Rare Event Simulation for a Static Distribution

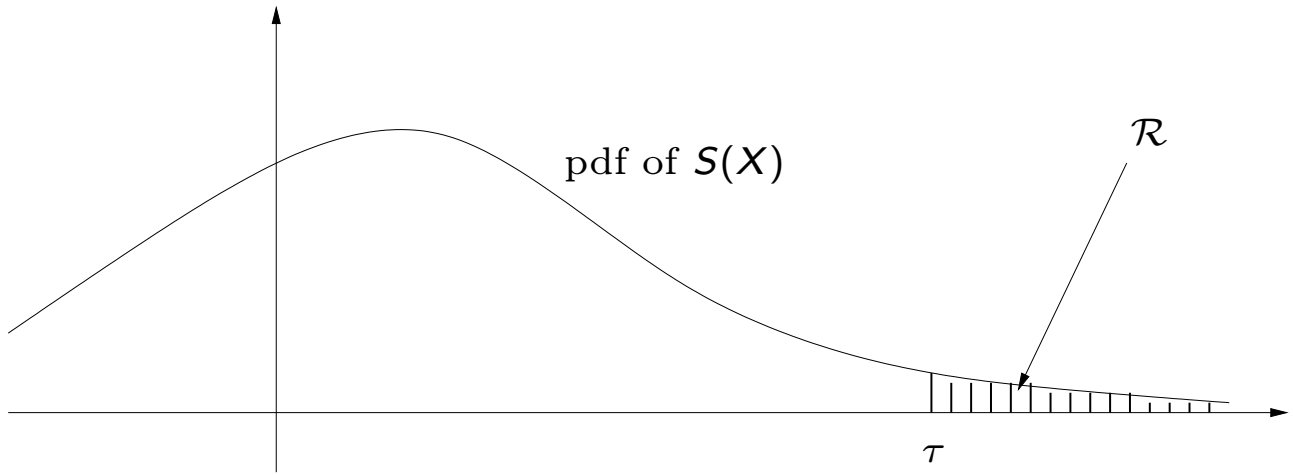
V. Bahuon F. Cérou[1] T. Furon[1] A. Guyader[2]

[1]INRIA Rennes

[2]Université Rennes 2

Journées MAS de la SMAI
Rennes, 27-29 août 2008

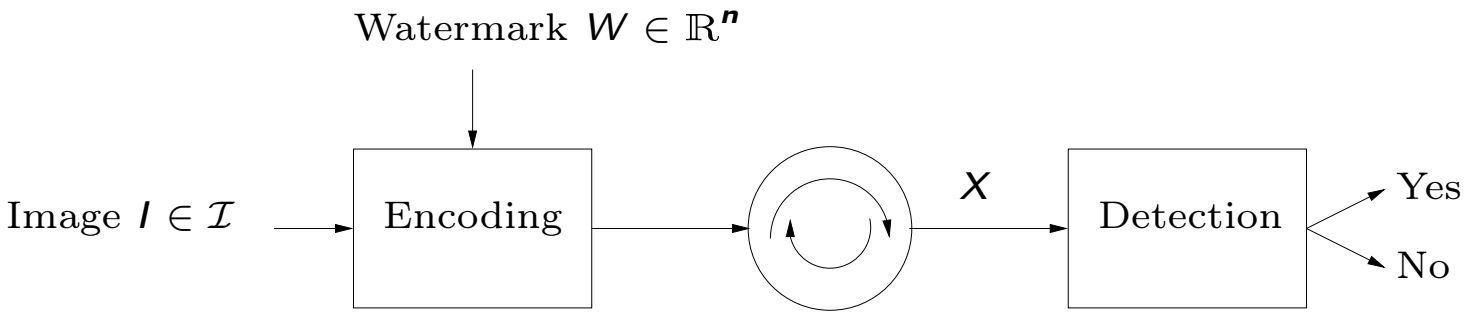
The Model



- Let $X \in E$ be a random vector and $S : E \to \mathbb{R}$ a score.
- **Goal** : Estimate $\alpha = \mathbb{P}(S(X) > \tau) < 10^{-6}$.
- **Framework** : we can only simulate $X \sim \mu$ and compute S at each point, but any analytical study is excluded.

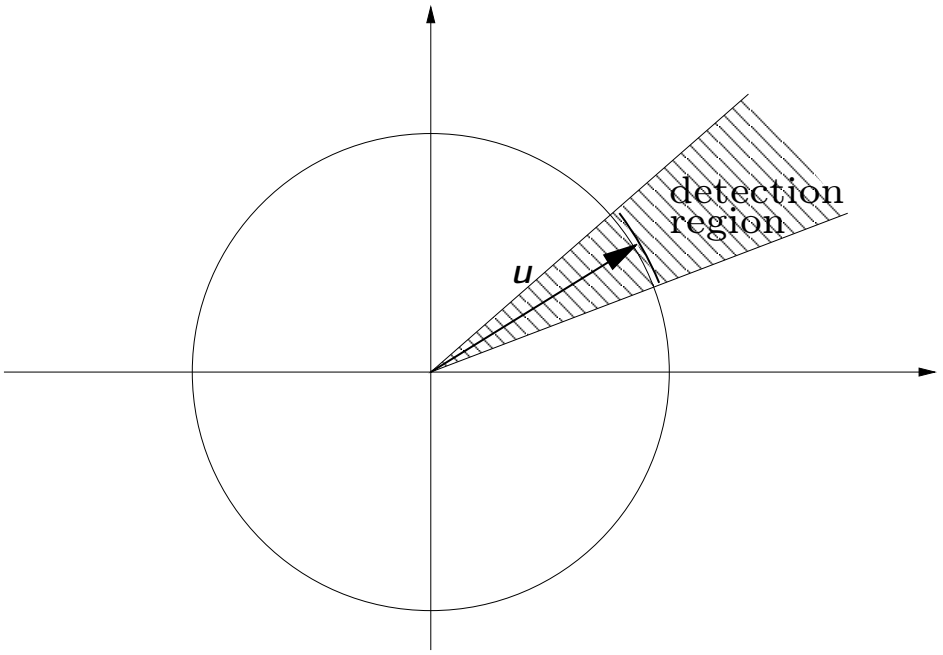
$\Rightarrow$ **Monte-Carlo methods**.

Toy Example : Zero-Bit Watermarking



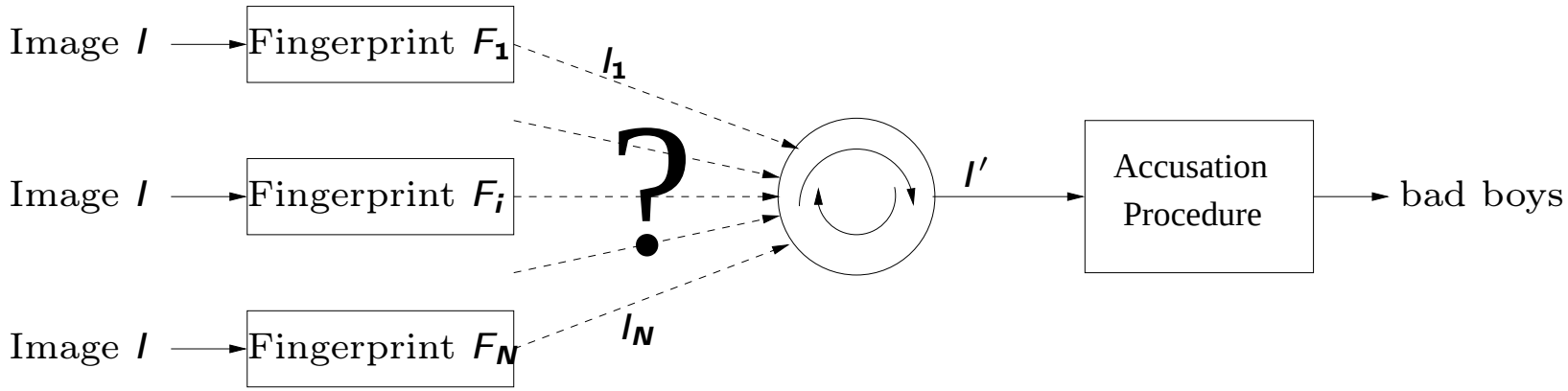
- **Principle** : The watermark must be both invisible and robust.
- **False Detection** : An unwatermarked content detected as watermarked.
- **Constraint** : Copy Protection Working Group $\Rightarrow P_{fd} < 10^{-5}$.

Toy Example : Zero-Bit Watermarking



- $u \in \mathbb{R}^d$ is a fixed and normalized secret vector.
- A content X is deemed watermarked if $S(X) = \frac{\langle X,u \rangle}{\|X\|} > \tau$.
- **Classic Assumption** : An unwatermarked content X has a radially symmetric pdf.
- **False detection** : $P_{fd} = \mathbb{P}(S(X) > \tau | X \text{ unwatermarked})$.

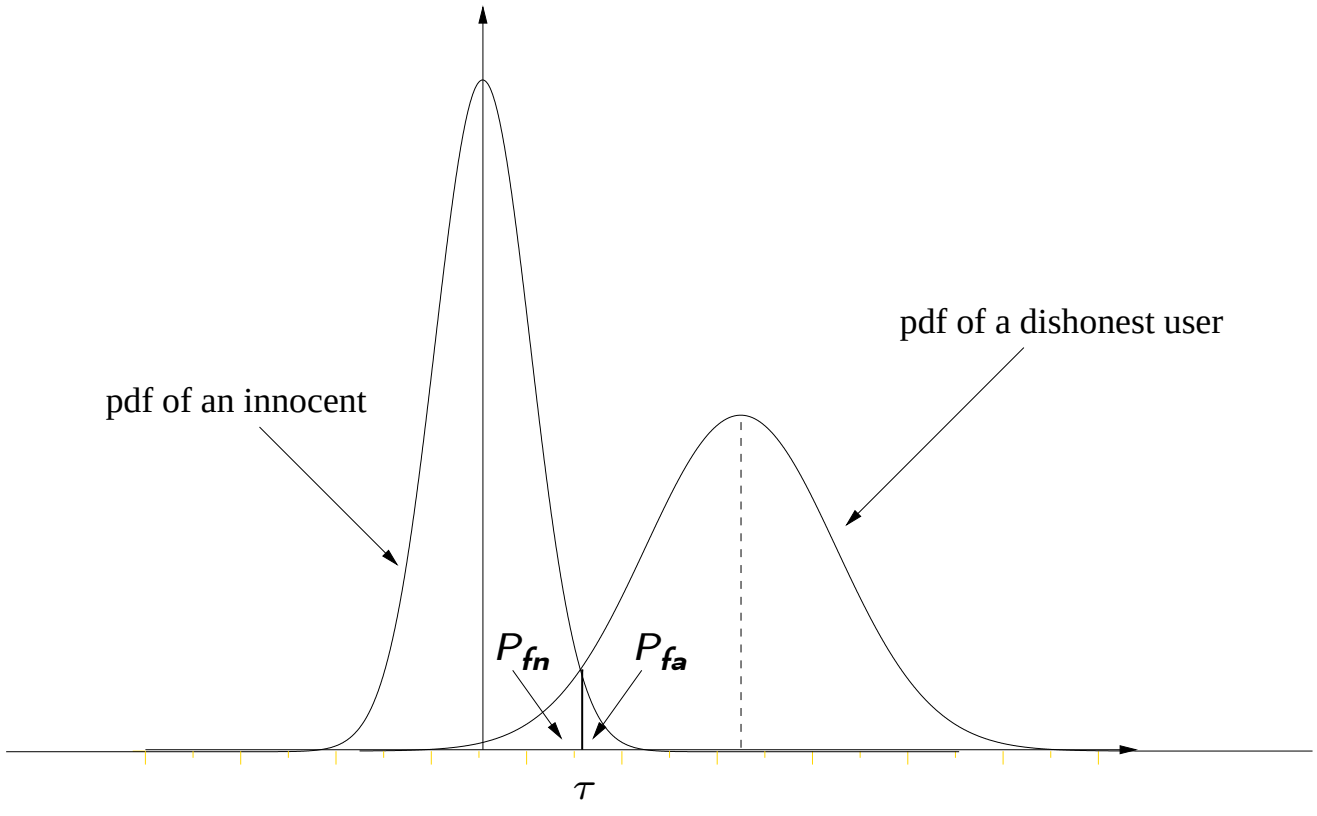
Tricky Example : Probabilistic Fingerprinting



- **Principle** : $F_i \in \{0,1\}^m$ is hidden in the copy of each user.
- **Benefit** : Find a dishonest user via his fingerprint.
- **Question** : What if several dishonest users collude ?
- **False Detections** : Accusing an innocent (false alarm) or accusing none of the colluders (false negative).

$\Rightarrow$ **Answer** : Tardos probabilistic codes.

Tricky Example : Probabilistic Fingerprinting



- **Fingerprint** : $X = [X_1, \ldots, X_m]$, $X_i \sim \mathcal{B}(p_i)$ and $p_i \sim f(p)$.
- **Pirated Copy** : $y = [y_1, \ldots, y_m] \in \{0, 1\}^m$.
- **Accusation procedure** : $S(X) = \sum_{i=1}^{m} y_i g_i(X_i) \gtrless \tau$.

Naive Monte-Carlo

Recall : the aim is to estimate

$$\alpha = \mathbb{P}(\mathcal{R}) = \mathbb{P}(S(X) > \tau).$$

- Simulate $\xi_1, \dots, \xi_N \sim \mu$ and denote

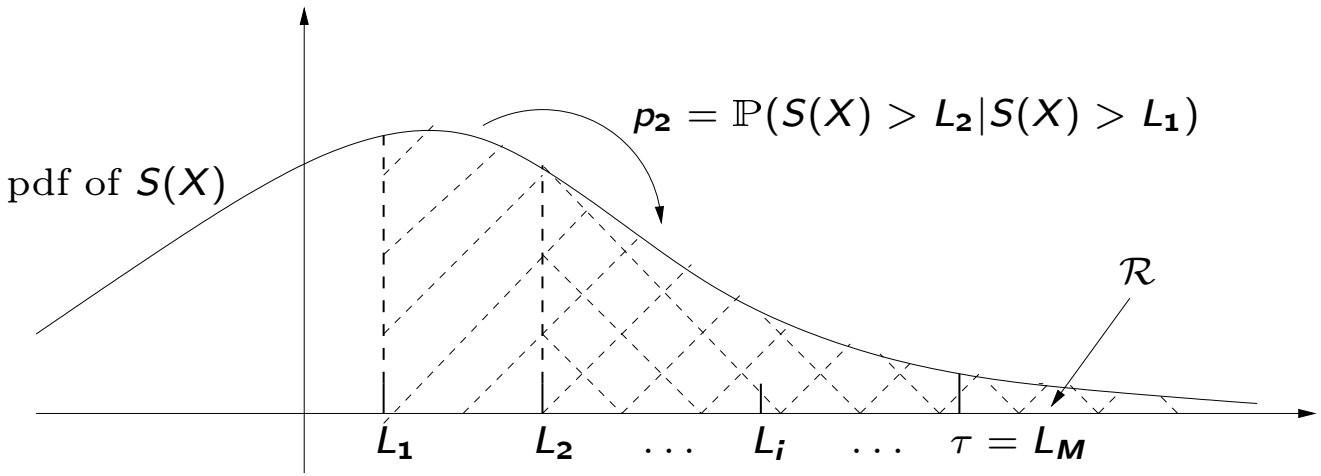
$$N_{\mathcal{R}} = \#\{i \in \{1, \dots, N\} \ : \ S(\xi_i) > \tau\}$$

- Monte-Carlo Estimator : $\hat{\alpha}_N = N_{\mathcal{R}}/N$, but...
  - About α^{-1} simulations are necessary to make $\mathcal{R}$ occur.
  - The relative standard deviation is a disaster :

$$\frac{\sigma(\hat{\alpha}_N)}{\alpha} = \frac{\sqrt{1-\alpha}}{\sqrt{N\alpha}} \approx \frac{1}{\sqrt{N\alpha}}.$$

$\Rightarrow$ **Idea** : Multilevel Monte-Carlo Method.

The Idea



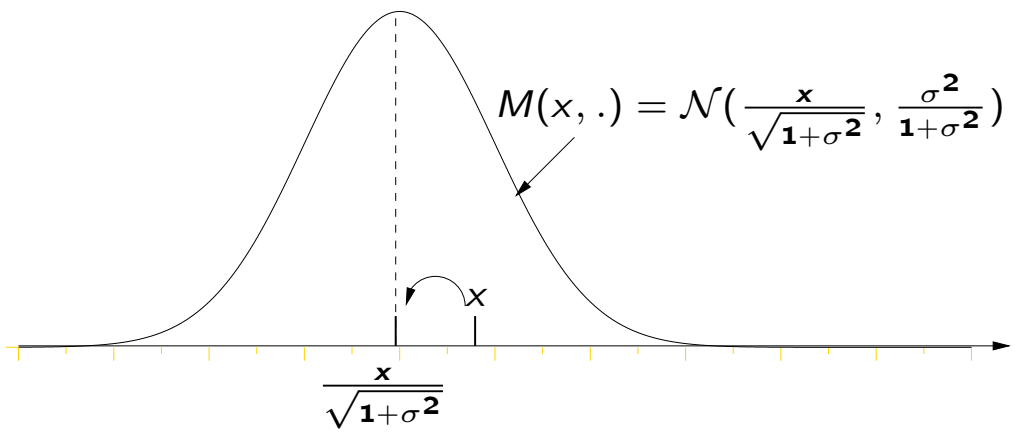
- **Ingredients** : fix M and $L_1 < \cdots < L_M = \tau$ so that each $p_i = \mathbb{P}(S(X) > L_i | S(X) > L_{i-1})$ is not too small.
- **Bayes decomposition** : $\alpha = p_1 p_2 \ldots p_M$.
- **Unreasonable assumption** : suppose you can estimate each p_i independently with classic Monte-Carlo : $p_i \approx \hat{p}_i = N_i/N$.
- **Multilevel Estimator** : $\hat{\alpha}_N = \hat{p}_1 \hat{p}_2 \ldots \hat{p}_M$.

The Shaker

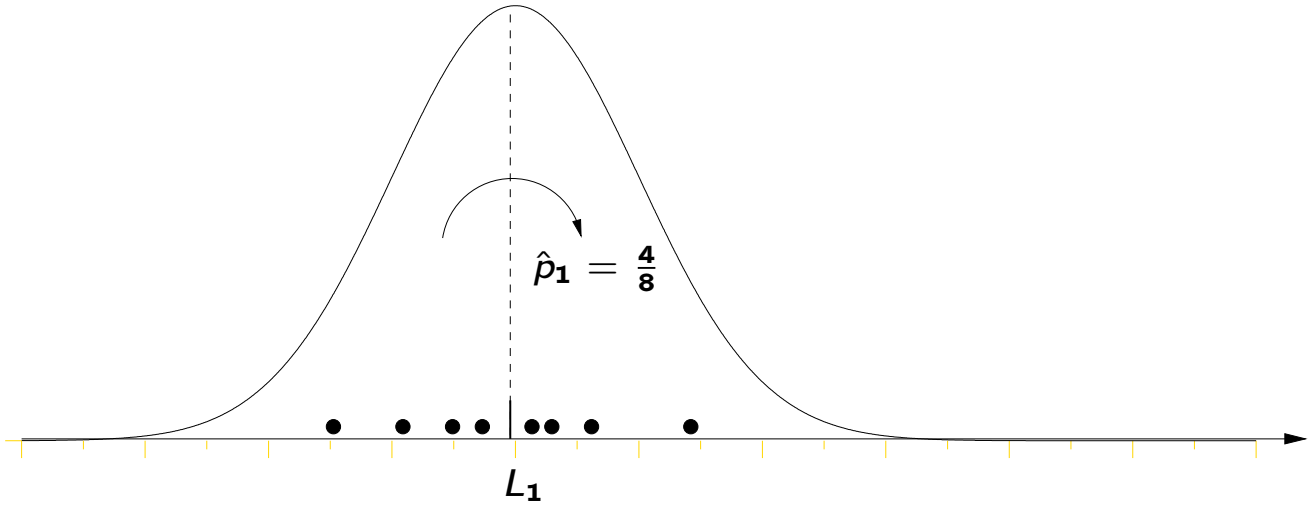
- **Recall** : $X \sim \mu$ on E.
- **Ingredient** : a μ reversible transition kernel $M(x, dx')$ on E :

$$\forall (x, x') \in E^2 \qquad \mu(dx)M(x, dx') = \mu(dx')M(x', dx).$$

- **Consequence** : $\mu M = \mu$.
- **Example** : if $X \sim \mathcal{N}(0,1)$ then $X' = \frac{X+\sigma W}{\sqrt{1+\sigma^2}} \sim \mathcal{N}(0,1)$, i.e. $M(x, dx') \sim \mathcal{N}(\frac{x}{\sqrt{1+\sigma^2}}, \frac{\sigma^2}{1+\sigma^2})(dx')$ is a "good shaker".



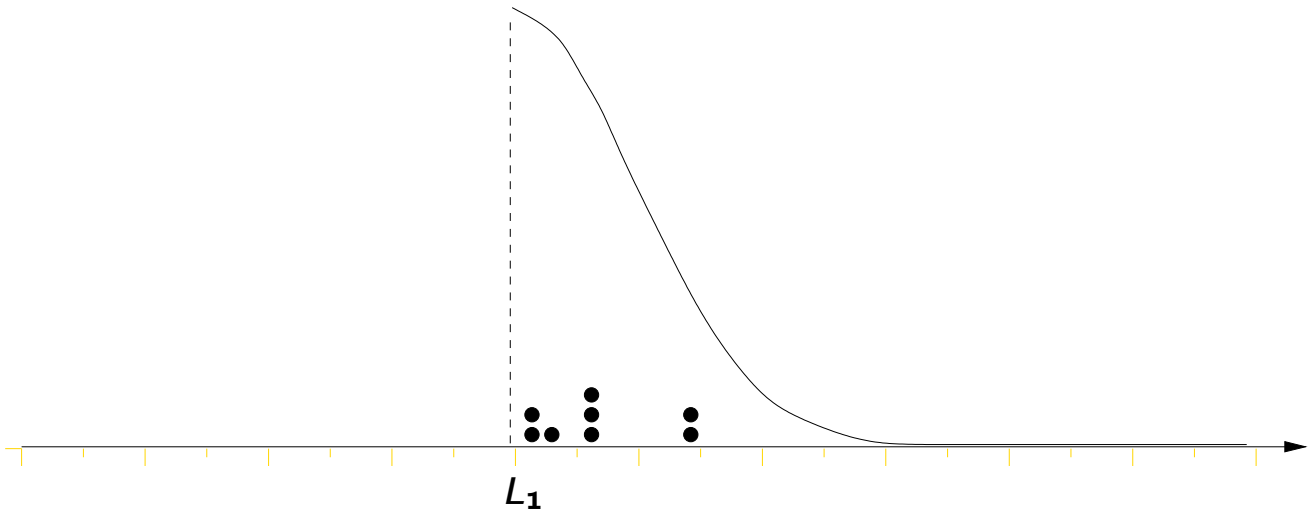
A Selection/Mutation Algorithm



- **Initialization** : Simulate an i.i.d. sample $\xi_0^1, \ldots, \xi_0^N \sim \mu$.
- **Selection** : $\hat{\xi}_0^i = \xi_0^i$ if $S(\xi_0^i) > L_1$, else pick at random among the N_1 selected particles.
- **Mutation** : $\tilde{\xi}_0^i \sim M(\hat{\xi}_0^i, dx')$ and

$$\forall i \in \{1, \ldots, N\} \qquad \xi_1^i = \begin{cases} \tilde{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) > L_1 \\ \hat{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) \leq L_1 \end{cases}$$

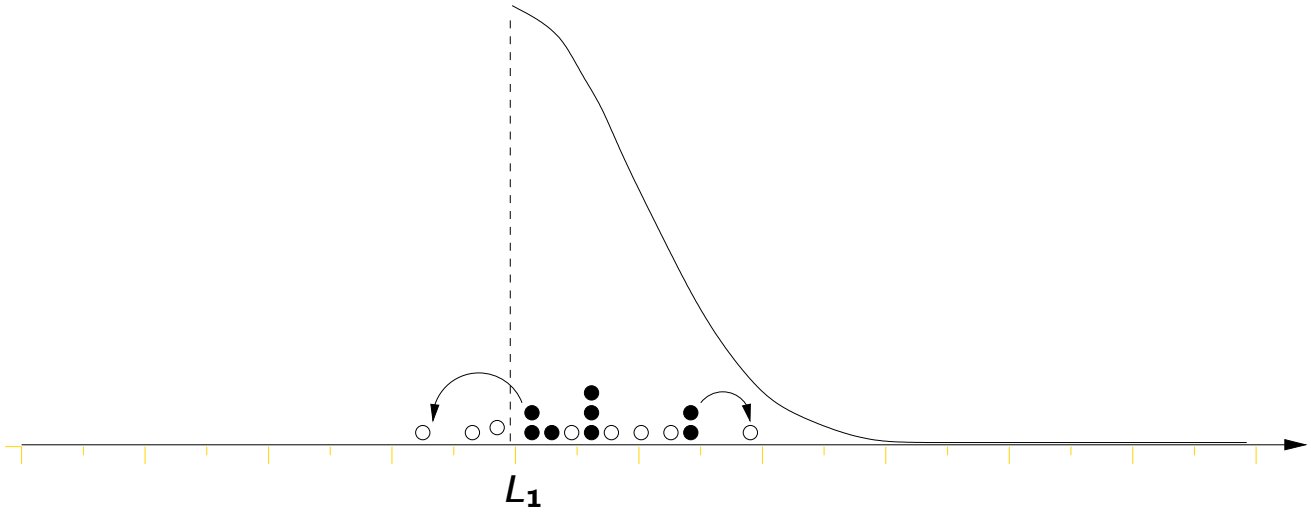
A Selection/Mutation Algorithm



- **Initialization** : Simulate an i.i.d. sample $\xi_0^1, \ldots, \xi_0^N \sim \mu$.
- **Selection** : $\hat{\xi}_0^i = \xi_0^i$ if $S(\xi_0^i) > L_1$, else pick at random among the N_1 selected particles.
- **Mutation** : $\tilde{\xi}_0^i \sim M(\hat{\xi}_0^i, dx')$ and

$$\forall i \in \{1, \ldots, N\} \qquad \xi_1^i = \begin{cases} \tilde{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) > L_1 \\ \hat{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) \leq L_1 \end{cases}$$

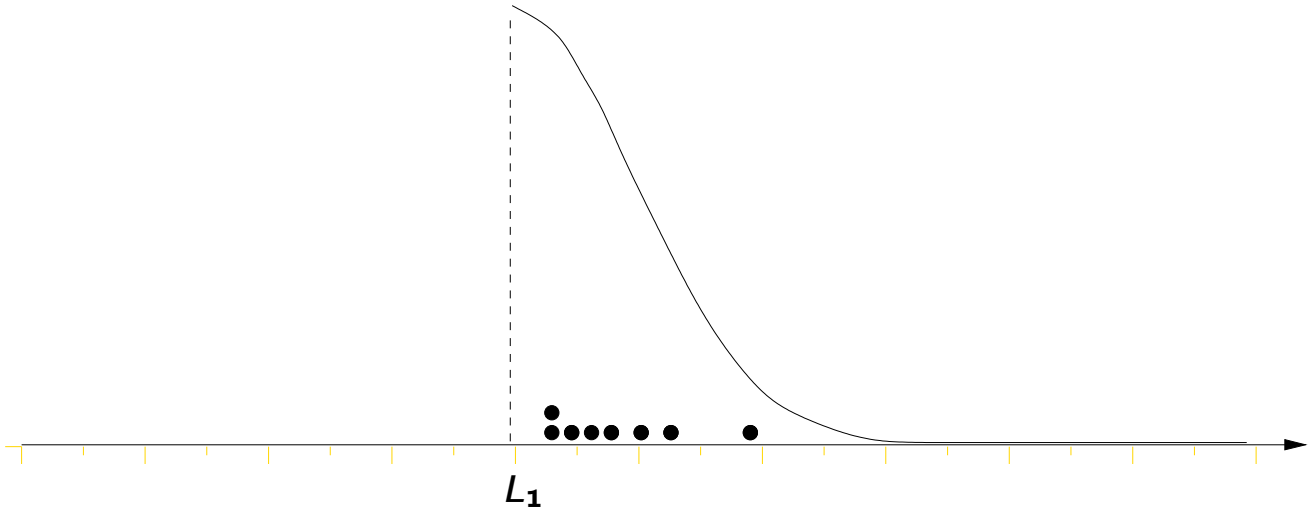
A Selection/Mutation Algorithm



- **Initialization** : Simulate an i.i.d. sample $\xi_0^1, \ldots, \xi_0^N \sim \mu$.
- **Selection** : $\hat{\xi}_0^i = \xi_0^i$ if $S(\xi_0^i) > L_1$, else pick at random among the N_1 selected particles.
- **Mutation** : $\tilde{\xi}_0^i \sim M(\hat{\xi}_0^i, dx')$ and

$$\forall i \in \{1, \ldots, N\} \qquad \xi_1^i = \begin{cases} \tilde{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) > L_1 \\ \hat{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) \leq L_1 \end{cases}$$

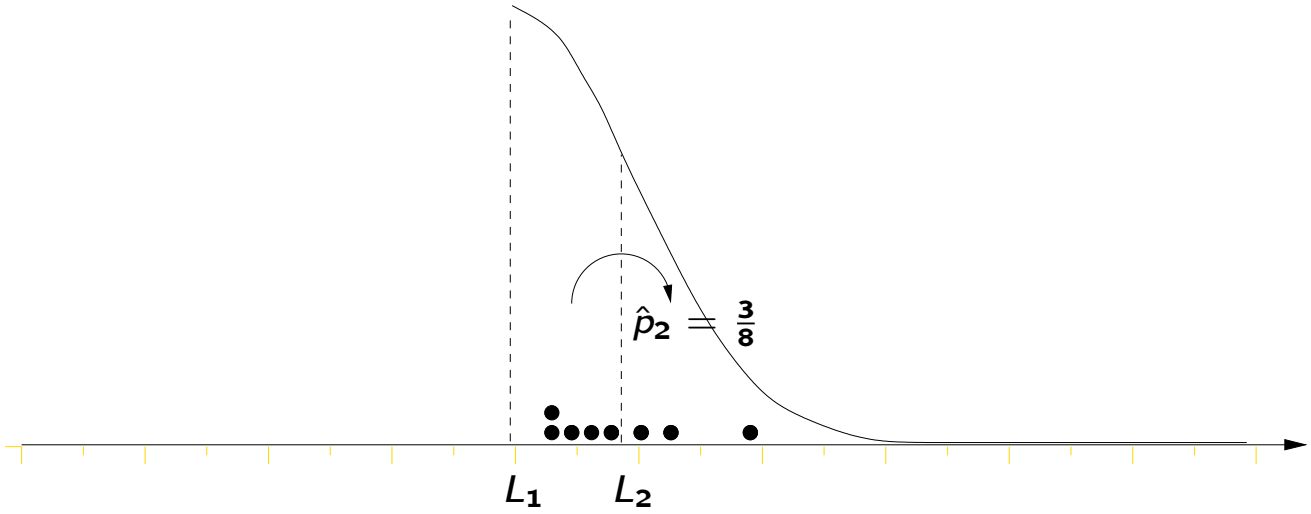
A Selection/Mutation Algorithm



- **Initialization** : Simulate an i.i.d. sample $\xi_0^1, \ldots, \xi_0^N \sim \mu$.
- **Selection** : $\hat{\xi}_0^i = \xi_0^i$ if $S(\xi_0^i) > L_1$, else pick at random among the N_1 selected particles.
- **Mutation** : $\tilde{\xi}_0^i \sim M(\hat{\xi}_0^i, dx')$ and

$$\forall i \in \{1, \ldots, N\} \qquad \xi_1^i = \begin{cases} \tilde{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) > L_1 \\ \hat{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) \leq L_1 \end{cases}$$

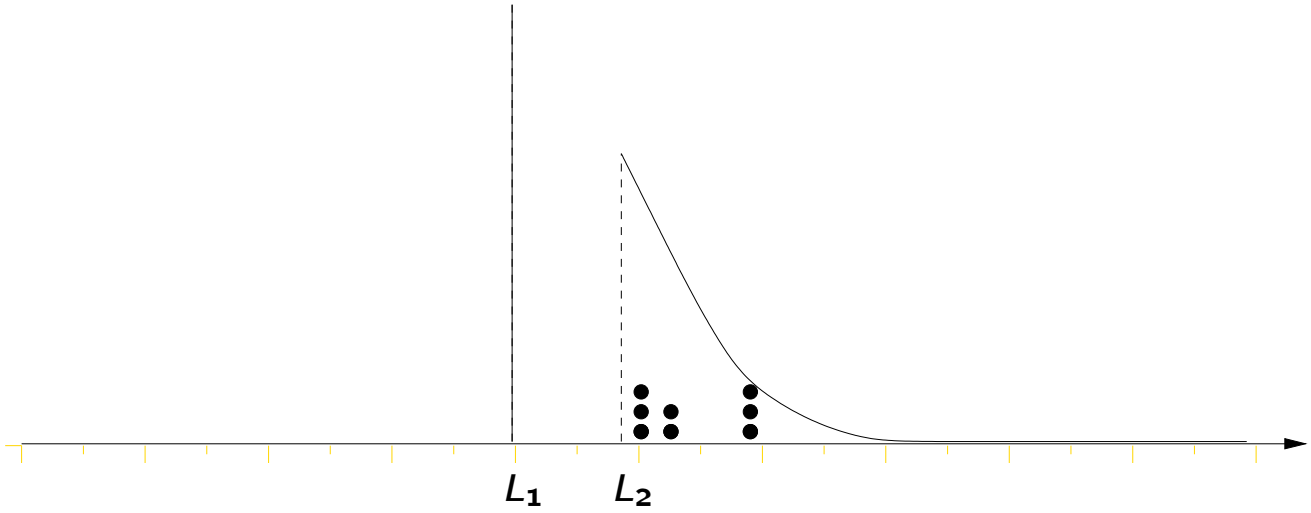
A Selection/Mutation Algorithm



- **Initialization** : Simulate an i.i.d. sample $\xi_0^1, \ldots, \xi_0^N \sim \mu$.
- **Selection** : $\hat{\xi}_0^i = \xi_0^i$ if $S(\xi_0^i) > L_1$, else pick at random among the N_1 selected particles.
- **Mutation** : $\tilde{\xi}_0^i \sim M(\hat{\xi}_0^i, dx')$ and

$$\forall i \in \{1, \ldots, N\} \qquad \xi_1^i = \begin{cases} \tilde{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) > L_1 \\ \hat{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) \leq L_1 \end{cases}$$

A Selection/Mutation Algorithm



- **Initialization** : Simulate an i.i.d. sample $\xi_0^1, \ldots, \xi_0^N \sim \mu$.
- **Selection** : $\hat{\xi}_0^i = \xi_0^i$ if $S(\xi_0^i) > L_1$, else pick at random among the N_1 selected particles.
- **Mutation** : $\tilde{\xi}_0^i \sim M(\hat{\xi}_0^i, dx')$ and

$$\forall i \in \{1, \ldots, N\} \qquad \xi_1^i = \begin{cases} \tilde{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) > L_1 \\ \hat{\xi}_1^i & \text{if } S(\tilde{\xi}_1^i) \leq L_1 \end{cases}$$

Convergence of the Algorithm

- $A_n = \{x \in E \ : \ S(x) > L_n\}$ and $\mu_n = \mathcal{L}(X|S(X) > L_n)$.
- Non-homogeneous transition kernel :

$$M_n(x, dx') = M(x, dx')\mathbb{1}_{A_n}(x') + M(x, A_n^c)\delta_x(dx').$$

It is easy to check that μ_n is invariant by M_n.

Theorem (Feynman-Kac Formula)

Define a Markov chain (X_n) having the transition kernels (M_n) and initial law μ, then for any test function φ and any n :

$$\mu_n(\varphi) = \frac{\mathbb{E}[\varphi(X_n)\prod_{k=1}^n \mathbb{1}_{A_k}(X_k)]}{\mathbb{E}[\prod_{k=1}^n \mathbb{1}_{A_k}(X_k)]}.$$

Remark : thus, after M steps : $\mu_M = \mathcal{L}(X|S(X) > \tau)$.

Constrained Optimization

- **Multilevel Estimator** : $\hat{\alpha}_N = \hat{p}_1 \hat{p}_2 \dots \hat{p}_M$.
- **Fluctuations** : If the $\hat{p}_i$'s are independent, then

$$\sqrt{N} \cdot \frac{\hat{\alpha}_N - \alpha}{\alpha} \xrightarrow[N \to \infty]{\mathcal{L}} \mathcal{N}(0, \sum_{i=1}^{M} \frac{1 - p_i}{p_i}).$$

- **Constrained Minimization** :

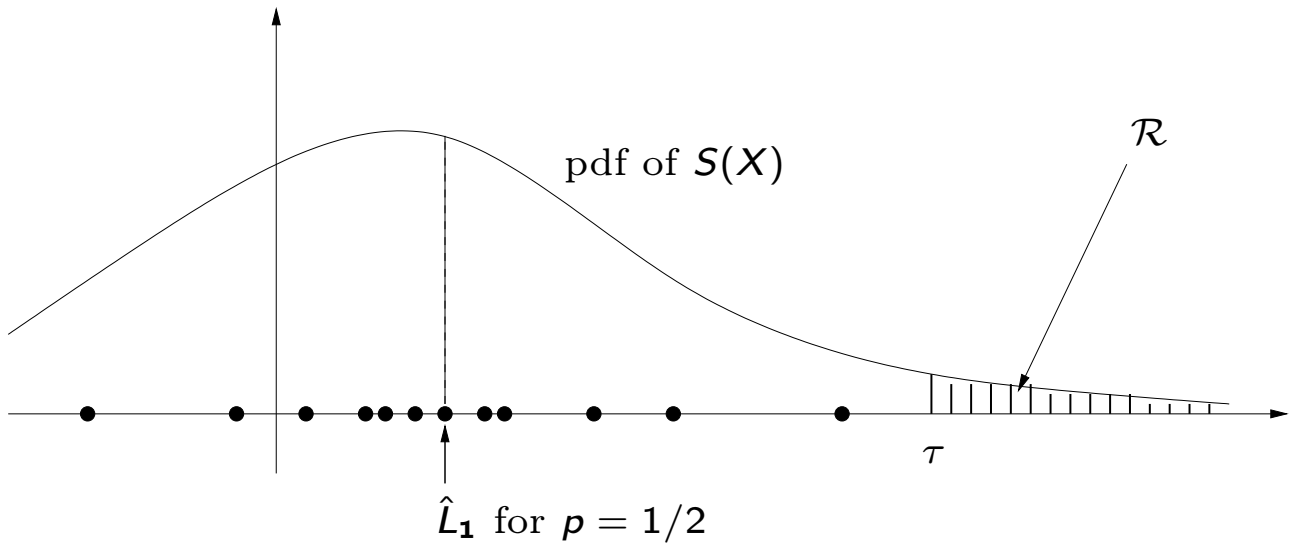
$$\arg \min_{p_1, \dots, p_M} \sum_{i=1}^{M} \frac{1 - p_i}{p_i} \qquad \text{s.t.} \qquad \prod_{i=1}^{M} p_i = \alpha.$$

- **Optimum** : $p_1 = \dots = p_M = \alpha^{1/M}$.

$\Rightarrow$ **Solution** : Adaptive levels.

Adaptive Levels

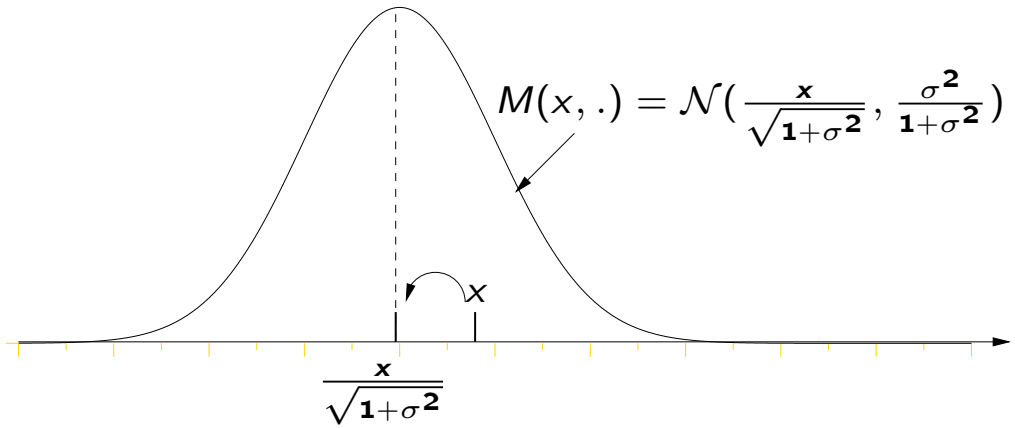
Parameter : fix a proportion p of surviving particles from one step to another rather than M and the levels $L_1, \ldots, L_M$.



$\Rightarrow$ **Adaptive multilevel estimator** :

$$\alpha = r \times p^M \approx \hat{\alpha}_N = \hat{r} \times p^{\hat{M}}.$$

The Impact of the Kernel



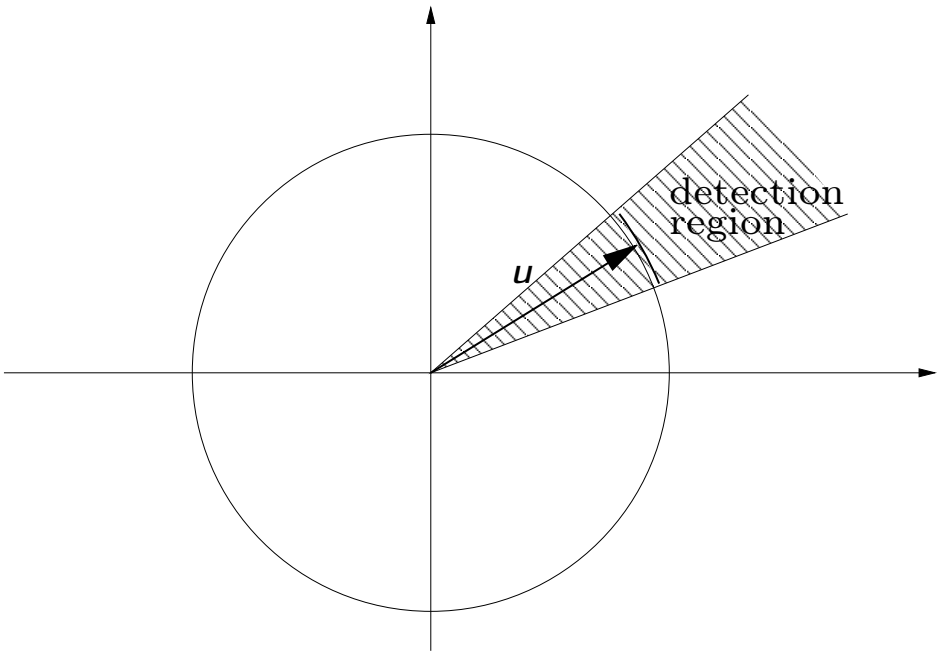
- **The model :** $X' = \frac{X+\sigma W}{\sqrt{1+\sigma^2}} \sim \mathcal{N}(0,1)$.
- **Expected square distance :** $\mathbb{E}[(X'-X)^2] = 2\left(1 - \frac{1}{\sqrt{1+\sigma^2}}\right)$.

Trade-off between two drawbacks :

- σ too large : most proposed mutations are refused.
- σ too small : particles almost don't move.

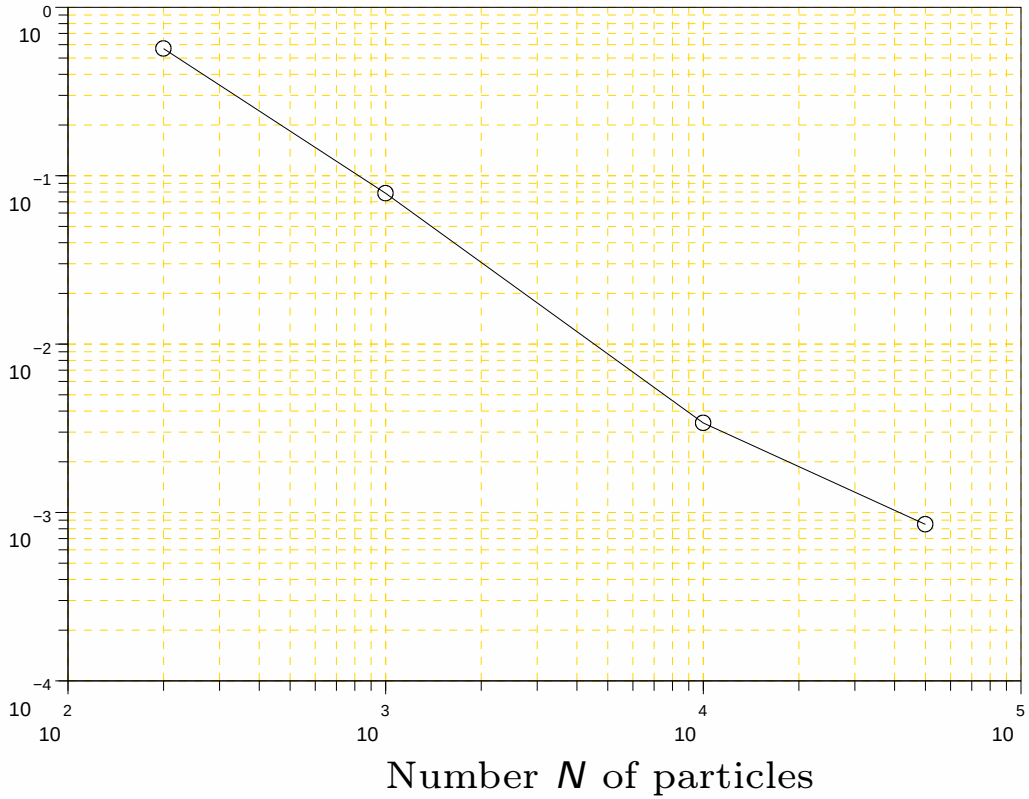
- **Solution :** Adaptive σ based on the filtering rate.

Simulations : The Model



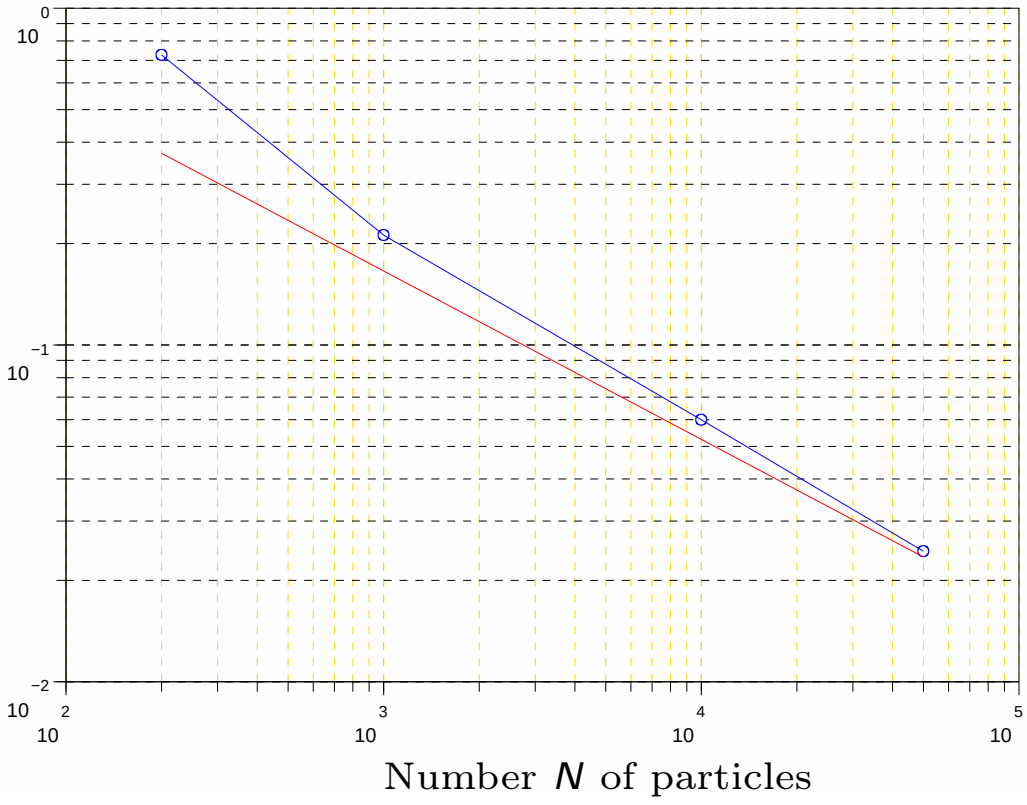
- **The model :** $X \sim \mathcal{N}(0, I_{20})$.
- **Rare event :** $\alpha = \mathbb{P}(\frac{\langle X,u \rangle}{\|X\|} > 0.95)$.
- **Numerical computation :** $\alpha = 4.704 \cdot 10^{-11}$.
- **Parameter :** $p = 3/4 \rightsquigarrow \alpha = r \times p^M = .83 \times (3/4)^{82}$.

The Bias



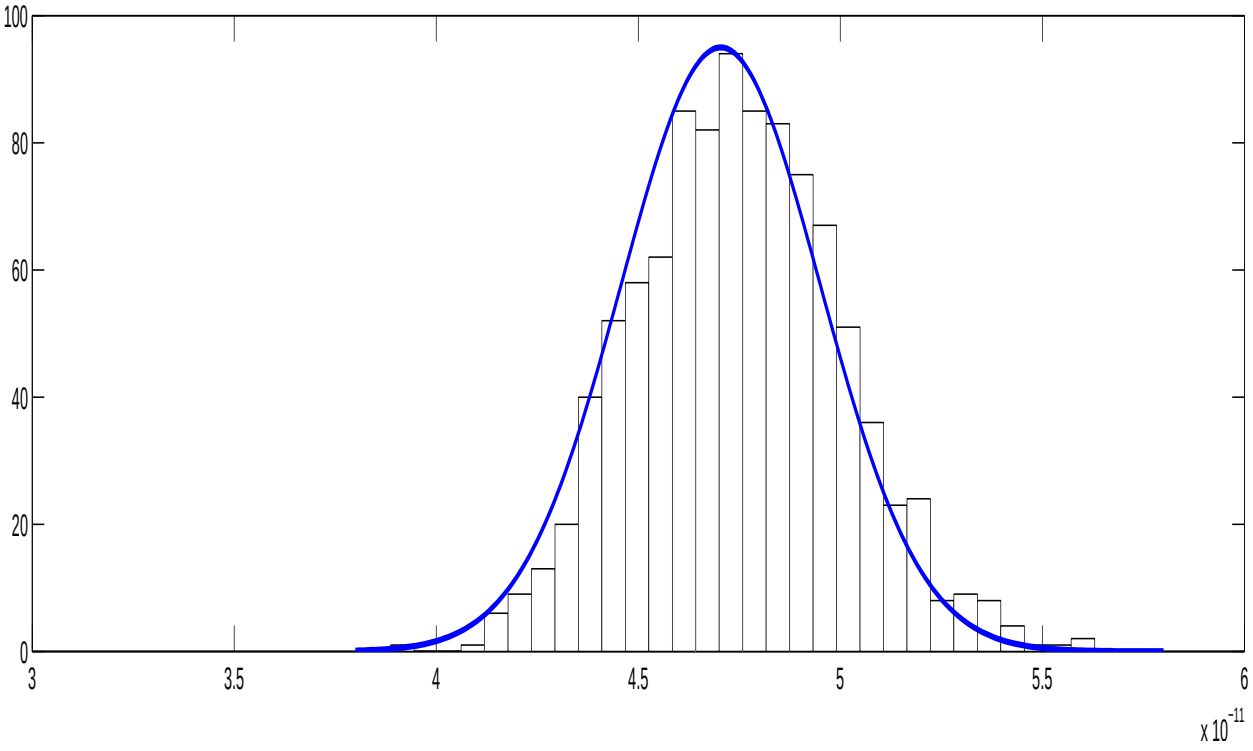
$$\frac{\mathbb{E}[\hat{\alpha}_N] - \alpha}{\alpha} \approx \frac{c^2}{N}.$$

The Standard Deviation



$$\hat{\sigma}_N \xrightarrow[N\to\infty]{} \sigma_{\text{th}} = \sqrt{M \cdot \frac{1-p}{p} + \frac{1-r}{r}}.$$

Asymptotic Development



Summary : Putting all things together, we have obtained

$$\hat{\alpha}_N = \alpha + \frac{\sigma_{\text{th}}}{\sqrt{N}} \mathcal{N}(0,1) + \frac{c^2}{N} + \dots$$