P_k^{yy} = \sum_{i=0}^{2n} W_i^P \left(\mathbf{Y}^{x}_{i,k/k-1} - \hat{y}_{k/k-1}\right)\left(\mathbf{Y}^{x}_{i,k/k-1} - \hat{y}_{k/k-1}\right)^T$$

$$P_k^{yx} = \sum_{i=0}^{2n} W_i^P \left(\boldsymbol{\chi}^{x}_{i,k/k-1} - \hat{x}_{k/k-1}\right)\left(\mathbf{Y}^{x}_{i,k/k-1} - \hat{y}_{k/k-1}\right)^T$$

Gain $\quad K_k = P_k^{yx}\left(P_k^{yy}\right)^{-1}$

Estimation $\quad \hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k(y_k - \hat{y}_{k/k-1})$

Covariance $\quad P^{xx}_{k/k} = P^{xx}_{k/k-1} - K_k\left(P_k^{yy}\right)K_k^T$

End

Remarks:

- Tuning parameters are: *initial values, covariance values and the scaling parameters*
- For the scaling parameters:
  - Choose $\kappa \geq 0$ to guarantee positive semi-definiteness of the covariance matrix. The default choice is $\kappa = 0$.
  - Choose $0 \leq \alpha \leq 1$ and $\beta \geq 0$. this last incorporate knowledge of the higher order moments of the distribution, for a Gaussian an optimal choice is $\beta = 2$.
- If the input vector is considered as random, then it is incorporate into the augmented vector and we apply the same algorithm

$$x_k^a = \begin{bmatrix} x_k \\ u_k \\ v_k \\ \omega_k \end{bmatrix} \qquad P_{xx}^a = \begin{bmatrix} P_{xx} & 0 & 0 & 0 \\ 0 & P_{uu} & 0 & 0 \\ 0 & 0 & P_{vv} & 0 \\ 0 & 0 & 0 & P_{\omega\omega} \end{bmatrix} \qquad \dim(x_k^a) = n = n_x + n_u + n_v + n_\omega$$

Problem statement

- We seek to estimate the state of the following nonlinear SS system :

$$\begin{cases} \dot{\underline{x}}(t) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_2 u + \beta \\ -\alpha x_2^2 + u x_2 \end{bmatrix} + \zeta(t) \\ y(t) = C\underline{x} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \omega(t) \end{cases}$$

where the state vector and parameters (supposed here in this first part known) are

$$\underline{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \quad , \ \alpha = 4 \ and \ \beta = 1$$

Linearization

1. Determine an equilibrium input u_e and state $\underline{x}_e = \begin{bmatrix} x_{1e} & x_{2e} \end{bmatrix}^T$

2. Linearize the SS around this equilibrium point by considering first Taylor series expansion

$$\begin{cases} \dot{\underline{x}}_l = A\underline{x}_l + Bu_l + \zeta(t) \\ y_l = C\underline{x}_l + \omega(t) \end{cases} \quad with \quad \underline{x}_l = \underline{x} - \underline{x}_e \ , \ \underline{u}_l = \underline{u} - \underline{u}_e \ and \ y_l = y - x_{1e}$$

MATLAB PROBLEM

Simulations

1. Create a Matlab file (*.m) with all the "actual" parameters
2. Create a Matlab function (*.m) that calculate the linearized A,B matrices
3. Create two Simulink files (*.mdl) that simulate the non linear and linear unnoised systems (the noises will be added in Matlab)
4. For the linear simulation, we will use the following input

$$u(t) = \begin{cases} u_e & if \quad t < 4s \\ u_e + 1 & if \quad t \geq 4s \end{cases}$$

Kalman filter

1. Create a main program for Kalman that:
   - Simulate and plot output and states of linear system in it first cells (Sampling time is Te= 0.1 s)
   - Program the Kalman algorithm
   - Evaluate it performances
2. Evaluate the filter when simulated with nonlinear system

MATLAB PROBLEM

EKF filter

1. Create a main program for EKF that:
   - Simulate and plot output and states of nonlinear system
   - Calculate the discrete nonlinear system by numerically integrating it between (you can use for example *ode45.m* function)

$$t = kT_e \quad and \quad t = (k+1)T_e$$

   - To calculate the value of the linearized matrices F and G you should use the analytical solution evaluated at the considering state point
   - Program the EKF algorithm
   - Evaluate it performances
2. Evaluate the filter when simulated with different initial values and covariances.

Monte-Carlo simulations

1. Evaluate the performances of these two filters using Monte-Carlo simulations
2. We shall also test the robustness of these filters by using "filter" parameters different from the "actual" parameters

MATLAB PROBLEM

UKF filter

1. Create a main program for UKF that:
   - Simulate and plot output and states of nonlinear system
   - Program the UKF algorithm (use Chol.m for the square root of a matrix)
   - Evaluate it performances
2. Evaluate the filter when simulated with different initial values and covariances.
3. Evaluate the performances of these two filters using Monte-Carlo simulations
4. We shall also test the robustness by using "filter" parameters different from the "actual" parameters
5. Compare performances with EKF

SYLLABUS PART 1

- **Lecture 1:**
  - Course organization
  - Motivations
  - Introductive Examples
  - Review of Probability
- **Lecture 2:**
  - Estimation theory
  - Exercises
- **Lecture 3:**
  - Linear estimators: Least Square, Maximum Likelihood
  - Exercise
- **Lecture 4:**
  - Linear estimator: Kalman Filter
  - Exercise & Matlab class
- **Lecture 5:**
  - Nonlinear estimation: EKF and UKF
  - Exercises & Matlab class
- **Lecture 6:**
  - Nonlinear estimation: PF and RBPF
  - Exercise & Matlab class

OUTLINE OF CHAPTER 3

I. Introduction

II. Recursive Bayesian Estimation

III. Extend Kalman Filtering (EKF)

IV. Unscented Kalman Filtering (UKF)

V. **Particle Filter (PF)**

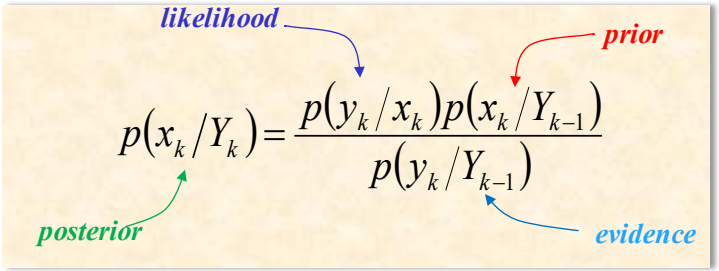
PARTICLE FILTER

Introduction

- In recent years, statistical simulation methods have emerged as approaches to make significant advances in areas as diverse as speech processing, bioinformatics, prosecution and the location, the vision , finance, diagnostic systems,...
- The **particle filtering** methods, simulation methods are sequential Monte-Carlo, in which particles:
  - Move independently
  - interact through selection algorithms (called **resampling**) that channel the particles in the regions of interest.
- They exploit the information accessible on the processes and their evolution laws.
- Require the generation of several weighted samples or particles, simulating the real trajectory of the system.

Introduction

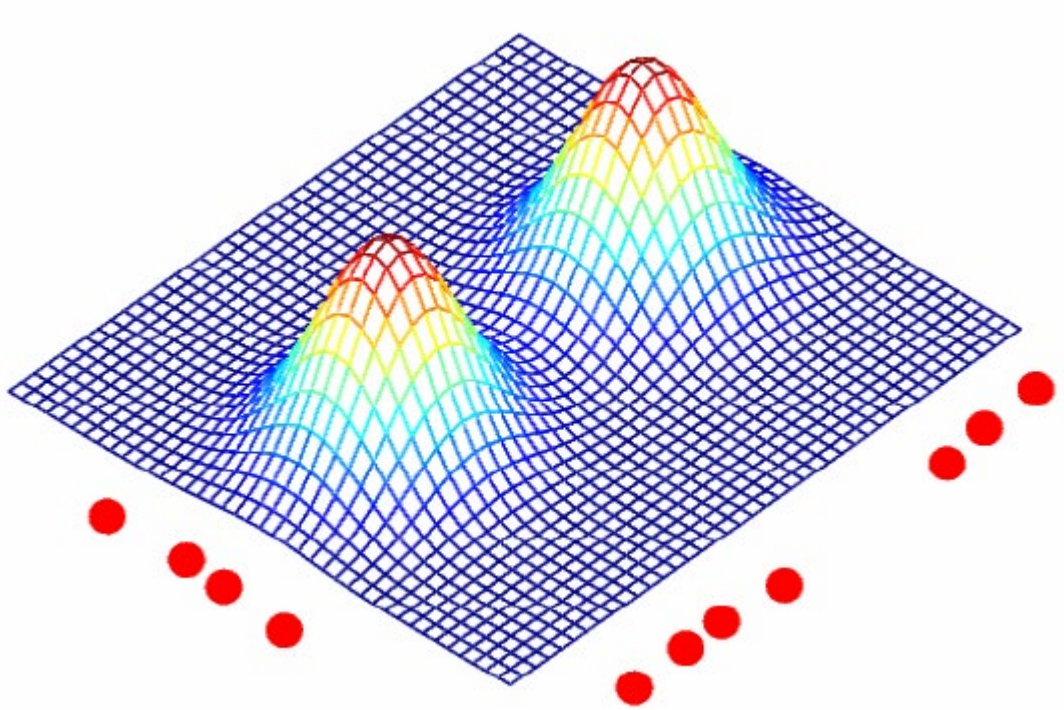
- *Sequential Monte Carlo methods* (SMCM), or particle methods, deal with the problem of recursively estimating the probability density function $p(x_k|Y_k)$.



- The key idea underlying the sequential Monte Carlo methods is to represent the probability density function *by a set of samples* (also referred to as particles, hence the name particle methods) and its associated weights.
- No explicit assumption about the form of prior *pdf* is made. Thus, could be used in general nonlinear, non-Gaussian systems.
- *Particle filtering* is based on Monte Carlo simulation with sequential importance sampling (SIS). The overall goal is to directly implement optimal recursive Bayesian estimation by recursively approximating the complete posterior state density.

Introduction

- These methods are in some sense ***asymptotically consistent***; *i.e. if the computational efforts increase without bounds, then the approximations will converge towards the ground truth.*



PARTICLE FILTER

- **In its simplest version, the FP method consists in:**
  - Between two instants of observation, the particles move independently according to *the dynamics* of the hidden state.
  - As soon as a *new observation* is available, the particles are selected for their relevance to the new observation (quantified by the *likelihood* function).
  - Under the effect of resampling, which is the essential step of the method: the particles are concentrated automatically in the regions of interest of the state space.
  - The method is **very easy** to implement, since it is sufficient *to simulate trajectories independently* of the state cache, the interaction taking place only when resampling.

Perfect Monte-Carlo Sampling

- Many statistical algorithms require the calculation of integrals of type

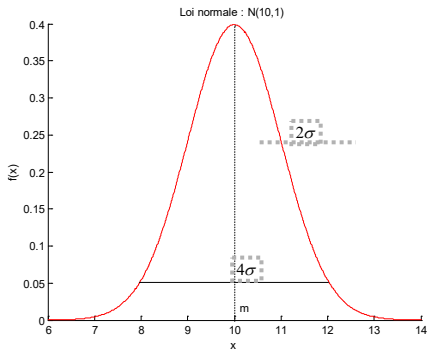
$$I = E[f(x)] = \int_{\Omega} f(x)\, p(x) dx$$

where x is a random variable on Ω , $p(x)$ its pdf and $f(\cdot)$ a function of x.

- **Example**: the mean calculation

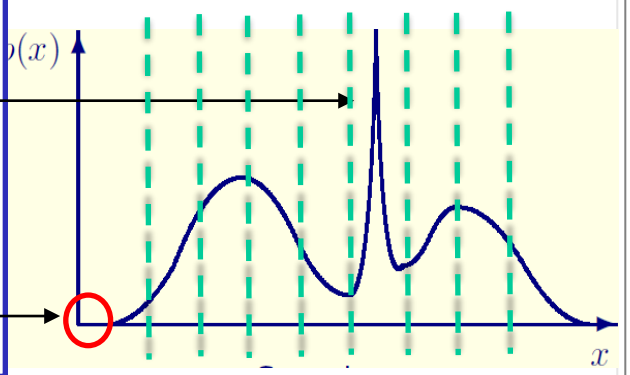
$$m_x = E[x] = \int_{\Omega} x\, p(x) dx$$

- The pdf $p(x)$ can take different forms:
  - Simple : Gaussian, uniform ,…
  - or highly complex with high dimension



To calculate these integrals, standard techniques may be inappropriate, because of their:

- ***Inaccuracies:*** *we can miss important peaks*
- ***Cost in computation time:*** *computing time is devoted to areas near-zero*

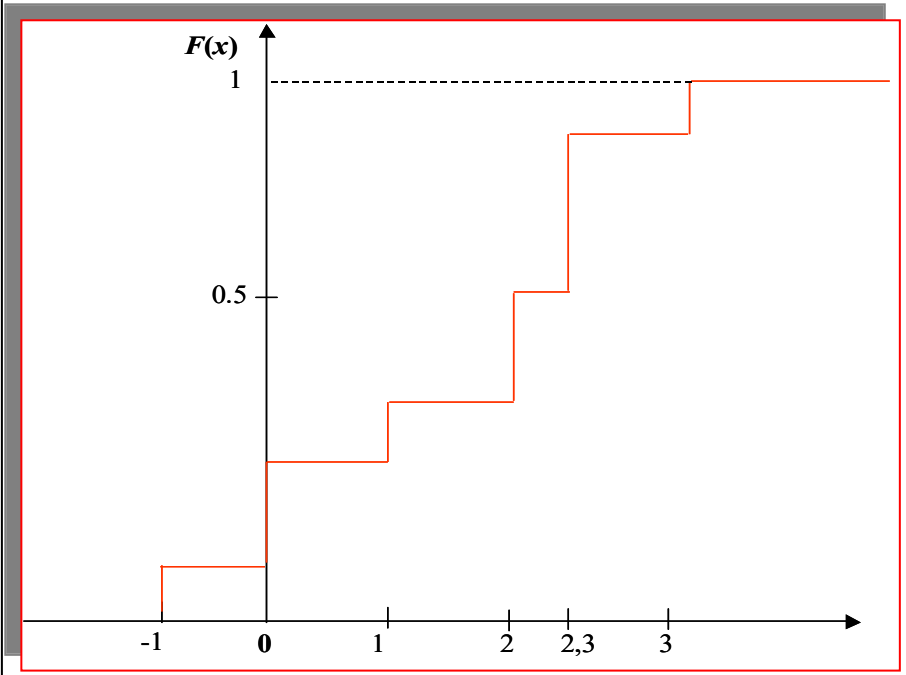


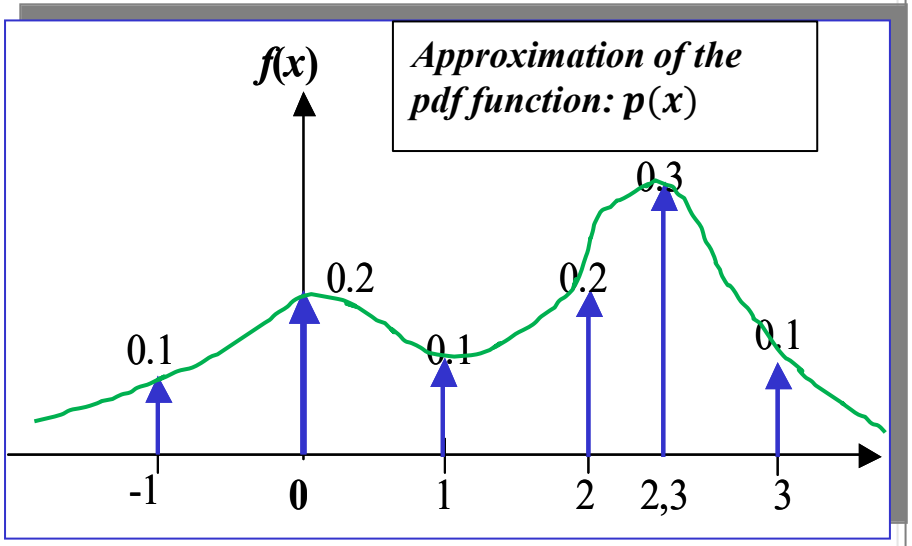
Discrete random variable

- Example: *consider the random variable X having its values in the set* $\{-1,0,1,2,2.3,3\}$ *and the corresponding probabilities are:*

$$\{0.1,\ 0.2,\ 0.1,\ 0.2,\ 0.3,\ 0.1\}$$

- *The cumulative distribution function (cdf) and density functions are given by;*





Step $F(x) = \sum_i \Pr ob(x_i) U(x - x_i)$

Dirac $f(x) = \sum_i \Pr ob(x_i) \delta(x - x_i)$

Perfect Monte-Carlo Sampling

- Suppose that we have N_p realization of the random variable x: $x^{(i)}{}_{i=1,\cdots,N_p} \sim p(x)$.
- Monte-Carlo approximation of the integral consists in this empirical estimate:

$$\hat{I} \approx \int_{\Omega} f(x)\,\hat{p}(x)dx \approx \frac{1}{N_p}\sum_{i=1}^{N_p} f(x^{(i)})$$

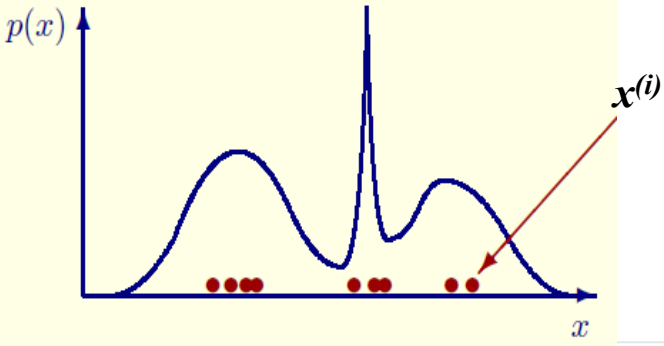
by using the following empirical approximation of the pdf:

$$\hat{p}(x) = \frac{1}{N_p}\sum_{i=1}^{N_p}\delta\left(x - x^{(i)}\right)$$

with

$$\delta\left(x - x^{(i)}\right) \text{ dirac of mass on } x^{(i)} \Rightarrow \int_{\Omega}\delta\left(x - x^{(i)}\right)dx = \begin{cases} 1 & \text{if } x^{(i)} \in \Omega \\ 0 & \text{else} \end{cases}$$

- A better **distribution of the computational** effort: generating a large number of realizations for the high weight and small number for low weight. ***The idea is to adapt the sampling to the shape of the integral***



Perfect Monte-Carlo Sampling

- The law of large numbers states that when the number of samples N_p is large, the *sample mean* tends to the expectation *almost surely* (**a.s**) :

$$\hat{f}_{N_p}(x) \approx \frac{1}{N_p}\sum_{i=1}^{N_p} f\left(x^{(i)}\right) \xrightarrow[N_p \to \infty]{} \mathbb{E}_p[f(x)]$$

- Moreover, if the posterior variance of $f(x)$ is bounded, that is

$$\text{var}[f(x)] = E[f(x)f(x)^T] = \sigma^2 < \infty$$

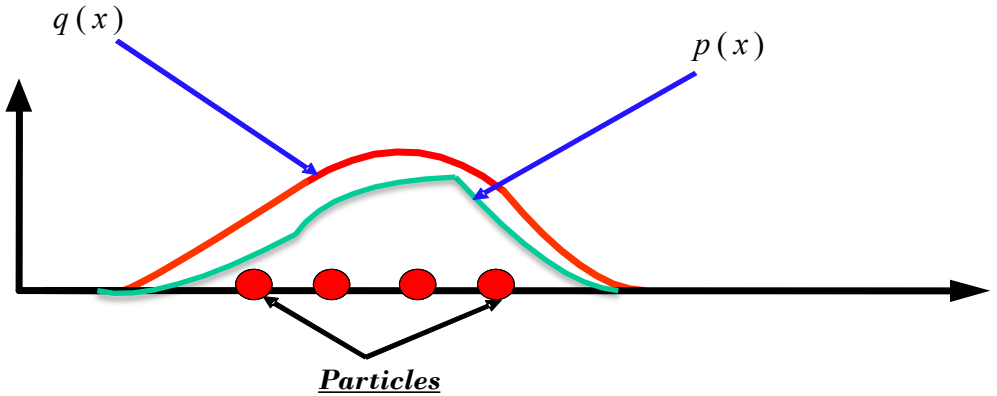
- and using the central limit theorem, we have

$$\lim_{N_p \to \infty} \sqrt{N_p}\left(\hat{I} - I\right) \to N(0, \sigma^2)$$

- The assumption underlying the above discussion is that: it is possible to obtain *independent and identically distributed, i.i.d., samples from* $p(x)$. However, in practice this assumption is *very seldom valid*.
- *In order to use the ideas sketched above, we need to be able to generate random numbers from complicated distributions* : *Importance Sampling*

Importance Sampling

- Problem: it is often impossible to sample directly from the posterior density function $p(x)$ (*Target distribution*).
- Solution: employ an alternate density and sample from a *known, easy-to-sample*, ***proposal distribution*** $q(x)$.
- The idea of importance sampling is to choose a proposal distribution $q(x)$ in place of the true probability distribution $p(x)$, which is hard-to-sample
- The exact form of this distribution is a **critical design** issue and is usually chosen in order to facilitate easy sampling. $q(x)$ has at least to be chosen with a **support** that is assumed **to cover** that of $p(x)$.



MC Importance Sampling

- Rewriting the integration problem as

$$\mathbb{E}_p[f(x)] = \int f(x)\,p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \int f(x)\,w(x)q(x)dx$$

- Then *Monte Carlo importance sampling* consist in using a number N_p of independent samples *drawn from* $q(x)$ to obtain a weighted sum to approximate the integral

$$\hat{f}_{N_p} = \frac{1}{N_p}\sum_{i=1}^{N_p} w(x^{(i)})\,f(x^{(i)})$$

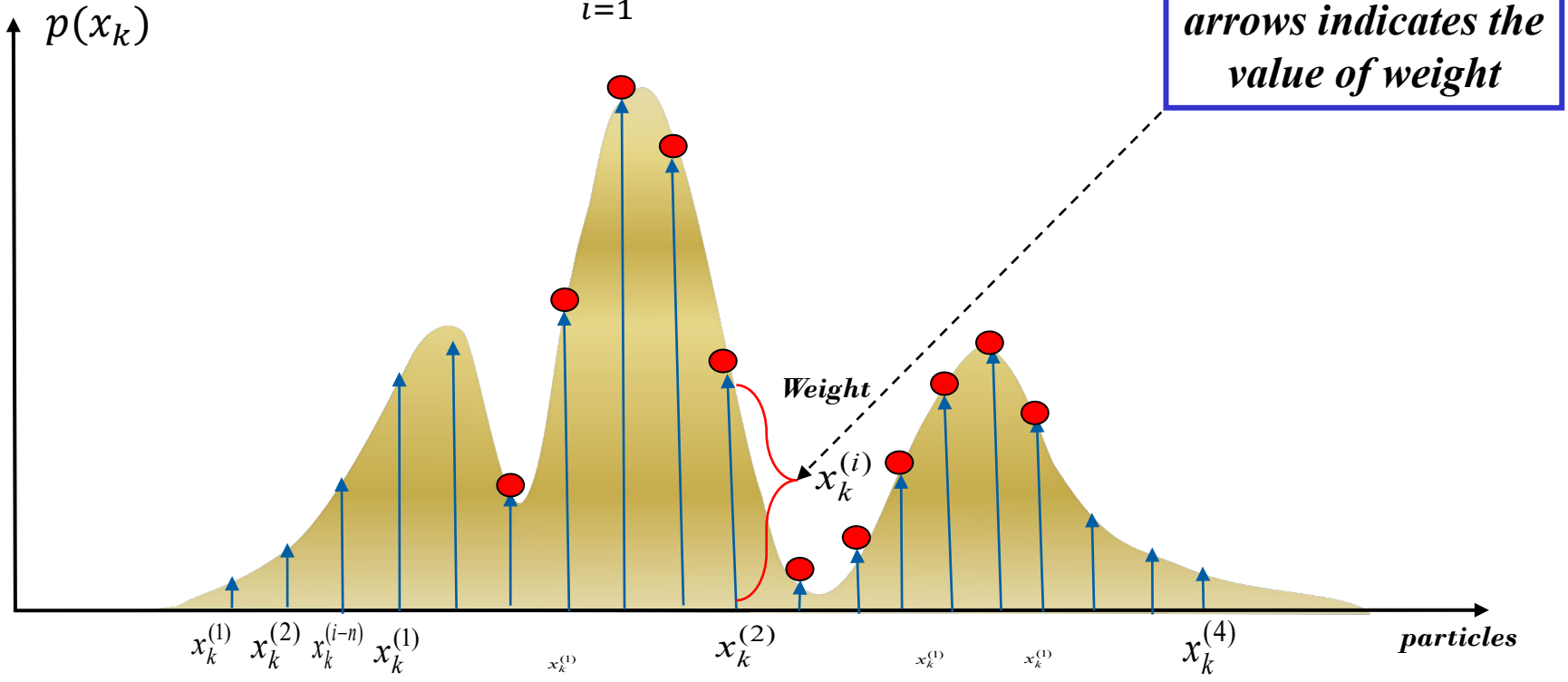
- $w(x^{(i)}) = \frac{p(x^{(i)})}{q(x^{(i)})}$ are called the ***importance weights***.
- If the normalizing factor of $p(x)$ is not known, the importance weights can be only evaluated up to *a* ***normalizing constant***. Hence, we normalize the importance weights to ensure that the *sum of the* ***weights is one*** and to obtain:

$$\sum_{i=1}^{N_p} w(x^{(i)}) = 1 \Rightarrow \hat{f}_{N_p} = \frac{\frac{1}{N_p}\sum_{i=1}^{N_p} w(x^{(i)})\,f(x^{(i)})}{\frac{1}{N_p}\sum_{j=1}^{N_p} w(x^{(j)})} = \sum_{i=1}^{N_p} \tilde{w}(x^{(i)})\,f(x^{(i)}) \quad with \quad \tilde{w}(x^{(i)}) = \frac{w(x^{(i)})}{\sum_{j=1}^{N_p} w(x^{(j)})}$$

MC Importance Sampling

- The pdf $p(x)$ is approximated by:

$$p(x_k) \approx \sum_{i=1}^{N_p} w_k^{(i)} \delta\left(x - x_k^{(i)}\right)$$



Example 1 : Importance Sampling

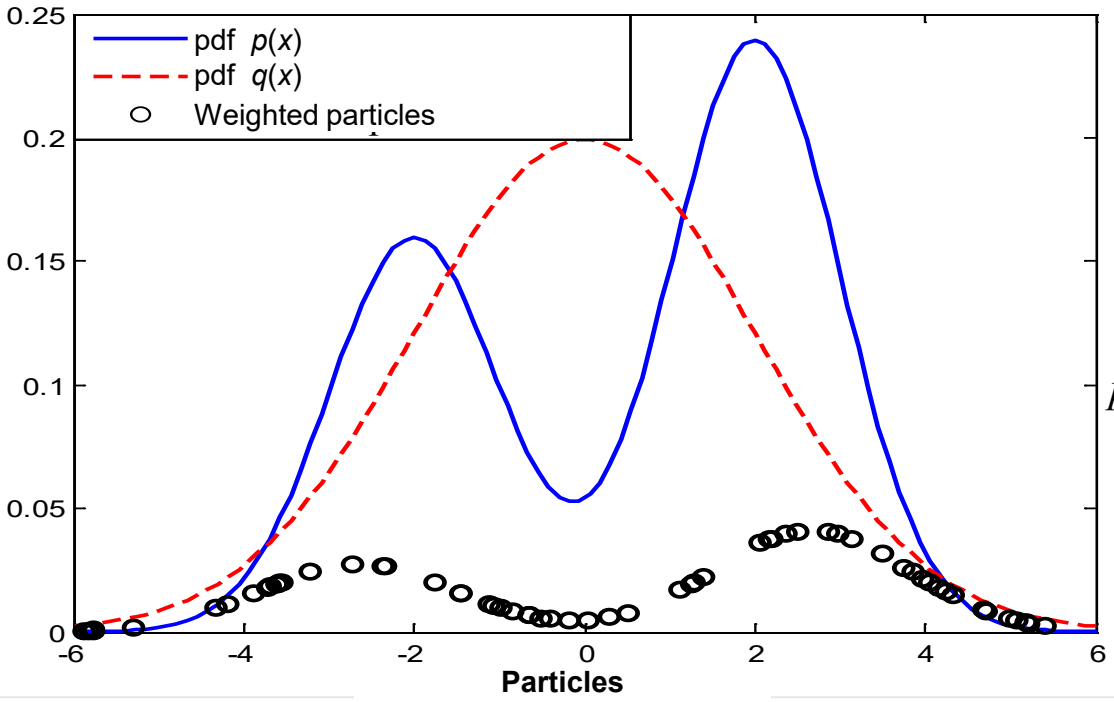
- We seek to approximate the following multi-modal pdf:

$$p(x) = 0.6 \times \mathcal{N}(x, 2, 1) + 0.4 \times \mathcal{N}(x, -2, 1)$$

- we use the proposal density given by a Gaussian density:

$$q(x) = 1 \times \mathcal{N}(x, 0, 4).$$

- *we simulate with:* $N_p = 50$



$$p(x_k) \approx \sum_{i=1}^{N_p} w_k^{(i)} \delta\left(x - x_k^{(i)}\right)$$

Sequential Importance Sampling (SIS)

- Problem: It is usually difficult to find a good proposal distribution especially in a high-dimensional space.
- Solution: A natural way to alleviate this problem is to construct the proposal distribution sequentially, which is the basic idea of *sequential importance sampling* (SIS):

$$p(x_{0:k}/Y_k) = p(x_{0:k-1}/Y_{k-1})\frac{p(y_k/x_{0:k})p(x_{0:k}/x_{0:k-1})}{p(y_k/Y_{k-1})}$$

- A recursive estimate for the importance weights can then be derived as follows:

$$w_k^{(i)} = \frac{p(x_{0:k}^{(i)}/y_{1:k})}{q(x_{0:k}^{(i)}/y_{1:k})} = \frac{p(y_k/x_k^{(i)})p(x_k^{(i)}/x_{k-1}^{(i)})p(x_{0:k-1}^{(i)}/y_{1:k-1}^{(i)})}{q(x_k^{(i)}/x_{0:k-1}^{(i)}, y_{1:k})q(x_{0:k-1}^{(i)}/y_{1:k-1})}$$

$$= w_{k-1}^{(i)}\frac{p(y_k/x_k^{(i)})p(x_k^{(i)}/x_{k-1}^{(i)})}{q(x_k^{(i)}/x_{0:k-1}^{(i)}, y_{1:k})}$$

$$= \boldsymbol{w_{k-1}^{(i)}}\frac{p(y_k/x_k^{(i)})p(x_k^{(i)}/x_{k-1}^{(i)})}{q(x_k^{(i)}/x_{0:k-1}^{(i)}, y_{1:k})}$$

Choice of Proposal Distribution

- The choice of proposal function is one of the most critical design issues in importance sampling algorithms and several solutions have been proposed:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(y_k/x_k^{(i)})p(x_k^{(i)}/x_{k-1}^{(i)})}{q(x_k^{(i)}/x_{0:k-1}^{(i)}, y_{1:k})}$$

Importance density

- An optimal choice of $q(x)$ could be the one that minimizes the variance of the proposal weights (*Doucet et al.*), It is given by this *optimal proposal distribution:*

$$q\left(x_k^{(i)}/x_{0:k-1}^{(i)}, y_{1:k}\right) = p\left(x_k^{(i)}/x_{k-1}^{(i)}, y_k\right)$$

- If now we choose the proposal as the *transition prior*:

$$q(x_k^{(i)}/x_{0:k-1}^{(i)}, y_{1:k}) = p(x_k^{(i)}/x_{k-1}^{(i)})$$

- Even it not incorporating the most *recent observations*, it is usually easier to implement, and it is the most popular choice of proposal distribution:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p\left(y_k/x_k^{(i)}\right)\cancel{p\left(x_k^{(i)}/x_{k-1}^{(i)}\right)}}{\cancel{p\left(x_k^{(i)}/x_{k-1}^{(i)}, y_k\right)}} \Rightarrow w\left(x_{0:k}^{(i)}\right) = w\left(x_{0:k-1}^{(i)}\right)p\left(y_k/x_{k-1}^{(i)}\right)$$

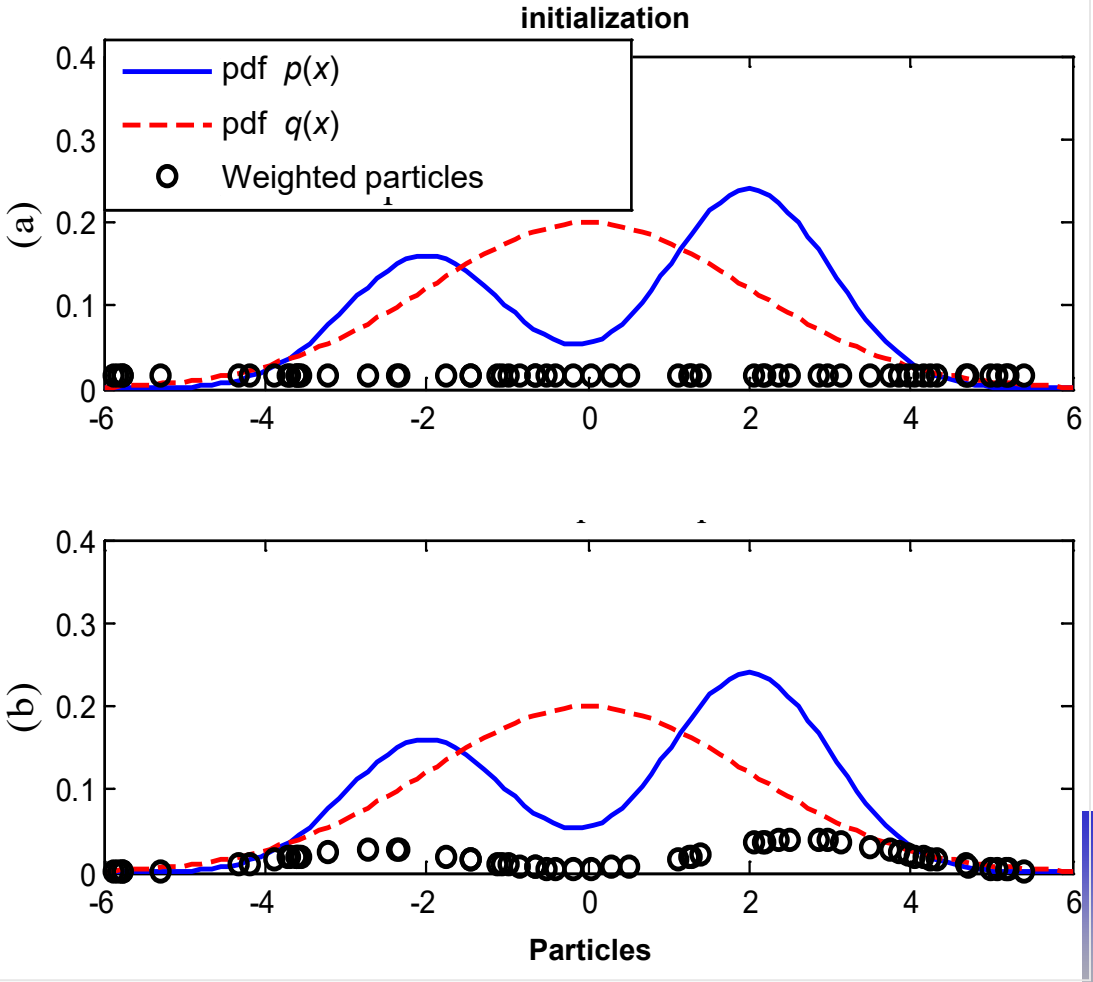
Exemple 2 : Sequential Importance Sampling

- Resume example 1 with the same simulations conditions

- **(a)** At initial state, all the particle have the same weight

- **(b)** presence of regions where the contribution of the particles is negligible in the estimation of $p(x)$.

This requires to concentrate particles in regions of high probability density and thus the use of ***resampling*** step



Weight degeneracy

- One of the drawbacks resulting with SIS algorithm is called *weight degeneracy or sample impoverishment* phenomena : *after some iterations, only very few particles have non-zero importance weights.*
- An intuitive solution is to multiply the particles with high normalized importance weights, and discard (*kill relative to* $1/N_p$) the particles with low normalized importance weights, which can be done in the ***resampling step***.
- To monitor how bad is the **weight degeneration**, we uses as measure the so-called **effective sample size:**

$$N_{eff} = \frac{N_p}{1 + Var\left(\tilde{w}\left(x_{0:k}^{(i)}\right)\right)} \leq N_p$$

- In practice, the true N_{eff} is not available, thus its estimate, is alternatively used

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p}\left(\tilde{w}\left(x_{0:k}^{(i)}\right)\right)^2}$$

- *When this estimate of the effective sample size is below a predefined threshold the resampling procedure is preformed*
- *Performing resampling at each step time can lead also to an **impoverishment** of diversity of weights*

Resampling methods

- The resampling step is aimed to *eliminate the samples* with *small importance* weights and *duplicate* the samples with *high* weights.
- The sampling-importance resampling (SIR) is motivated from the **Bootstrap and jackknife techniques**. The intuition of bootstrapping is to evaluate the properties of an estimator through the *empirical cumulative distribution function* (*cdf*) of the samples instead of the true *cdf*.
- Resampling schedule can be deterministic or dynamic.
  - In deterministic framework, resampling is taken at every k time step (usually $k = 1$).
  - In a dynamic schedule, resampling is taken only when the variance of weights is over a *threshold*. For example, using the *effective sample size*, we perform sampling when

$$\widehat{N}_{eff} \leq \frac{N_p}{2} or \frac{N_p}{3}$$

- Algorithms: Several selection schemes have been proposed in the literature, including *sampling-importance resampling* (SIR) or *Multinomial resampling*, *residual resampling* and *minimum variance sampling*

Multinomial resampling

- Multinomial resampling: uniformly generates N_p new independent particles from the old particle set.

- SIR involves mapping the Dirac random measure $\{x^{(i)}, \widetilde{w}^{(i)}\}$ into an equally weighted random measure $\{x^{(i)}, 1/N_p\}$:

$$\left\{x_k^{(i)}, \widetilde{w}_k^{(i)}\right\} \rightarrow \left\{x_k^{(j)}, 1/N_p\right\}$$

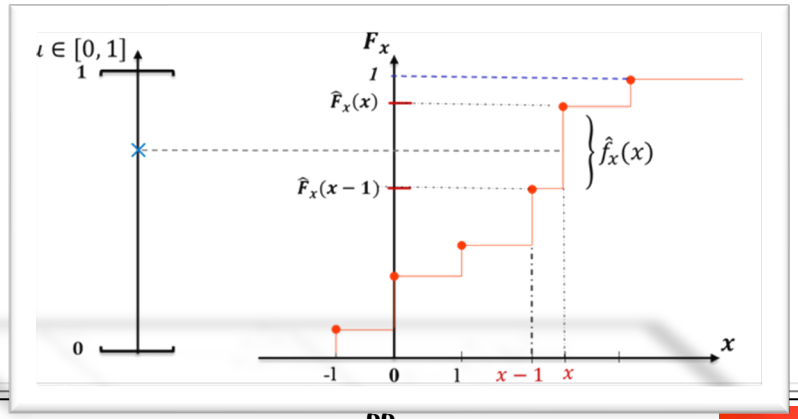
- Here, the idea is to select the new particles by comparing an ordered set of uniformly distributed random numbers $U(0,1)$ to the cumulative distribution sum of the normalized importance weights.

How to generate a discrete random variable

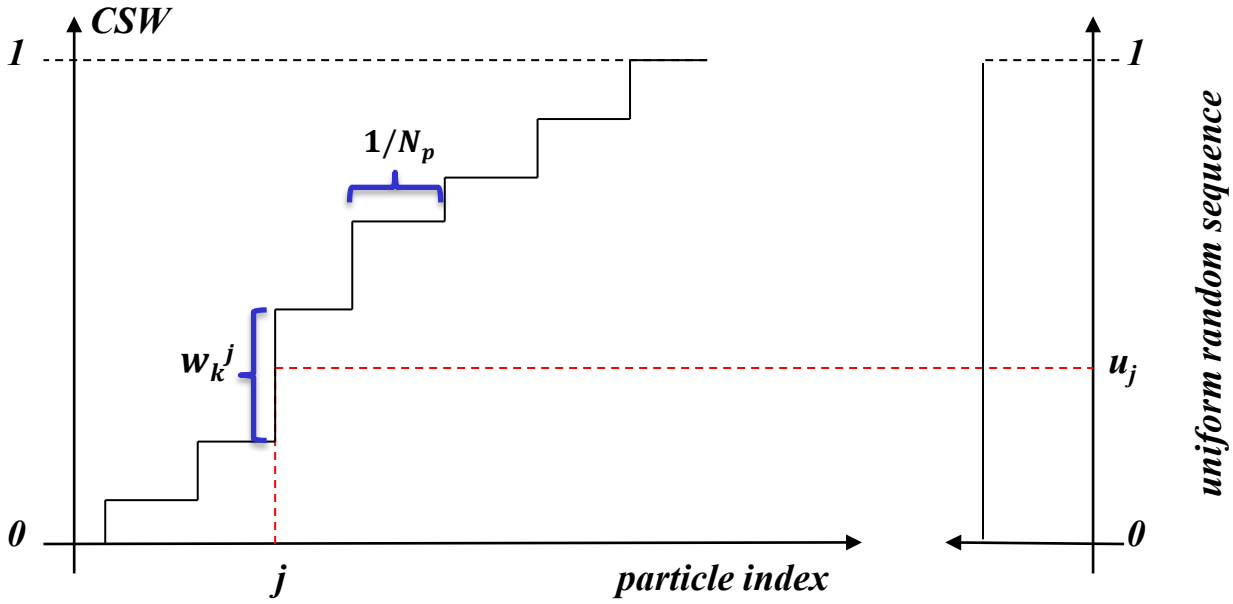
- Most mathematical libraries have functions that generate a uniform distribution $f_u(u)$ over the interval [0,1] like the Matlab function *rand* :

$$f_u(u) = \begin{cases} 1 & \textbf{if } u \in [\mathbf{0\ 1}] \\ 0 & otherwise \end{cases}$$

- If now we wish to generate a discrete random variable $(r.v)$ with an arbitrary PDF, we can do so by inferring from this uniform PDF
- Procedure: Let a desired PDF $\hat{f}_x(x)$ associated with the $r.v$,
  - The cumulative distribution function (cdf) is calculated: $\hat{F}_x(x) = \sum_{\bar{x}=-\infty}^{x} \hat{f}_x(\bar{x})$ (for continuous case: $\hat{F}_x(x) = \int_{-\infty}^{x} f_x(\bar{x})d\bar{x}$)
  - We generate the $r.v.$ $\boldsymbol{u}$ of the uniform distribution $f_u(u)$
  - We calculate the value of the $r.v.$ $\boldsymbol{x}$ such that $\hat{F}_x(x-1) < u \leq \hat{F}_x(x)$
- It is then shown that the $r.v.$ $\boldsymbol{x}$ is a random variable with a PDF $f_x(x) = \hat{f}_x(x)$



Multinomial resampling



Resamping algorithm: **Multinomial resampling**

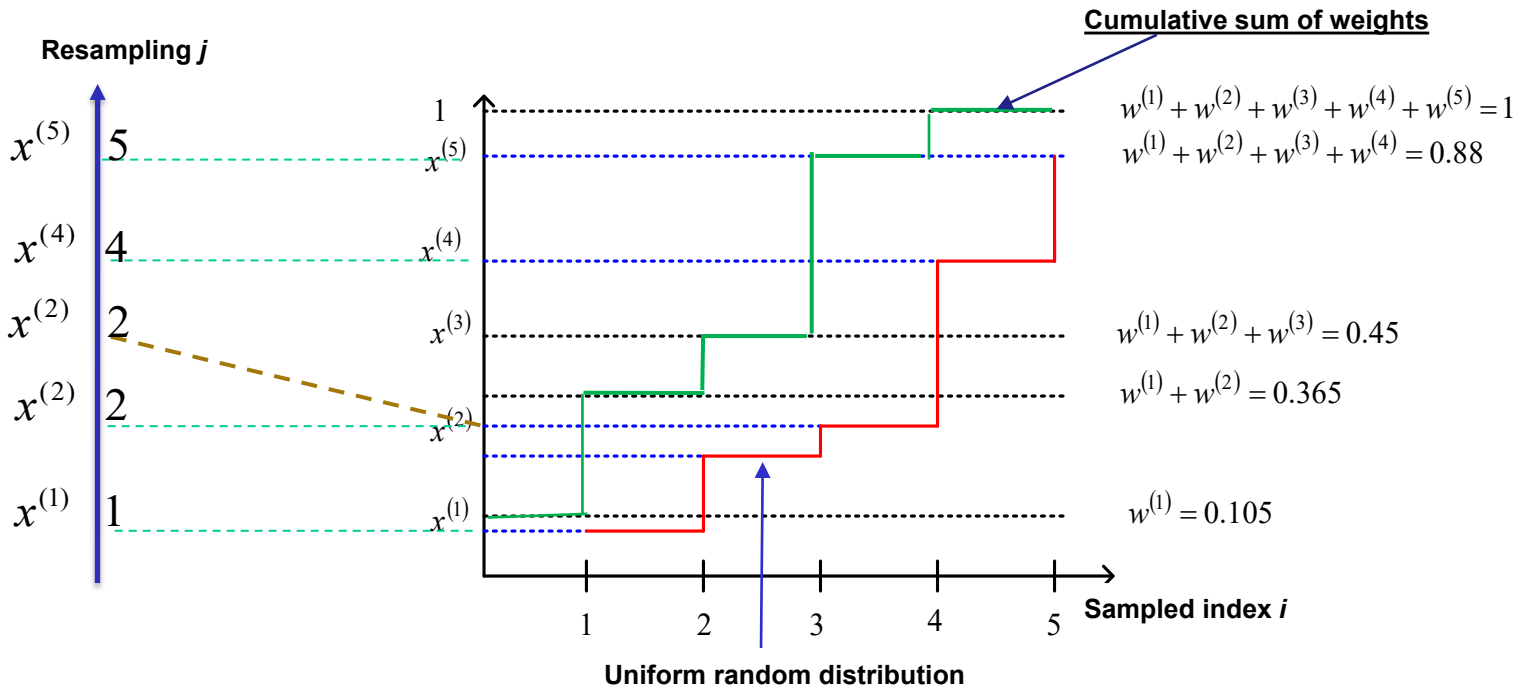
- Produce a uniform distribution $u \sim \mathcal{U}(0,1)$, *then construct the cumulative distribution function (cdf) for the importance weights,* $s_i = \sum_{i=1}^{j} \tilde{w}^j$.
- *Find* s_i *tel que* $s_{i-1} \leq u_j \leq s_i$, *the particle with index* i *is chosen.*
- *Given* $\{x^{(i)}, \tilde{w}^{(i)}\}$ *for* $j = 1, \cdots, N_p$ *generate new samples* $x^{(j)}$ *by duplicating* $x^{(i)}$ *according to the associated s* $\tilde{w}^{(i)}$.
- *reset,* $w^{(i)} = \frac{1}{N_p}$.

Example 3: Multinomial resampling

Consider five particles $x^{(i)}$, $i = 1, \cdots, 5$ which are respectively associated weights:

$$\{w^{(i)}\}_{i=1}^{5} = \{0.105, 0.26, 0.085, 0.43, 0.12\}$$

and suppose that we have five realizations of a uniform random variable $u^{(i)} \in [0\ 1]$, $i = 1, \cdots, 5$ for example $\{u^{(i)}\}_{i=1}^{5} = \{0.07,\ 0.27, 0.32, 0.68, 0.88\}$. These uniform variables are compared to the cumulative sum of weights s_i.

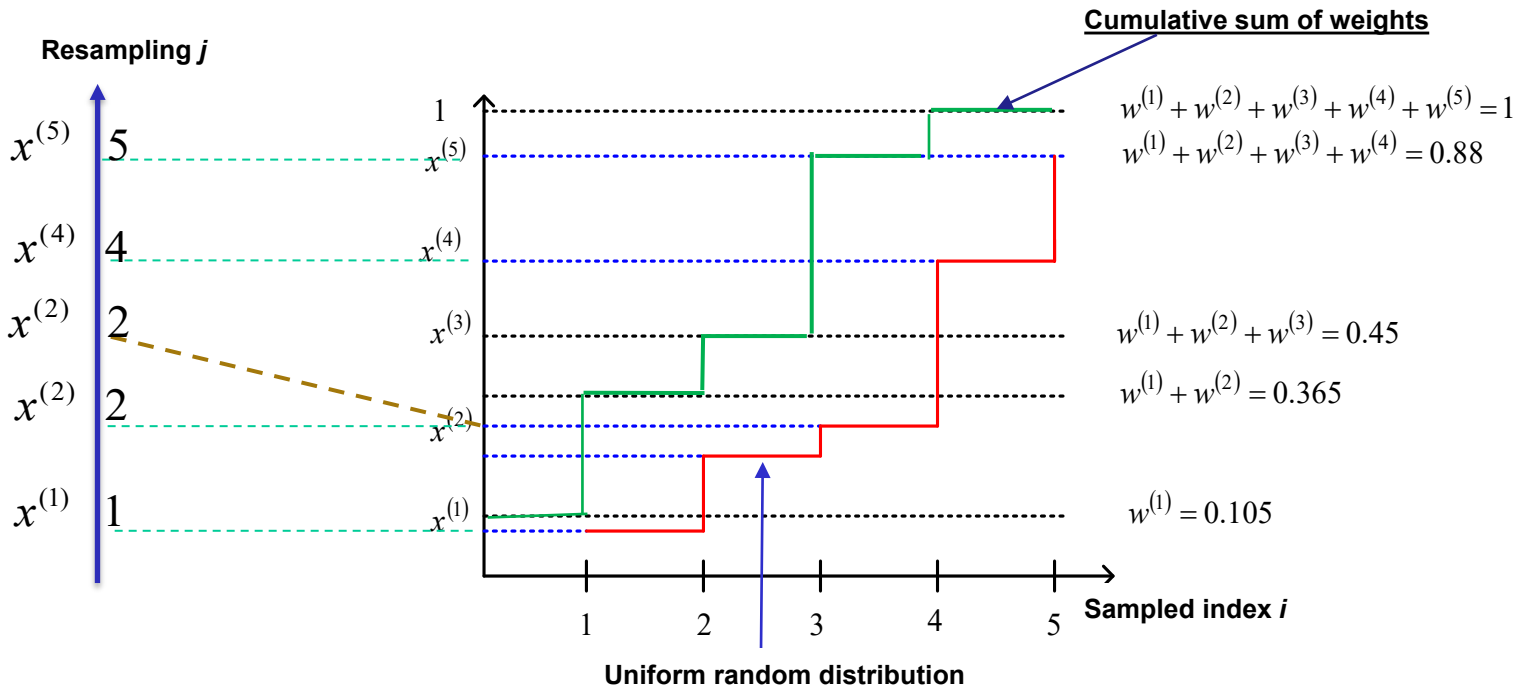


Example 3: Multinomial resampling

When the generated random variable $u^{(i)}$ **is in the interval** $[s_i, s_{i-1}]$ then the particle with the weight $w^{(i)} = s_i - s_{i-1}$ is maintained. The new particles are:

$$\{x'^{(i)}\}_{i=1}^{5} = \{x^{(1)}, x^{(2)}, x^{(2)}, x^{(4)}, x^{(5)}\}$$

The sequence of particles shows that the particle $x^{(2)}$ is duplicated at the expense of the particle $x^{(3)}$ which is eliminated. Then, the weights are reset $\frac{1}{N_p}$.

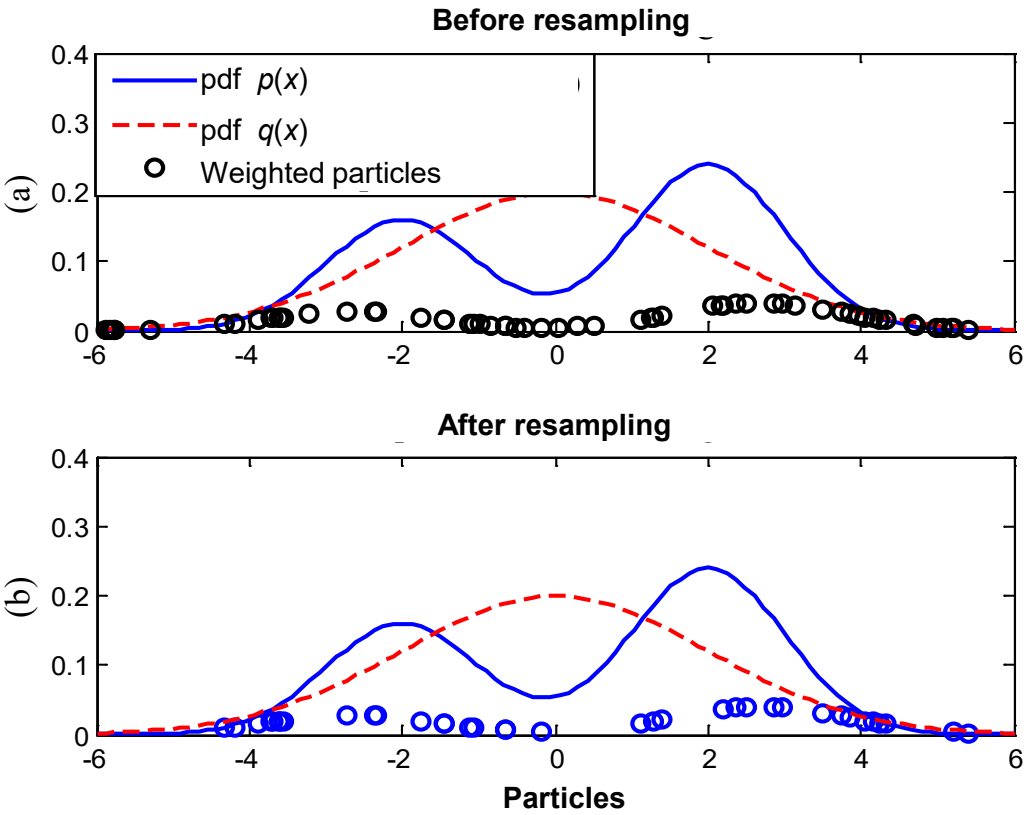


Exemple 4 : Sequential Importance Resampling

- Resume the example 2 with the same conditions of simulation. Then introduce into the SIS algorithm a multinomial resampling stage:

- **(a)** presents the weighted particles before the step of resampling

- **(b)** shows the results obtained after resampling. The particles belonging to the low probability areas have been eliminated in favor of particles belonging to areas of high probability. By concentrating the particles in these regions of high probability estimate of the state becomes more accurate.



The estimates ….

- In the Bayesian framework the optimal estimator is given by:

$$\hat{x}_{k/k} = E(x_k/Y_k) = \int x_k p\,(x_k/Y_k) dx$$

- Using the following **estimate** of the probability density function, results

$$p(x_k/Y_k) \approx \sum_{i=1}^{N_p} \tilde{w}_k^{(i)} \delta\left(x_k - x_k^{(i)}\right) \quad \Rightarrow \quad \hat{x}_{k/k} \approx \sum_{i=1}^{N_p} \tilde{w}_k^{(i)} x_k^{(i)}$$

- These point-mass estimates can well approximate any general arbitrarily distribution. It's *limited only by the number of particles*.
- Similarly, an estimate of the **covariance** is obtained using

$$P_{xx} \approx \sum_{i=1}^{N_p} \tilde{w}_k^{(i)} (x_{k/k}^{(i)} - \hat{x}_{k/k})\,(x_{k/k}^{(i)} - \hat{x}_{k/k})^T$$

- These formula show how the estimates are affected by the information in the normalized importance weights. *The more likely a certain particle is, the more it influences the estimate, which is a quite reasonable fact*.

Implementations steps:

Particle Filtering (SIR or Boostsrap Filter)

- **Initialization**: $k=0$
  - *For* $i = 1, \cdots, N_p$, sample $x_0^{(i)} \sim p(x_0)$ with $w_0^{(i)} = \frac{1}{N_p}$

- *For* $k = 1,2, \cdots, K$ *(data length)*

1. ***Importance sampling step***
   - Propagate particles : *For* $i = 1, \cdots, N_p$, draw particle $x_k^{(i)} \sim p\left(x_k / x_{k-1}^{(i)}\right)$
   - Evaluate the importance weights: *For* $i = 1, \cdots, N_p$ calculate up to a normalizing constant

$$w\left(x_k^{(i)}\right) \propto w\left(x_{k-1}^{(i)}\right) p\left(y_k / x_k^{(i)}\right)$$

   - Normalize: *For* $i = 1, \cdots, N_p$ normalize the importance weights

$$\tilde{w}\left(x_k^{(i)}\right) = \frac{w\left(x_k^{(i)}\right)}{\sum_{j=1}^{N_p} w\left(x_k^{(j)}\right)}$$

Propagate particles through the process model:

$$x_k^{(i)} = f_k\left(x_{k-1}^{(i)}, \eta_k\right)$$

Update particle weights based on likelihood:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(y_k \mid x_k^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)})}{q(x_k^{(i)} \mid x_{0:k-1}^{(i)}, Y_k)}$$

$$q(x_k^{(i)} \mid x_{0:k-1}^{(i)}, Y_k) \approx p(x_k^{(i)} \mid x_{k-1}^{(i)})$$

$$\Rightarrow w_k^{(i)} = w_{k-1}^{(i)} p(y_k \mid x_k^{(i)})$$

Compute likelihood of measurement w.r.t. each particle:

$$y_k = h_k(x_k^{(i)}, \omega_k) \leftrightarrow p(y_k \mid x_k^{(i)})$$

Implementation steps

We can also perform estimation before sampling step (SIS)

2. ***Resampling step***

- Measure of degeneracy: effective sample size

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p}\left(\tilde{w}\left(x_{0:k}^{(i)}\right)\right)^2}$$

- Resample whenever $\hat{N}_{eff} < N_{threshold}$
- Resample with appropriate algorithm (e.g. Multinomial resamplig)
- Normalize: new set of particles have same statistical properties

The important weights of the surviving particles are not necessarily reset to $1/N_p$

$$\{x_k^i, w_k^i\} \Leftrightarrow \{x_k^j, 1/N_p\}$$

3. ***State estimation***

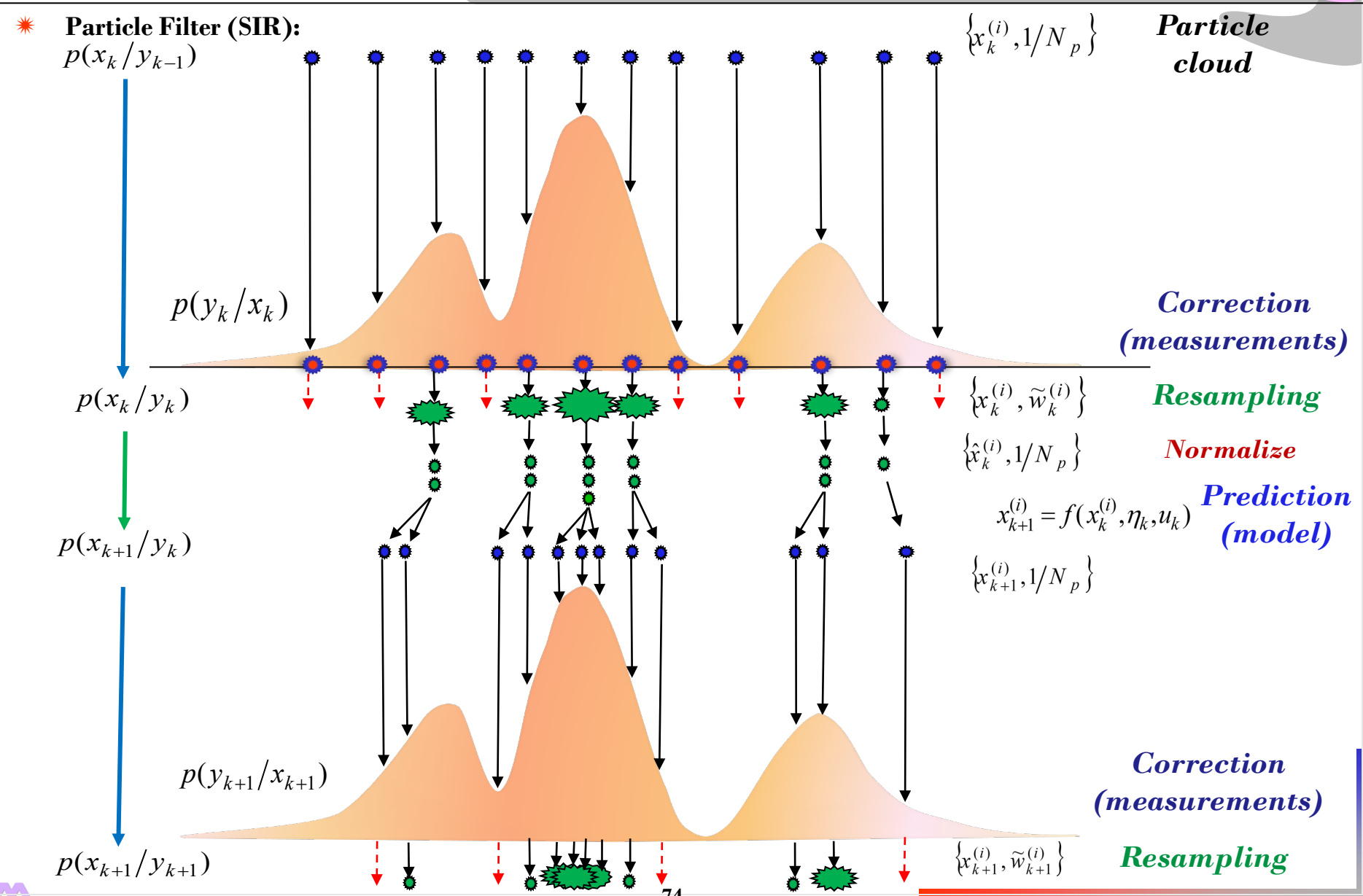
- Use set of samples $\left(x_k^{(i)}, \frac{1}{N_p}\right)_{i=1,\cdots,N_p}$ *that* can be used to approximate the posterior distribution $p(x_k/y_k)$:

$$\hat{p}(x_k/Y_k) = \frac{1}{N_p}\sum_{i=1}^{N_p}\delta(x_k - x_k^{(i)})$$

- The estimated state is then given using the expectation

$$\hat{x}_k = E(x_k/Y_k) = \frac{1}{N_p}\sum_{i=1}^{N_p} x_k^{(i)}$$

PARTICLE FILTER



Choice of Proposal Distribution

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(y_k / x_k^{(i)}) p(x_k^{(i)} / x_{k-1}^{(i)})}{q(x_k^{(i)} / x_{0:k-1}^{(i)}, y_{1:k})}$$

- Particles are drawn from this distribution $q(x)$
- Importance weights are evaluated using $q(x)$
- The support of $q(x)$ must include support of the *posterior distribution*
- $q(x)$ must include recent observations
- $q(x)$ equal to the *prior* is easier to implement but results in higher Monte-Carlo variation and it not incorporating the most recent observations:

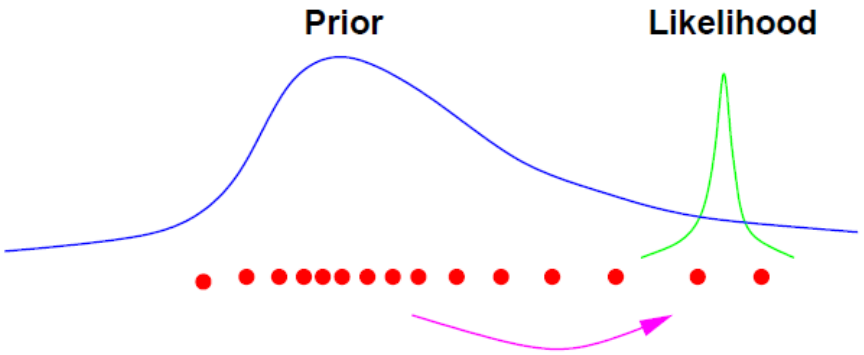
$$q(x_k^{(i)} / x_{0:k-1}^{(i)}, y_{1:k}) = p(x_k^{(i)} / x_{k-1}^{(i)}) \Rightarrow w(x_{0:k}^{(i)}) = w(x_{0:k-1}^{(i)}) p(y_k / x_{k-1}^{(i)})$$

- If we fail to use the latest available information contained in current observation to propose new values for the states, only a few particles will have significant importance weights when their *likelihood* are evaluated !

Including the most current observation into the proposal distribution, allows us to move the samples in the prior to regions of high likelihood.

This is always a big concern if the transition prior and the observation likelihood densities are both peaked and have very little support overlap.

Transition prior could be a bad choice



Prior choice will work well when $p(y/x)$ is vague but will fail when it is peaky

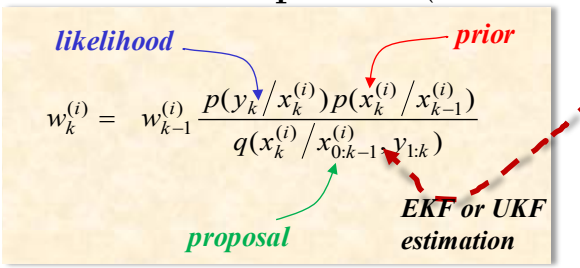
Choice of Proposal Distribution

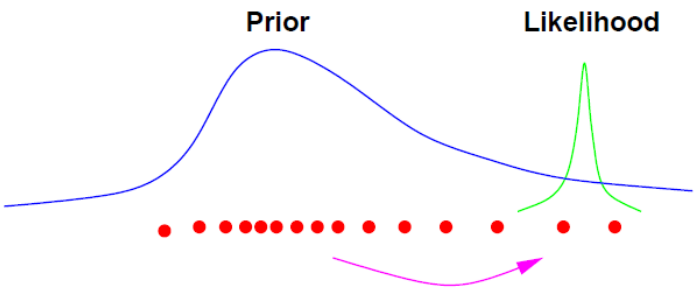
- The idea is to move the particles toward regions of increased likelihood as dictated by the most recent observation y_k . This can be done by choosing a proposal distribution that is conditioned on y_k.
- ***Solution**: $q(x)$ is Approximate by a tractable single Gaussian distribution generated by a Gaussian approximate using a recursive Bayesian estimation such **Kalman.***
- Within the PF framework, a tractable way is to use a separate **EKF** or **UKF** to generate a Gaussian proposal distribution for each particle:

$$q(x_k^{(i)} / x_{0:k-1}^{(i)}, y_{1:k}) = N(\hat{x}_k^{(i)}, m_{x_k^{(i)}}, P_{x_k^{(i)}})$$

EKPF* *or* *UKPF

- ***Procedure***:
  - At time k, use the **EKF** equations, with the new observed data, to compute the mean and covariance of the importance distribution for each particle from the previous time step $(k-1)$.
  - Redraw the i-th particle (at time k) from this new updated distribution.





- **Example:**
  - Consider this highly nonlinear SISO system (Example inspired from the benchmark of Gordon *et al.* 1993)

$$x_k = 0.4x_{k-1} + 0.001\frac{k}{1+x_{k-1}^2} + 0.1e^{(k/30)} + \omega_k$$

$$y_k = x_k + v_k$$

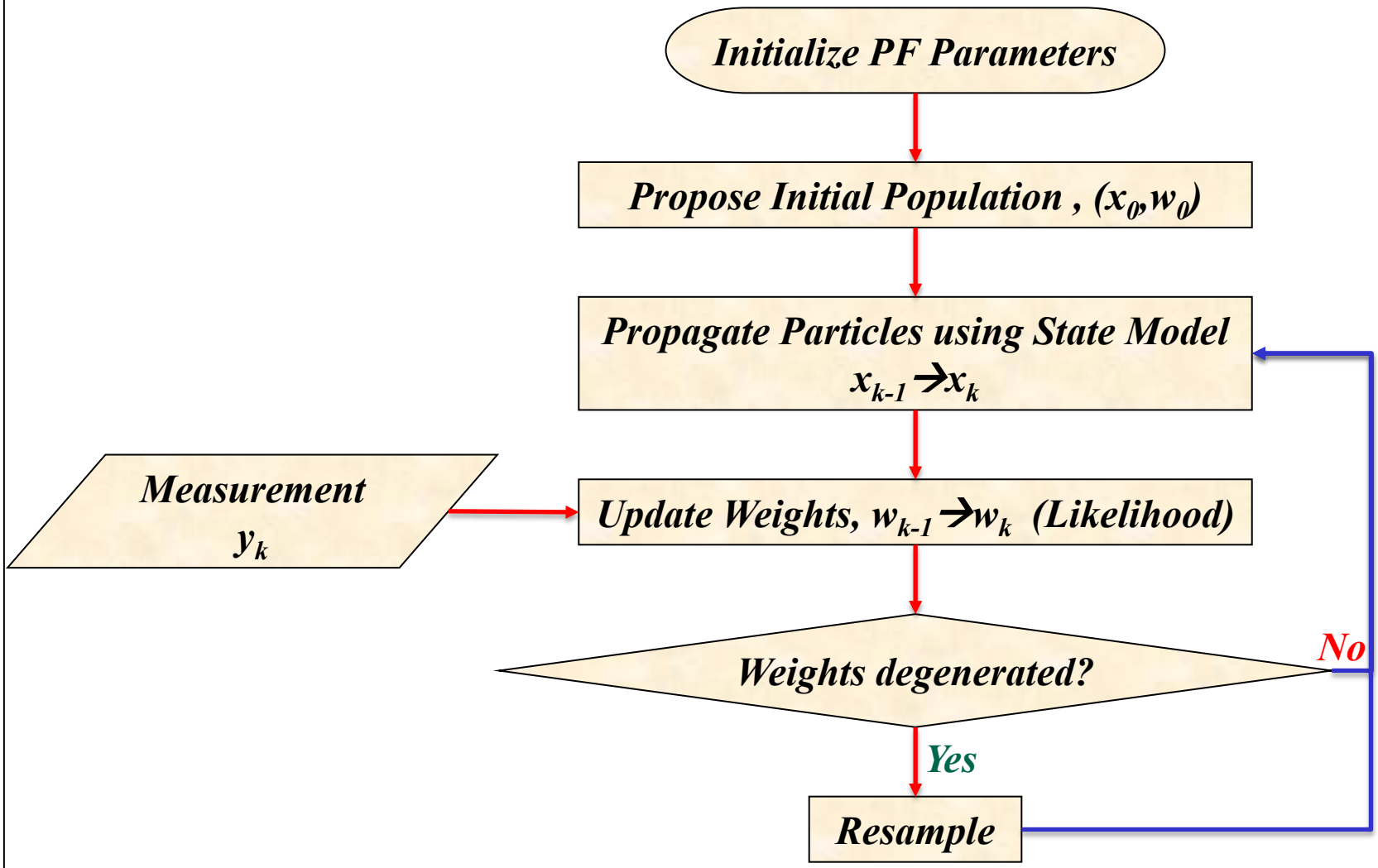
  - Noises are supposed Gaussian with zero means and covariances:

$$Q = 0.5$$
$$R = 0.05$$

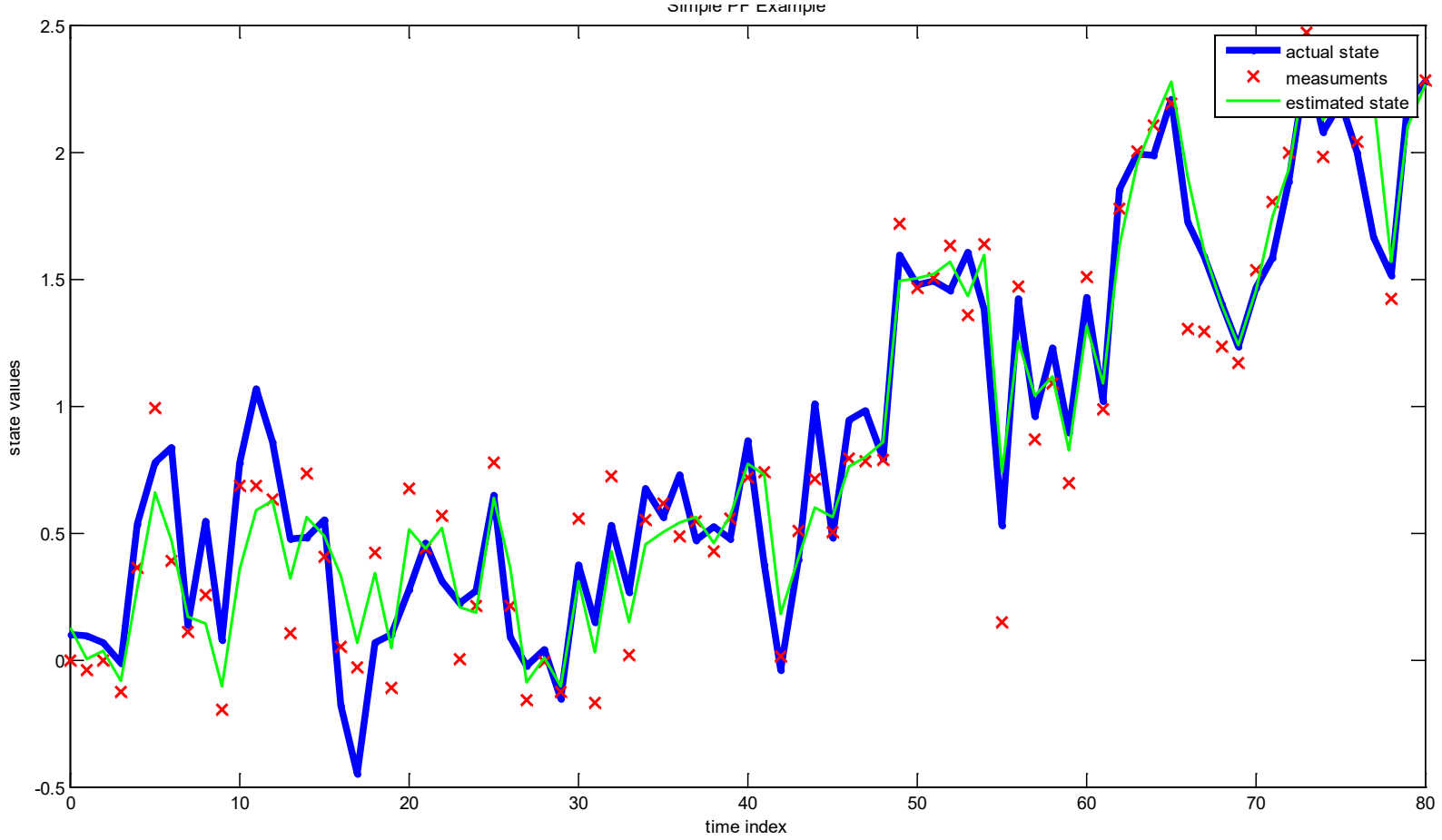
  - For the FP filter, we will choose:

$$N_p = 20$$
$$x_0 = 0.1$$

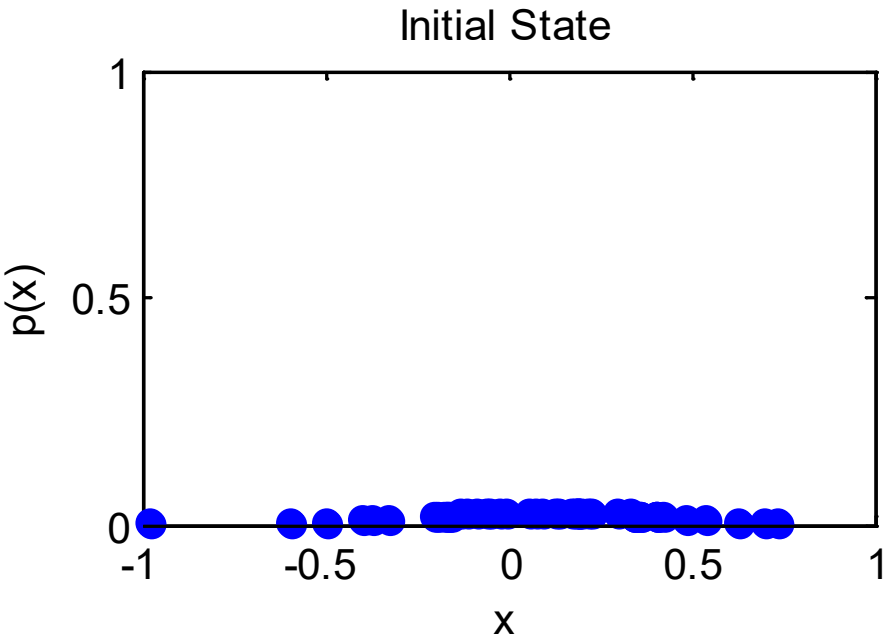
PF Algorithm



State estimation



- **Particle filter in action**



Change of variables

- Let x et y two continuous random variables and consider g a continuously differentiable and strictly monotonic function that relates them: $x = g(y)$
- If $f_y(y)$ is given, then we can show that $f_x(x)$ is given by:

$$f_x(x) = \frac{f_y(y)}{\frac{\partial g}{\partial y}(y)}$$

- The proof is straightforward considering $\Pr(y \in [\overline{y}, \overline{y} + \Delta y]) \to f_y(\overline{y})\Delta y$
- Indeed, in this case we have (as $\Delta y \to 0$):

$$g(\overline{y} + \Delta y) = g(\overline{y}) + \frac{\partial g}{\partial y}(\overline{y})\Delta y = \overline{x} + \Delta x \quad \Rightarrow \quad \Delta x := \frac{\partial g}{\partial y}(\overline{y})\Delta y$$

$$\Rightarrow \Pr(x \in [\overline{x}, \overline{x} + \Delta x]) = \Pr(y \in [\overline{y}, \overline{y} + \Delta y]) \to f_x(\overline{x})\Delta x = f_y(\overline{y})\Delta y \Rightarrow f_x(\overline{x}) = \frac{f_y(\overline{y})}{\frac{\partial g}{\partial y}(\overline{y})}$$

- Example: *The likelihood function !* :
- Suppose that $f_\omega(\omega)$ is given and

$$y = g(x, \omega) \qquad and \qquad \omega = h(y, x)$$

Then we can evaluate the following conditional pdf:

$$f_y(y/x) = f_\omega(h(y,x)) / \frac{\partial g}{\partial \omega}(\omega)$$