Principe des méthodes de Monte-Carlo

1. Introduction

Les méthodes de Monte-Carlo utilisent des nombres pseudo aléatoires (générés par un algorithme) pour simuler des phénomènes comportant une ou plusieurs variables aléatoires. Le nom provient du célèbre casino de Monte-Carlo.

On considère une simulation de Monte-Carlo élémentaire, visant à évaluer l'espérance et la variance d'une variable aléatoire en générant un grand nombre d'échantillons qui suivent la même loi de probabilité que la variable aléatoire.

2. Variable aléatoire discrète

2.a. Espérance et variance

Soit une variable aléatoire X pouvant prendre les M valeurs x_k avec $k = 0, ..M-1$. La probabilité d'obtenir la valeur x_k est notée p_k. La donnée de ces probabilités constitue la loi de la variable aléatoire, appelée aussi distribution des probabilités. La somme des probabilités doit être égale à 1 :

$$\sum_{k=0}^{M-1} p_k = 1 \tag{1}$$

L'espérance de la variable aléatoire est :

$$E(X) = \sum_{k=0}^{M-1} p_k x_k \tag{2}$$

Remarque : en physique statistique, l'espérance d'une grandeur physique aléatoire est appelée *moyenne* ou *moyenne statistique* de cette grandeur.

La variance est l'espérance du carré de l'écart entre la variable et son espérance :

$$\begin{aligned} var(X) =& E\left[(X - E(X))^2\right] && (3)\\ =& E\left[X^2 - 2XE(X) + E(X)^2\right] && (4)\\ =& E(X^2) - E(X)^2 && (5)\\ =& \sum_{k=0}^{M-1} p_k x_k^2 - \left(\sum_{k=0}^{M-1} p_k x_k\right)^2 && (6) \end{aligned}$$

L'écart type est la racine carrée de la variance :

$$\Delta X = \sqrt{var(X)} \tag{7}$$

2.b. Simulation de Monte-Carlo

Principe

On utilise un générateur de nombres pseudo aléatoires pour générer des échantillons x_i de la variable aléatoire X. Les valeurs de ces échantillons sont dans l'ensemble des valeurs x_k.

Soit N le nombre d'échantillons générés. L'espérance est évaluée en calculant la *moyenne empirique* définie par :

$$y_N = moy(X, N) = \frac{1}{N}\sum_{i=0}^{N-1} x_i \tag{8}$$

L'espérance de la moyenne empirique est égale à l'espérance de X.

Cette moyenne est généralement calculée pour des valeurs de N croissantes, ce qui nécessite le stockage de la somme des x_i.

La variance empirique est définie par :

$$v_N = var(X, N) = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - y_N)^2 \tag{9}$$

L'espérance de la variance empirique est égale à la variance de X.

En pratique, on préfère utiliser la forme développée suivante :

$$v_N = \frac{1}{N-1}\sum_{i=0}^{N-1} x_i^2 - \frac{N}{N-1}y_N^2 \tag{10}$$

Dans une simulation de Monte-Carlo, le nombre N de tirage est très grand (au moins 1000). On peut donc confondre $N-1$ et N, ce qui donne l'expression approchée suivante de la variance empirique :

$$v_N \approx \frac{1}{N}\sum_{i=0}^{N-1} x_i^2 - y_N^2 \tag{11}$$

L'estimation de la variance nécessite donc de stocker la somme des carrés des échantillons.

La moyenne y_M est elle-même une variable aléatoire : pour N fixé, la répétition de N tirages de la variable X donne des valeurs aléatoires de la moyenne. La variance de y_N est d'autant plus faible que N est grand. Pour calculer cette variance, on utilise les deux propriétés suivantes valables pour deux variables aléatoires indépendantes :

$$var(x_1 + x_2) = var(x_1) + var(x_2) \tag{12}$$
$$var(ax) = a^2 var(x) \tag{13}$$

On obtient ainsi la variance de la moyenne :

$$var(y_N) = \frac{var(X)}{N} \tag{14}$$

Il s'en suit que l'écart-type de la moyenne varie comme l'inverse de la racine carrée :

$$\Delta(y_N) = \frac{\sqrt{var(X)}}{\sqrt{N}} \tag{15}$$

Cet écart-type est évalué avec la variance empirique v_N.

Le *théorème de la limite centrale* établit que la moyenne empirique suit (si N est assez grand) une distribution continue gaussienne (voir plus loin). Un intervalle de confiance à 95 pour cent est donné par :

$$\left[y_N - \frac{1,96\sqrt{v_N}}{\sqrt{N}}, y_N + \frac{1,96\sqrt{v_N}}{\sqrt{N}} \right] \tag{16}$$

La probabilité pour que l'espérance cherchée se trouve dans cet intervalle est de $0,95$.

Exemple

On considère des tirages d'un dé. Les valeurs possibles sont 1,2,3,4,5,6 avec chacun la probabilité 1/6. La fonction `random.randint(1,6)` permet d'obtenir les échantillons. Dans ce cas élémentaire, on connaît la valeur exacte de l'espérance :

$$E(X) = \frac{1}{6}(1+2+3+4+5+6) = \frac{7}{2} = 3,5 \tag{17}$$

et de la variance :

$$var(X) = \frac{1}{6}\left((1-7/2)^2 + (2-7/2)^2 + \cdots\right) \simeq 2,9167 \tag{18}$$

La fonction suivante effectue N tirages, calcule la moyenne et la variance empiriques, et le demi-intervalle de confiance à 95 pour cent :

```
import random
import math

def tirages(N):
    somme = 0.0
    somme2 = 0.0
    for i in range(N):
        x = random.randint(1,6)
        somme += x
        somme2 += x*x
    moyenne = somme*1.0/N
    variance = somme2*1.0/N-moyenne*moyenne
    ecart = 1.96*math.sqrt(variance*1.0/N)
    return (moyenne,variance,ecart)
```

Voici un exemple avec 1000 tirages :

```
t = tirages(1000)
```

```
print(t)
--> (3.526, 2.979324000000002, 0.10698304107848126)
```

Voici une deuxième série de tirages :

```
t = tirages(1000)
```

```
print(t)
--> (3.418, 2.945276, 0.10636997829086928)
```

Pour réduire l'écart-type de la moyenne, on doit augmenter le nombre de tirages :

```
t = tirages(10000)
```

```
print(t)
--> (3.5017, 2.8911971100000002, 0.03332690027256661)
```

Cette dernière simulation donne donc, avec un intervalle de confiance à 95 pour cent :

$$E(X) = 3.50 \pm 0.03 \tag{19}$$

3. Variable aléatoire continue

3.a. Densité de probabilité

Soit X une variable aléatoire continue (appelée aussi variable à densité), qui peut prendre toute valeur réelle dans l'intervalle $[a, b]$. On définit une fonction $p(x)$ appelée *densité de probabilité* de la variable aléatoire, telle que la probabilité d'obtenir une valeur dans l'intervalle $[x_1, x_2]$ soit :

$$P(x_1 \leq x \leq x_2) = \int_{x_1}^{x_2} p(x)\, dx \tag{20}$$

La densité de probabilité doit vérifier la condition suivante, appelée *condition de normalisation* :

$$\int_a^b p(x)\, dx = 1 \tag{21}$$

On introduit aussi la fonction de répartition $F(x)$ qui donne la probabilité d'obtenir une valeur inférieure ou égale à x :

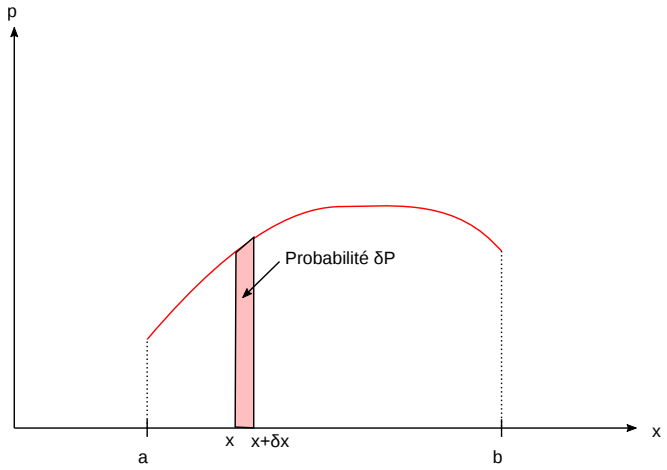
$$F(x) = \int_a^x p(x')\, dx' \tag{22}$$

En dérivant la fonction de répartition, on obtient :

$$p(x) = \frac{dF}{dx} \tag{23}$$

La probabilité d'obtenir une valeur x donnée dans l'intervalle $[a, b]$ est nulle. En pratique, on s'intéresse à la probabilité d'obtenir une valeur entre x et $x + \delta x$, égale approximativement à :

$$\delta P \simeq p(x)\delta x \tag{24}$$



L'espérance d'une variable aléatoire $f(X)$ est définie par :

$$E(f(X)) = \int_a^b p(x) f(x)\, dx \tag{25}$$

La variance se définit comme pour une variable discrète.

Une *densité de probabilité uniforme* est constante sur l'intervalle $[a, b]$. Si les bornes de cet intervalle ne sont pas à l'infini, la densité de probabilité est obtenue avec la condition de normalisation :

$$p(x) = \frac{1}{b-a} \tag{26}$$

L'espérance de x est alors :

$$E(x) = \int_a^b \frac{x\, dx}{b-a} = \frac{a+b}{2} \tag{27}$$

La *distribution gaussienne* (ou normale) est définie sur l'intervalle $-\infty, \infty$ par :

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{28}$$

Elle vérifie la condition de normalisation et on a :

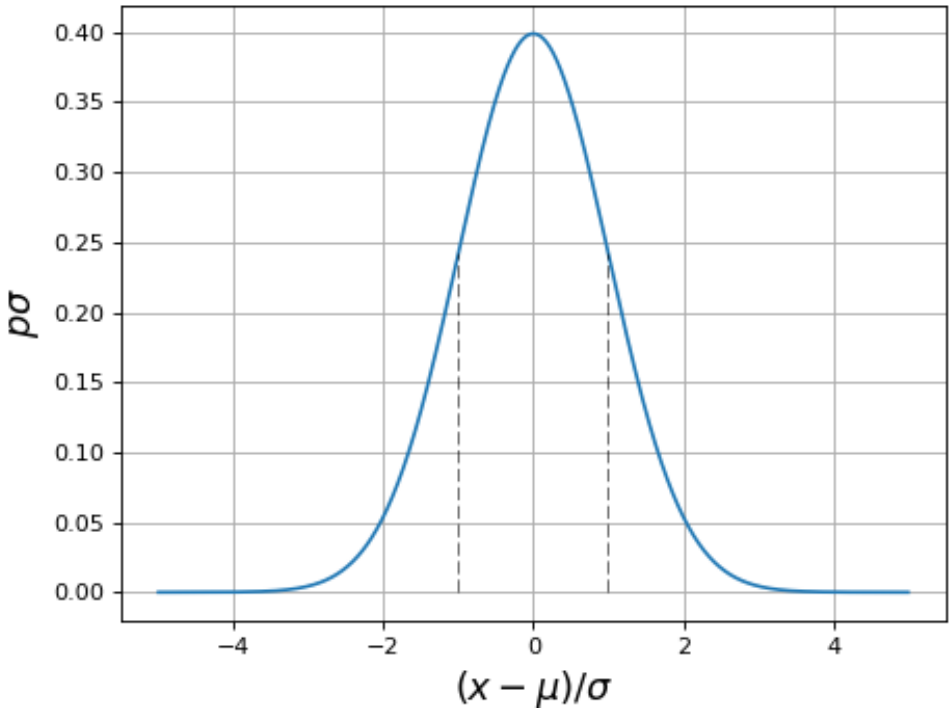
$$E(x) = \mu \tag{29}$$

$$var(x) = \sigma^2 \tag{30}$$

Voici la représentation graphique de cette densité de probabilité en fonction de $(x-\mu)/\sigma$.

```
from matplotlib.pyplot import *
import numpy
x=numpy.linspace(-5,5,1000)
p=1/(numpy.sqrt(2*numpy.pi))*numpy.exp(-x**2/2)
figure()
plot(x,p)
p1=1/(numpy.sqrt(2*numpy.pi))*numpy.exp(-1/2)
```

```
plot([-1,-1],[0,p1],"k--",linewidth=0.5)
plot([1,1],[0,p1],"k--",linewidth=0.5)
xlabel(r"$(x-\mu)/\sigma$",fontsize=16)
ylabel(r"$p\sigma$",fontsize=16)
grid()
```



La probabilité d'obtenir une valeur dans un intervalle de largeur 2σ centré sur l'espérance est :

$$P(\mu - \sigma \leq x \leq \mu + \sigma) = \int_{\mu-\sigma}^{\mu+\sigma} p(x)dx \approx 0,68 \tag{31}$$

La loi normale est courrament utilisée pour représenter les variations aléatoires des mesures d'une grandeur physique (incertitude expérimentale). L'incertitude type est par définition l'écart type $\sqrt{\sigma}$. Soit m la valeur mesurée, obtenue soit par une mesure unique, soit par une moyenne de mesures répétées. m est une estimation de l'espérance. Il faut aussi obtenir une estimation de l'écart type, appelée incertitude de la mesure, et notée Δm. Le résultat de la mesure est généralement présenté en disant que la valeur de la grandeur mesurée se trouve dans l'intervalle $[m - \Delta m, m + \Delta m]$ avec une probabilité de $0{,}68$. On peut aussi écrire que la valeur de la grandeur mesurée se trouve dans l'intervalle $[m - 1,96\Delta m, m + 1,96\Delta m]$ avec une probabilité de $0{,}95$.

3.b. Simulation de Monte-Carlo

Principe

Considérons la simulation d'une variable aléatoire continue définie sur l'intervalle $[a, b]$. Les nombres réels sont représentés par des nombres à virgule flottante, qui ont bien sûr une précision limitée. Si l'on note δx cette précision, l'intervalle $[a, b]$ est divisé en M sous-intervalles

égaux de largeur δx. Le tirage d'un nombre à virgule flottante dans cet intervalle revient donc à tirer un entier compris entre 0 et M. La probabilité d'obtenir un nombre x est égale à $p(x)\delta x$.

La fonction `random.uniform(a,b)` délivre un flottant avec une densité de probabilité uniforme sur l'intervalle $[a, b]$. On peut aussi utiliser la fonction `random.random()` qui fait la même chose sur l'intervalle $[0, 1[$ (la valeur 1 est exclue).

La fonction `random.gauss(mu,sigma)` délivre un flottant avec la distribution de Gauss.

L'estimation de l'espérance et de la variance se fait comme déjà expliqué pour une variable aléatoire discrète.

Exemple

On considère comme exemple le tirage de nombres aléatoires avec la loi de Gauss. La fonction suivante fait N tirages. Elle calcule la moyenne empirique, la variance empirique, et l'intervalle de confiance à 95 pour cent pour l'espérance.

La fonction génère aussi un histogramme des échantillons. Pour générer un histogramme, il faut choisir un intervalle $[a, b]$ et des classes de valeur dans cet intervalle. Supposons que ces classes soient les n_b sous intervalles de largeur $h = (b-a)/n_b$. Notons H_j le nombre de tirages donnant une valeur dans l'intervalle $[a+jh, a+(j+1)h]$ avec j variant de 0 à n_b-1. Lorsque les valeurs de la variable aléatoire ne sont pas dans un intervalle borné, il faut choisir des valeurs de a et b. Une première approche consiste à générer tous les échantillons en les stockant dans un tableau puis à choisir le minimum de le maximum des valeurs des échantillons pour a et b. Il est préférable de fixer la largeur des classes (h) et donc de calculer n_b en conséquence. Cette approche a deux inconvénients : elle nécessite le stockage des échantillons ; elle ne permet pas de générer un histogramme qui se met à jour au fûr et à mesure de la génération des échantillons. Nous préférons calculer l'histogramme en fixant *a priori* les valeurs de a et b. L'inconvénient de cette approche est le risque d'avoir des valeurs d'échantillons en dehors de cet intervalle. L'intervalle doit donc être choisi assez grand pour que ce risque soit négligeable.

Le nombre de tirages H_j dans une classe est finalement divisé par le nombre de tirages total (comptés dans l'histogramme) afin d'obtenir une évaluation de la probabilité d'obtenir un tirage dans cette classe.

```
import numpy

def tirages_gauss(N,mu,sigma,a,b,nb):
    somme = 0.0
    somme2 = 0.0
    h = (b-a)/nb
    H = numpy.zeros(nb)
    n = 0
    for i in range(N):
        x = random.gauss(mu,sigma)
        somme += x
        somme2 += x*x
        j = int((x-a)/h)
        if j>=0 and j<nb:
            H[j] += 1
            n += 1
    moyenne = somme*1.0/N
    variance = somme2*1.0/N-moyenne*moyenne
    ecart = 1.96*math.sqrt(variance*1.0/N)
    H=H/n
```

```
    x=numpy.linspace(a,b,nb)
    return (moyenne,variance,ecart,x,H)
```

Voici un exemple :

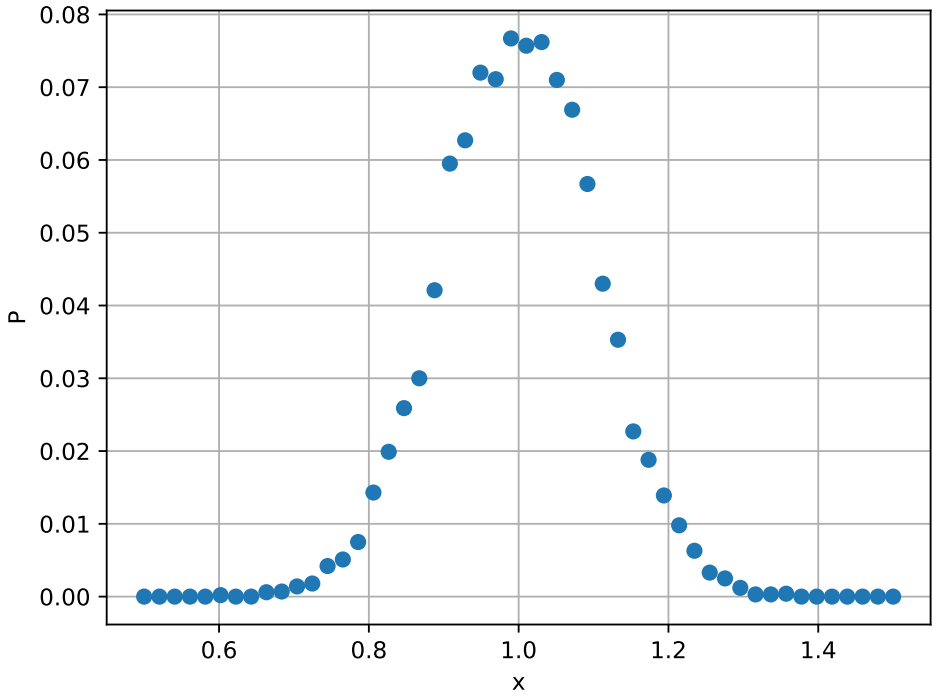
```
mu=1.0
sigma=0.1
a=0.5
b=1.5
nb=50
(m,v,e,x,H) = tirages_gauss(10000,mu,sigma,a,b,nb)


print((m,v,e))
--> (1.0006659135758849, 0.010047662063773277, 0.001964665329876603)
```

On obtient ainsi l'estimation de l'espérance suivante :

$$E(X) = 1.001 \pm 0.002 \tag{32}$$

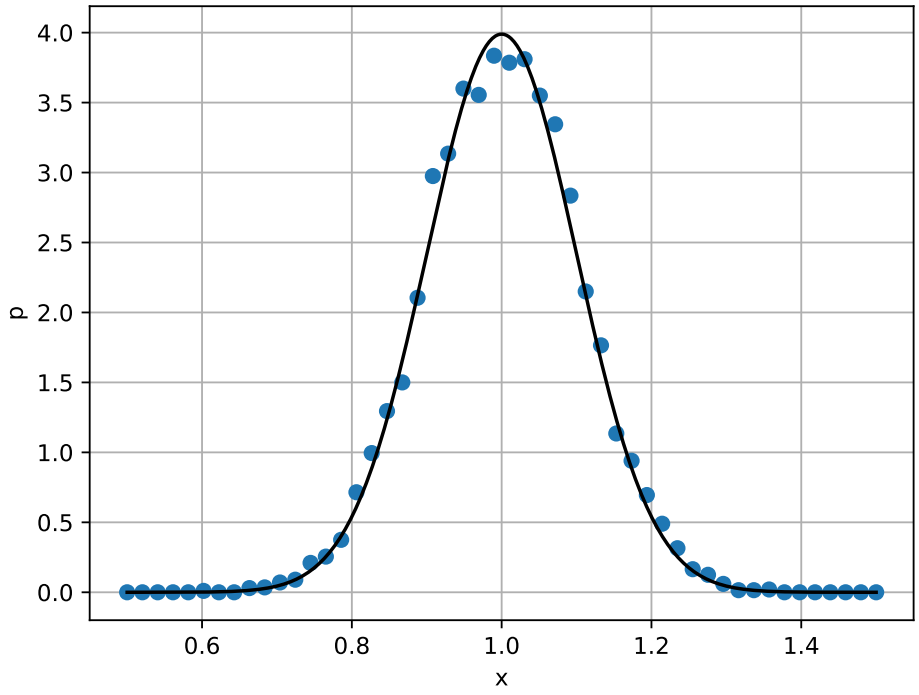
Voici une représentation graphique de l'histogramme :

```
from matplotlib.pyplot import *
figure()
plot(x,H,"o")
grid()
xlabel("x")
ylabel("P")
```



Un histogramme d'une variable aléatoire continue permet d'évaluer la densité. Pour cela, il faut diviser chaque probabilité par la largeur du sous-intervalle correspondant. Dans le cas présent, on peut faire une comparaison avec la densité de probabilité :

```
h=(b-a)/nb
H=H/h
figure()
plot(x,H,"o")
x=numpy.linspace(a,b,1000)
p=1/(numpy.sqrt(2*numpy.pi)*sigma)*numpy.exp(-(x-mu)**2/(2*sigma**2))
plot(x,p,"k-")
grid()
xlabel("x")
ylabel("p")
```



3.c. Échantillonnage d'une densité non uniforme

Une distribution continue de densité non uniforme peut être échantillonnée en inversant la fonction de répartition, définie par :

$$F(x) = \int_a^x p(x')\,dx' \tag{33}$$

On suppose que la densité de probabilité est strictement positive. La fonction de répartition est alors continue et strictement croissante. De plus $F(a) = 0$ et $F(b) = 1$. Il s'en suit qu'elle admet une fonction inverse F^{-1} définie sur l'intervalle $[0, 1]$ et à valeurs dans $[a, b]$. Considérons alors une variable aléatoire U de densité uniforme sur l'intervalle $[0, 1]$ et soit la variable X définie par :

$$X = F^{-1}(U) \tag{34}$$

La probabilité d'obtenir x compris entre x_1 et x_2 est :

$$\begin{aligned} P(x_1 \leq x \leq x_2) =& P(x_1 \leq F^{-1}(u) \leq x_2) && (35)\\ =& P(F(x_1) \leq u \leq F(x_2)) && (36)\\ =& F(x_2) - F(x_1) && (37)\\ =& \int_{x_1}^{x_2} p(x)\,dx && (38) \end{aligned}$$

ce qui montre que X a la densité p. Si la fonction de répartition est inversible analytiquement, on peut obtenir l'échantillonnage à partir d'un échantillonnage uniforme sur l'intervalle $[0, 1]$.

Considérons par exemple une densité de probabilité proportionnelle à x sur l'intervalle $[0, 1]$:

$$p(x) = ax \tag{39}$$

En écrivant la condition de normalisation on obtient :

$$p(x) = 2x \tag{40}$$

La fonction de répartition est :

$$F(x) = \int_0^x 2x'\,dx' = x^2 \tag{41}$$

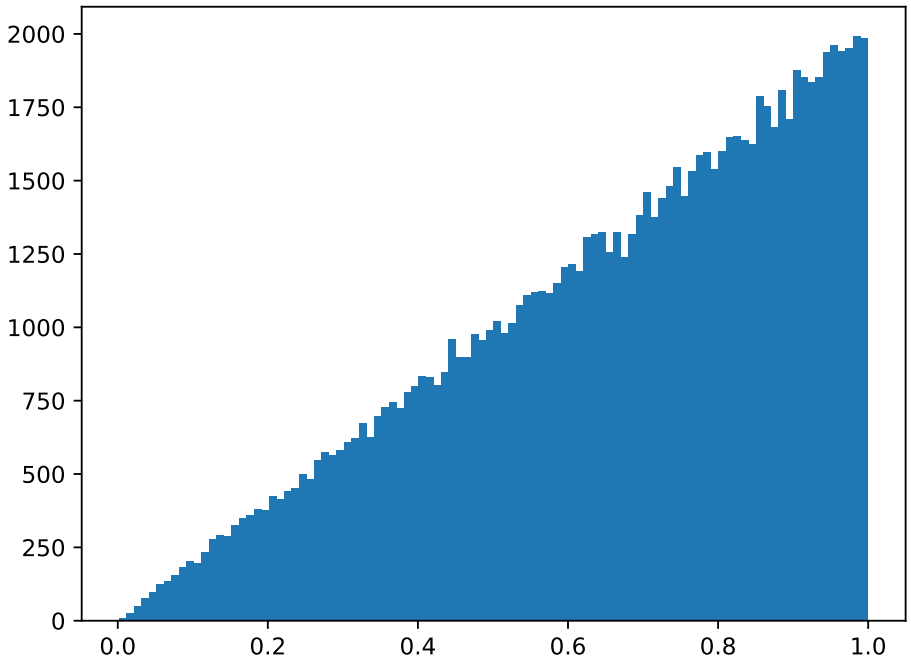
L'inversion s'écrit :

$$x = \sqrt{u} \tag{42}$$

où u est tiré aléatoirement avec une densité uniforme sur l'intervalle $[0, 1]$.

Pour tester l'échantillonnage, on fait un histogramme avec plusieurs milliers de tirages :

```
import numpy.random


N = 100000
x = numpy.sqrt(numpy.random.random_sample(N))
figure()
hist(x,100)
```



Si l'inverse de la fonction de répartition n'est pas disponible, on peut la calculer numériquement et la stocker dans une table (sous forme discrète).

Pour la génération de nombres suivant une loi discrète, voir Échantillonnage des distributions de probabilité.