Mathématiques Numériques MN1

Chapitre 2 : Simulations stochastiques et générateurs de nombres pseudo-aléatoires

Bruno Pinçon /J.-F. Scheid

Télécom Nancy 1A

2025-2026

Version du 16/06/2026

1. Simulations numériques de lois de probabilités
   - Introduction et principe général
   - Méthode de la transformée inverse
   - Loi normale : méthode de Box-Muller

2. Générateurs de nombres pseudo-aléatoires
   - Généralités
   - Générateur linéaire de congruence (LCG)
   - Tests des générateurs (test du χ^2)

I. Simulations numériques de lois de probabilités

1. Introduction et principe général.

Simuler numériquement une loi de probabilité d'une v.a. réelle X consiste à donner les valeurs $x_1, x_2, \ldots, x_n$ obtenues lors d'une réalisation d'un échantillon[1] $(X_1, X_2, \ldots, X_n)$ de X.

On veut simuler numériquement des réalisations d'une v.a. réelle dont on connait la fonction de répartition F[2]. On rappelle que

$$F(x) = \mathbb{P}(X \leq x), \ \forall x \in \mathbb{R}$$

et F possède les propriétés suivantes.

---

[1] v.a.i.i.d, variables aléatoires indépendantes et identiquement distribuées (de même loi)

[2] La loi d'une v.a. est complètement caractérisée par sa fonction de répartition

Propriétés d'une fonction de répartition

1. $\forall x \in \mathbb{R}, \quad 0 \leq F(x) \leq 1$
2. F est croissante.
3. $\lim_{x \to -\infty} F(x) = 0, \quad \lim_{x \to +\infty} F(x) = 1.$
4. [a] F est continue *à droite* en tout point $x \in \mathbb{R}$.
5. [b] F admet une limite finie *à gauche* en tout point $x \in \mathbb{R}$.

---

[a] $\forall x \in \mathbb{R}, \quad \lim\limits_{\substack{t \to x \\ t > x}} F(t) = F(x)$

[b] $\forall x \in \mathbb{R}, \quad \lim\limits_{\substack{t \to x \\ t < x}} F(t) = F(x) - \mathbb{P}(X = x) = \mathbb{P}(X < x)$

☛ **Importance de la loi uniforme.**

La loi uniforme joue un rôle important pour les simulations numérique d'un large ensemble de lois de probabilités. On peut simuler une loi donnée à partir de la loi uniforme $\mathcal{U}([0,1])$.

Plusieurs méthodes :

- méthode de la transformée inverse (fonction de répartition inverse ou fonction quantile)
- méthode de rejet (avec densité)

De façon plus particulière, pour la loi normale $\mathcal{N}(0,1)$:

- algorithme de Box-Muller
- méthode de rejet "ziggurat"

2. Méthode de la transformée inverse

a) Cas où la fonction de répartition est bijective

Dans le cas où la fonction de répartition F est continue et bijective, on a le résultat suivant :

Théorème d'inversion avec F bijective

Soit X une v.a. réelle de fonction de répartition $F : \mathbb{R} \to]0,1[$ et U une v.a. de loi uniforme sur $]0,1[$, $U \sim \mathcal{U}(]0,1[)$.
On suppose de plus que :

- F est continue sur $\mathbb{R}$
- F est *strictement* croissante sur $\mathbb{R}$.

Alors :

- F est bijective de $\mathbb{R}$ dans $]0,1[$.
- La v.a. réelle $V = F^{-1}(U)$ a pour fonction de répartition F i.e. V a même loi que X.

Remarque. Si X est une v.a. absolument continue de densité f telle que $f > 0$ (p.p.) alors sa fonction de répartition F est *strictement* croissante[3].

b) Cas général : fonction inverse généralisée.

Ce résultat se généralise au cas de v.a. réelles quelconques (dont F n'est pas forcément *strictement* croissante) en introduisant la **fonction inverse généralisée** (ou fonction *quantile*)

$$\forall y \in]0,1[, \quad F_g^{-1}(y) = \inf\{x \in \mathbb{R} : F(x) \geq y\} \tag{1}$$

• Si F est continue et *strictement* croissante sur $\mathbb{R}$, alors l'inverse généralisée F_g^{-1} coïncide avec l'inverse F^{-1}.

• On montre que la fonction inverse généralisée est croissante et vérifie:

$$\forall x \in \mathbb{R}, \ \forall y \in]0,1[, \quad F(x) \geq y \Leftrightarrow x \geq F_g^{-1}(y). \tag{2}$$

[3] On a $F(x) = \int_{-\infty}^{x} f(t)\,dt$

La propriété (2) permet de montrer le résultat suivant.

Théorème d'inversion généralisée

Soit X une v.a. réelle de fonction de répartition $F : \mathbb{R} \to [0, 1]$ et U une v.a. de loi uniforme sur $]0, 1[$, $U \sim \mathcal{U}(]0, 1[)$. Alors la v.a. réelle $V = F_g^{-1}(U)$ a pour fonction de répartition F i.e. V a même loi que X.

Application du théorème d'inversion.

Le théorème d'inversion permet de générer des réalisations $x_1, \ldots, x_n$ d'un échantillon d'une v.a. X de fonction de répartition F **dont on connait l'inverse** (généralisée) F_g^{-1}, à partir de réalisations de loi uniforme. On calcule

$$\boxed{x_n = F_g^{-1}(u_n)}$$

où u_n est la réalisation d'une v.a. $U \sim \mathcal{U}(]0, 1[)$.

Exemples.

1. *Loi exponentielle.* Soit $X \sim \mathcal{E}(\lambda)$ avec $\lambda > 0$. La densité de X est donnée par $f(x) = \lambda e^{-\lambda x} \mathbb{1}_{\mathbb{R}^+}(x)$ et sa fonction de répartition $F(x) = 1 - e^{-\lambda x}$ est une bijection de $\mathbb{R}$ dans $[0, 1[$ avec $F^{-1}(y) = -\frac{1}{\lambda} \ln(1 - y)$. Donc pour obtenir une réalisation de X, il suffit de poser
$$V = -\frac{1}{\lambda} \ln(U) \tag{3}$$
avec $U \sim \mathcal{U}([0, 1[)$ et V a même loi que X (U et $1 - U$ ont même loi).
2. *Loi de Cauchy.* Soit X une v.a. suivant la loi de Cauchy de paramètre $c > 0$ avec la densité $f(x) = \frac{c}{\pi(x^2+c^2)}$ et la fonction de répartition $F(x) = \frac{1}{\pi} \arctan\left(\frac{x}{c}\right) + \frac{1}{2}$. La v.a.
$$V = c \tan\left(\pi(U - \frac{1}{2})\right) \tag{4}$$
avec $U \sim \mathcal{U}(]0, 1[)$ a même loi que X

c) Cas d'une v.a. discrète finie

Soit X une v.a. discrète prenant les valeurs $x_1, \ldots, x_N$.

On introduit

$$y_k = \sum_{i=1}^{k} \mathbb{P}(X = x_i)$$

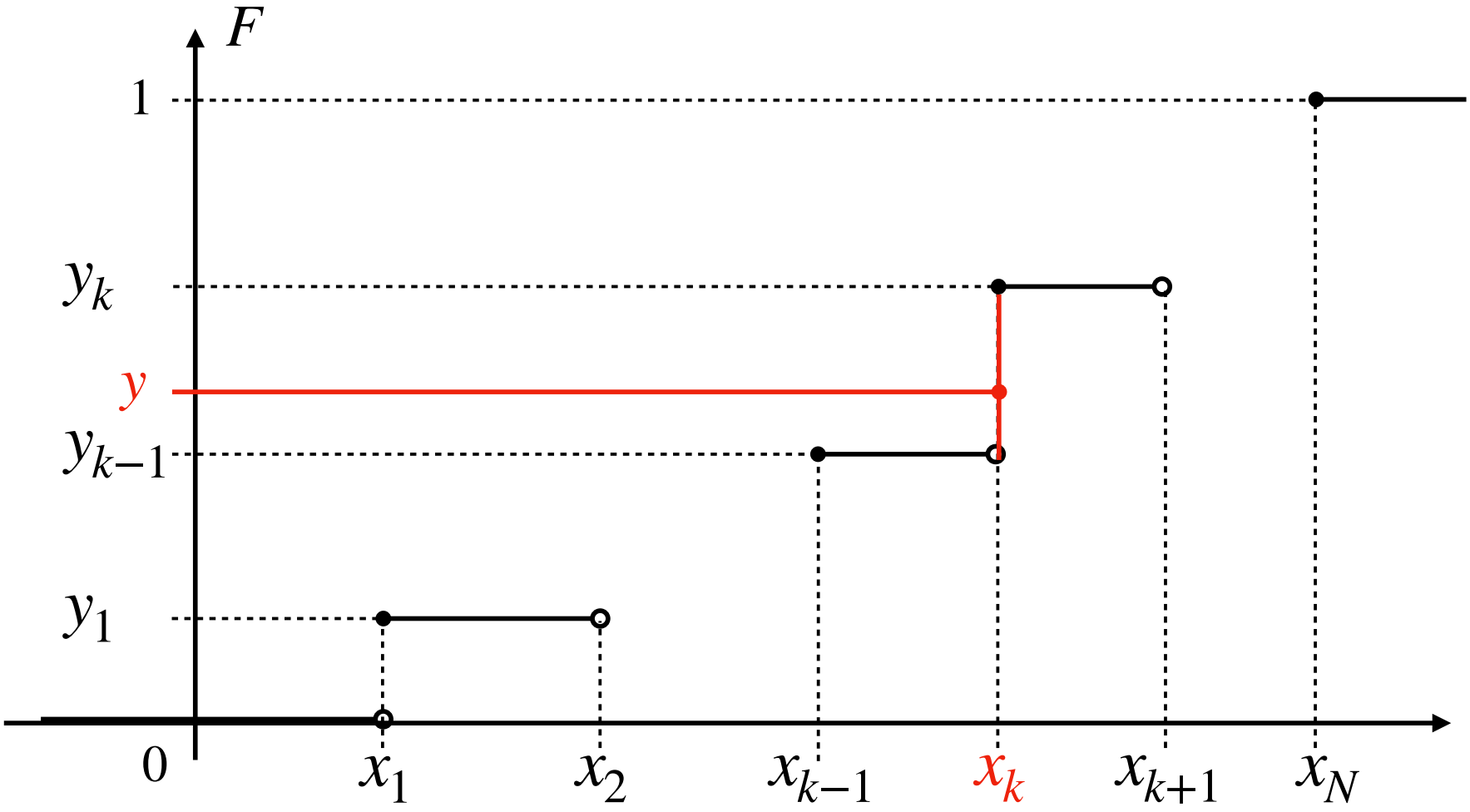
de sorte qu'on peut écrire la fonction de répartition F de X : pour tout $x \in \mathbb{R}$,

$$F(x) = \begin{cases} 0 & \text{si } x < x_1 \\ y_k & \text{si } x_k \leq x < x_{k+1} \quad (k \in [\![1, N-1]\!]) \\ 1 & \text{si } x \geq x_N \end{cases}$$

soit encore

$$F(x) = \sum_{i=1}^{N} y_k \mathbb{1}_{[x_k, x_{k+1}[}(x)$$

avec la convention $x_{N+1} = +\infty$.



On a $\boxed{F_g^{-1}(y) = x_k \text{ pour } y_{k-1} < y \leq y_k}$, soit encore

$$F_g^{-1}(y) = \sum_{k=1}^{N} x_k \mathbb{1}_{]y_{k-1}, y_k]}(y) \tag{5}$$

avec la convention $y_0 = 0$.

Exemples.

- *Loi de Bernoulli.*

  Soit $p \in]0,1[$. On a $V = \mathbb{1}_{U \leq p} \sim \mathcal{B}(p)$ si $U \sim \mathcal{U}(]0,1[)$.
- *Loi binomiale.* D'après la formule (5), la v.a.

$$V = \sum_{k=1}^{n} k \mathbb{1}_{\{y_{k-1} < U \leq y_k\}}$$

avec $U \sim \mathcal{U}(]0,1[)$, suit la loi binomiale $\mathcal{B}(n,p)$ avec

$$y_k = \sum_{i=1}^{k} C_n^i p^i (1-p)^{n-i}$$

Il est nécessaire de précalculer les y_k et pour une réalisation u de U, il faut déterminer l'intervalle $]y_{k-1}, y_k]$ contenant u. Cela peut s'avérer coûteux. Une alternative possible est de considérer la somme de n (réalisations de) variables qui suivent la loi de Bernoulli $\mathcal{B}(p)$.

Remarque : méthode de rejet.

Il peut être difficile, voire impossible de déterminer exactement l'inverse généralisée (loi normale, par exemple). Une alternative possible utilisant aussi la loi uniforme est la *méthode de rejet*.

Soit X une v.a. absolument continue de densité f dont on veut obtenir des réalisations. On suppose que X est à valeurs dans $[a, b]$ et que f est bornée par une constante c. Soit (Y, U) un couple de v.a. *indépendantes* et

$$(Y, U) \sim \mathcal{U}(]a, b[\times]0, c[)$$

i.e. un point tiré uniformément dans le rectangle $]a, b[\times]0, c[$.

On peut montrer que X suit la loi conditionnelle de Y sachant l'évènement

$$M = \{U \leq f(Y)\}$$

☛ Pour obtenir des réalisations d'un échantillon de X, il suffit donc d'effectuer une suite de tirages de couples (Y_i, U_i) uniformes dans le carré unité et indépendants puis de sélectionner les $X_i = Y_i$ vérifiant M et de *rejeter* les autres.

3. Loi normale : méthode de Box-Muller.

Soient U et V deux v.a. indépendantes, uniformes sur $]0,1]$, $U, V \sim \mathcal{U}(]0,1])$. On définit les v.a.

$$\begin{aligned} X &= \sqrt{-2\ln(U)}\,\cos(2\pi V) \\ Y &= \sqrt{-2\ln(U)}\,\sin(2\pi V) \end{aligned} \tag{6}$$

Alors X et Y sont indépendantes et suivent la loi normale centrée réduite $\mathcal{N}(0,1)$.

On utilise donc les formules (6) pour simuler la loi normale centrée réduite.

Remarques.

- On pose
$$\begin{cases} R = \sqrt{-2\ln(U)} \\ \Theta = 2\pi V \end{cases}$$
Alors $\Theta \sim \mathcal{U}(]0, 2\pi])$ et $R \sim \mathcal{R}(1)$ loi de Rayleygh de paramètre 1 dont la densité vaut $f(r) = re^{-\frac{r^2}{2}} \mathbb{1}_{\mathbb{R}+}(r)$.
- Il existe une autre version plus efficace dite *méthode polaire* qui n'utilise pas les fonctions cos et sin (mais qui utilise un peu plus de nombres de $\mathcal{U}(]0, 1])$.
- La méthode la plus efficace pour simuler $\mathcal{N}(0, 1)$ est la méthode "ziggurat" qui est une méthode de rejet. C'est cette méthode qui est couramment utilisée dans les bibliothèques numériques.

II. Générateurs de nombres pseudo-aléatoires

1. Généralités.

On cherche à simuler des réalisations indépendantes de la loi uniforme sur $[0, 1]$. On veut obtenir des nombres dits *pseudo-aléatoires* vérifiant de bonnes propriétés statistiques en adéquation avec la loi uniforme.

Un générateur de nombres pseudo-aléatoires est un algorithme **déterministe** généralement construit à partir d'une récurrence. Un générateur possède un état propre qui doit être initialisé. L'état initial est appelé *germe* ou *graine* (*seed* en anglais).

Un générateur est la composition de deux fonctions.

1. **fonction de transition** $f : \mathcal{S} \to \mathcal{S}$ où $\mathcal{S}$ est l'espace d'état fini du générateur avec $s_0 \in \mathcal{S}$ le *germe* (*seed*). Cette fonction de transition permet de mettre à jour l'état du générateur avec

$$s_n = f(s_{n-1})$$

Les états s_n sont représentés par un ou plusieurs nombres *entiers*.

2. **fonction de sortie** $g : \mathcal{S} \to \mathcal{U} \subset [0, 1[$ où $\mathcal{U}$ est l'espace de sortie constitué des nombres pseudo-aléatoires :

$$u_n = g(s_n)$$

Les séquences de nombres aléatoires sont généralement périodiques mais avec un très grande période.

$$\begin{array}{ccccccccccc} s_0 & \xrightarrow{f} & s_1 & \xrightarrow{f} & \cdots & \xrightarrow{f} & s_n & \xrightarrow{f} & s_{n+1} & \xrightarrow{f} & \cdots \\ g\downarrow & & g\downarrow & & & & g\downarrow & & g\downarrow & & \\ u_0 & & u_1 & & \cdots & & u_n & & u_{n+1} & & \end{array}$$

2. **fonction de sortie** $g : \mathcal{S} \to \mathcal{U} \subset [0, 1[$ où $\mathcal{U}$ est l'espace de sortie constitué des nombres pseudo-aléatoires :

$$u_n = g(s_n)$$

Les séquences de nombres aléatoires sont généralement périodiques mais avec un très grande période.

$$\begin{array}{ccccccccccccc} \cdots & \xrightarrow{f} & s_{\rho-1} & \xrightarrow{f} & s_0 & \xrightarrow{f} & s_1 & \xrightarrow{f} & \cdots & \xrightarrow{f} & s_n & \xrightarrow{f} & s_{n+1} & \xrightarrow{f} & \cdots \\ & & g\downarrow & & g\downarrow & & g\downarrow & & & & g\downarrow & & g\downarrow & & \\ & & u_{\rho-1} & & u_0 & & u_1 & & \cdots & & u_n & & u_{n+1} & & \end{array}$$

Période d'un générateur

On dit que le générateur a pour *période* $\rho \leq |\mathcal{S}|$ s'il existe $i_0 \geq 0$ tel que $s_{i+\rho} = s_i,\ \forall i \geq i_0$.

2. Générateur linéaire de congruence (LCG).

C'est un générateur simple et largement utilisé. Pour des entiers positifs m, a, c avec $0 \leq a, c < m$, il est défini par

$$\begin{cases} s_0 \in [\![0, m-1]\!], \quad \text{(le germe)} \\ s_n = a s_{n-1} + c \text{ modulo } m, \quad \forall n \geq 1 \end{cases} \tag{7}$$

puis

$$u_n = s_n / m \tag{8}$$

i.e. $u_n \in \{k/m,\ k \in [\![0, m-1]\!]\} \subset [0, 1[$ de sorte que 1 n'est jamais obtenu.

Dans un LCG, a est le multiplicateur,
c est l'incrément,
m est le module.

On prend généralement $m = 2^p$ avec p entier lié au format des nombres entiers utilisés.

Par exemple, pour la librairie `glibc` (utilisé avec `gcc`), les paramètres sont $a = 1103515245$, $c = 12345$, $m = 2^{31}$.

Choix des paramètres.

Pour pouvoir simuler correctement l'aléatoire, le générateur LCG doit posséder une grande période. La période maximale d'un LCG est m (à cause du modulo). On peut choisir les paramètres a et c pour que la période soit *maximale* et égale à m.

Proposition (Hull-Dobell, 1962)

La suite du générateur LCG est de période m (quelque soit s_0) si et seulement si

i) c et m sont premiers entre eux (pas de diviseurs communs autre que 1).

ii) $a - 1$ est un multiple de p si p est un nombre premier qui divise m.

iii) $a - 1$ est un multiple de 4 si m est un multiple de 4.

Exemples. Avec $(a = 5, c = 1, m = 12)$, la période vaut 4 $(< m)$.

Avec $(a = 13, c = 1, m = 24)$, la période est maximale 24.

Remarques.

- *Conditions alternatives.* Dans le cas où m est une puissance de 2 avec

$$m = 2^e, \ e \geq 2,$$

les 3 conditions de la proposition précédente donnant une période maximale sont équivalentes à :

  - c impair
  - $a = 1 \mod 4$

- Si la période n'est pas maximale, elle dépend a priori du germe initial s_0. Par exemple, avec le générateur ($a = 2$, $c = 0$, $m = 6$), on obtient une période $\rho = 2$ avec $s_0 = 1$ et une période $\rho = 1$ avec $s_0 = 3$.

Quelques générateurs standards.

De nombreux générateurs existent. Citons-en quelques-uns parmi les plus utilisés (disponibles dans `numpy`).

- MT199937 (Mersenne-Twister). La fonction de transition f est une transformation compliquée dans un espace matriciel, la fonction de sortie est simple comme (8). C'est le générateur le plus répandu actuellement (utilisé par défaut dans beaucoup de logiciels). Il a une période de $2^{19937} - 1$.
- PCG64. La fonction de transition est linéaire comme (7) et la fonction de sortie est une fonction de permutation bien choisie. C'est le générateur par défaut dans `numpy` et serait à l'heure actuelle l'un des meilleurs générateurs. Il a une période de 2^{64}.
- Philox. La fonction de transition est très simple: l'état est le numéro de l'itération n. La fonction de sortie est compliquée, similaire à des fonctions de chiffrement par blocs utilisées en cryptographie.

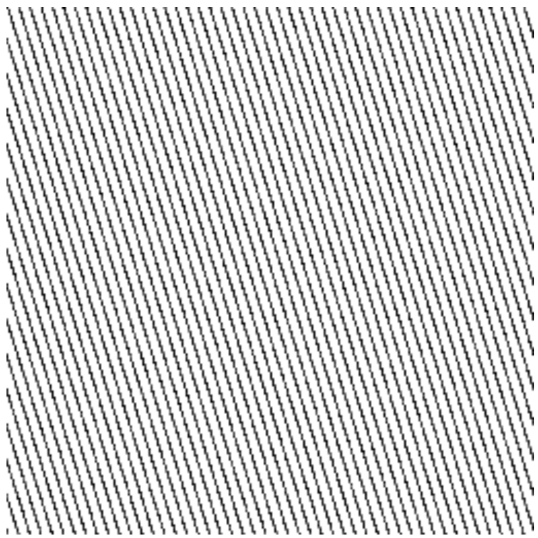
Reproduire une simulation. Quand on fait une simulation stochastique, on peut vouloir la reproduire exactement en engendrant les mêmes nombres. Dans `numpy`, le générateur est fixé par défaut (PCG64) et l'état initial est déterminé de manière aléatoire lors du premier appel (en utilisant par exemple la date et l'heure avec seconde, dixième et centièmes). Si vous relancez la simulation, il y a très peu de chance que vous retombiez sur le même état si vous quittez python et le relancez. On peut fixer l'état initial avec la fonction `seed` du sous-module `random` de `numpy`.

```
>>> import numpy as np
>>> np.random.seed(3) # on fixe l'état initial
>>> n=5; a=10; b=20
>>> np.random.randint(a,b,n) # n nombres aléatoires uniformes
                             # dans [a,b[
```

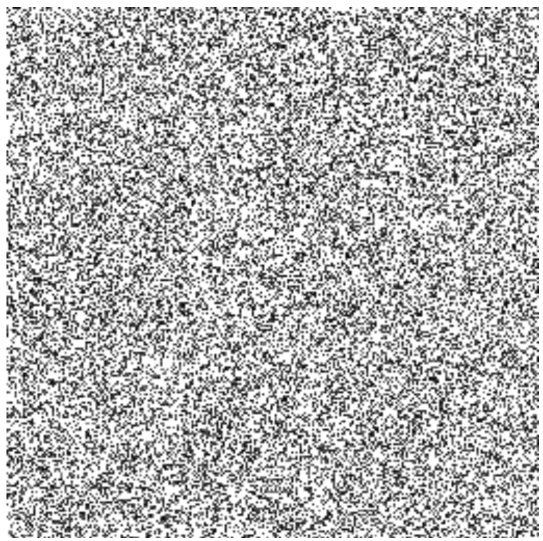
Les nombres ainsi engendrés seront toujours les mêmes.

3. Tests des générateurs.

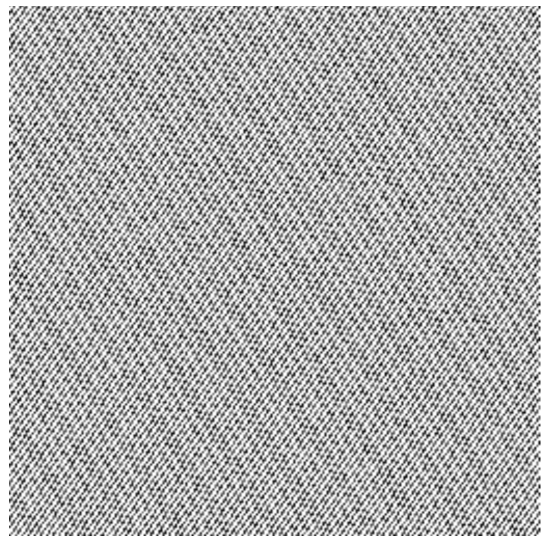
Exemple[4] d'un tirage uniforme d'un couple (x, y) sur une grille $2^8 \times 2^8$ avec un LCG avec $m = 2^{16}$.



Un mauvais LCG



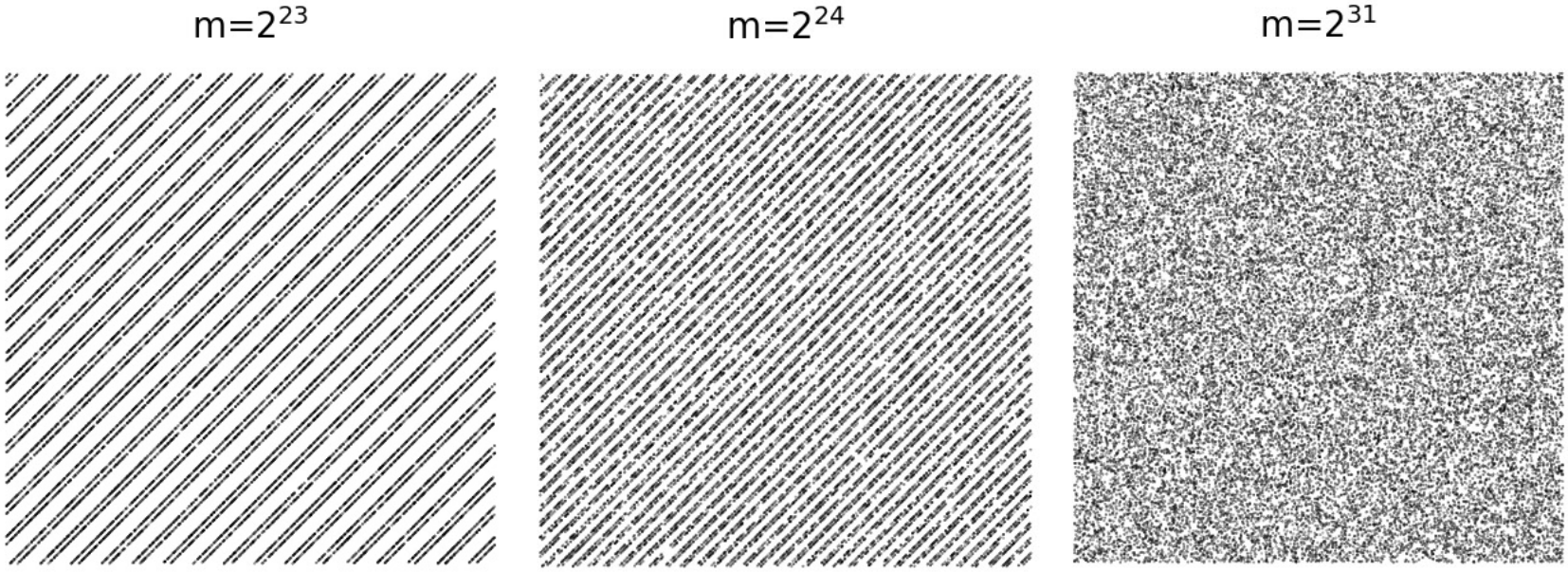
Du "vrai" hasard



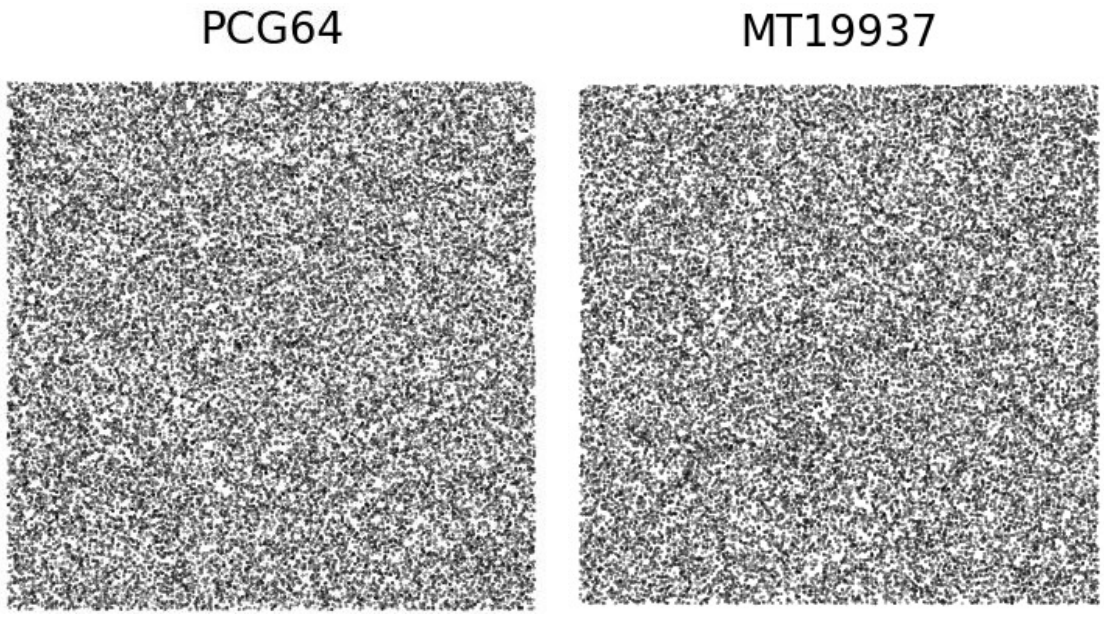
Un "bon" LCG

[4] Melissa E. O'Neill, *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation*, Harvey Mudd College Computer Science Department, Technical report, 2014.
`https://www.pcg-random.org/pdf/hmc-cs-2014-0905.pdf`

Un autre exemple de LCG avec $a = 13$, $c = 1$ et différentes valeurs du module $m = 2^e$ (période maximale m).



... et aussi



Pour évaluer l'efficacité d'un générateur de façon *plus quantitative*, on soumet les générateurs à de nombreux tests statistiques.

- Test du Khi2 et/ou Kolmogorov-Smirnov[5] d'adéquation à la loi uniforme (ou une loi donnée).
- Tests d'indépendance et de non-corrélation.
- Tests sur des problèmes classiques de probabilités. Les plus connus sont :
  - Diehard (Maraglia)
  - TestU01 (L'Ecuyer)
  - Dieharder (Brown, Eddelbuettel et Bauer)

[5] Le test de Komolgorov-Smirnov compare la fonction de répartition observée avec celle attendue (loi uniforme).

Test du χ^2 d'adéquation à la loi uniforme.

Ce test permet de vérifier si les fréquences observées des nombres produits par un générateur sont en adéquation avec les fréquences théoriques attendues pour une loi uniforme.

On veut vérifier l'hypothèse suivante.

Hypothèse nulle H_0 : *le générateur produit des nombres qui suivent une distribution uniforme.*

On va comparer les effectifs *observés* (valeurs issues du générateur) et *théoriques*. Pour cela, on génère n valeurs $x_1, \ldots, x_n$ dans $[0, 1[$ par le générateur.

Découpage des valeurs en classes. On subdivise l'intervalle $[0, 1[$ en k classes uniformes (sous-intervalles de même longueur). Le choix de k est important, les classes devant contenir suffisamment de données (mais pas trop !). Un critère souvent retenu pour que le test soit valide est le suivant (critère de Cochran) :

- Pour chaque classe, effectif théorique ≥ 1.
- 80% des classes doivent avoir des effectifs théoriques ≥ 5.

Statistique χ^2 du test.

• On note O_i le nombre de valeurs observées (issues du générateur) dans la classe i.

• Sous l'hypothèse nulle H_0, la probabilité qu'une valeur appartienne à une classe donnée est la même pour toutes les classes et vaut $p = \frac{1}{k}$. L'effectif théorique de la classe i vaut donc

$$E_i = np = \frac{n}{k}$$

La statistique du χ^2 est définie par :

$$T = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{9}$$

soit encore $T = \sum_{i=1}^{k} \frac{(O_i - n/k)^2}{n/k}$.

Test de conformité au risque α.

Sous l'hypothèse nulle H_0, la variable T suit asymptotiquement ($n \to +\infty$) une loi du χ^2_{k-1} à $(k-1)$ degrés de liberté[6].

On *rejette* alors l'hypothèse (H_0) quand[7]

$$\boxed{T \geq \chi^2_{1-\alpha,k-1} := F^{-1}_{\chi^2_{k-1}}(1-\alpha)} \tag{10}$$

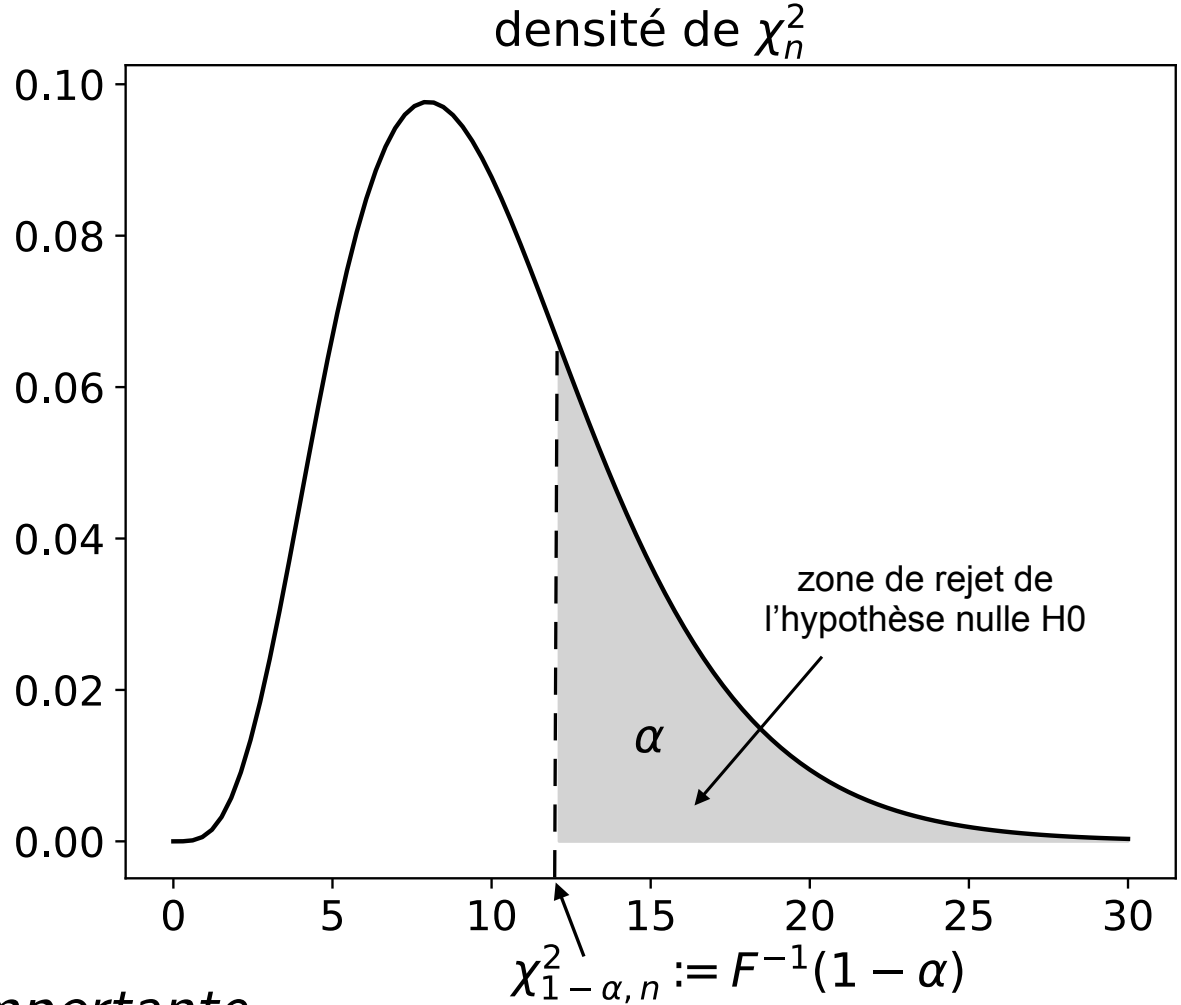
où $\chi^2_{1-\alpha,k-1}$ est le quantile d'ordre $1-\alpha$ de la loi χ^2_{k-1} (cf. table).

---

[6] *Loi du χ^2.* Soit k v.a. $X_1, \ldots, X_k$ indépendantes suivant la loi normale $\mathcal{N}(0,1)$. Alors la variable $X = \sum_{i=1}^{k} X_i^2$ suit la loi du χ^2 à k degrés de liberté : $X \sim \chi^2_k$.

[7] Pour tester H_0, on peut aussi utiliser la *p-value* définie par p-value $= \mathbb{P}(X \geq T)$ où $X \sim \chi^2_{k-1}$. Si p-value $< \alpha$, on rejette H_0.

Autrement dit, on rejette l'hypothèse nulle si $\mathbb{P}(T \geq \chi^2_{1-\alpha,k-1}) \geq \alpha$



Remarque importante.

• Echouer à un test du χ^2 (rejet de H_0) indique des déviations significatives par rapport à la loi uniforme.

• Mais attention ! Réussir un test du χ^2 ne garantit pas forcément un bon générateur.

Exemple. Générateur LCG ($a = 5$, $c = 1$, $m = 2^{16}$).
On choisit :

- germe $s_0 = 0$
- taille de l'échantillon : $n = 1000$
- nombre de classes : $k = 10$ (classes : $[0, 0.1[, [0.1, 0.2[, \ldots, [0.9, 1[$)
- niveau de risque : $\alpha = 5\%$

· *Données engendrées*

Classe	Intervalle	O_i
1	[0, 0.1[	108
2	[0.1, 0.2[	92
3	[0.2, 0.3[	114
4	[0.3, 0.4[	97
5	[0.4, 0.5[	75
6	[0.5, 0.6[	102
7	[0.6, 0.7[	79
8	[0.7, 0.8[	121
9	[0.8, 0.9[	109
10	[0.9, 1[	103
Total		1000

· *Effectifs théoriques*

$$E_i = n/k = 1000/10 = 100$$

· *Statistique du* χ^2

$$T = \sum_{i=1}^{10} \frac{(O_i - 100)^2}{100} = 19.34$$

· *Quantile d'ordre* $1 - \alpha = 0.95$ *de la loi* χ^2_9 *(à 9 degrés de liberté)*

$$\chi^2_{0.95,\,9} \simeq 16.92 \leq T$$

$\Rightarrow$ on rejette l'hypothèse (H_0), ce n'est pas un bon générateur.

Remarques et conclusion. Le χ^2 permet aussi d'autres tests :

- Test d'indépendance : 2 variables sont-elles indépendantes (absence de lien statistique entre elles) ?
- Test d'homogénéité : 2 échantillons proviennent-ils de deux variables aléatoires suivant la même loi ?

Le test du χ^2 est un test "basique" qui ne permet pas de décider si on a un "bon" générateur. Il permet avant tout de déterminer si on a un (très) mauvais générateur (rejet de l'hypothèse nulle).

Par exemple, le LCG avec ($a = 13$, $c = 1$, $m = 2^{24}$) qui présente pourtant des structures déterministes (cf. page 25) a un score $T = 6.12 < \chi^2_{0.95,\,9} = 16.92$ (avec $s_0 = 0$ et $k = 10$). On ne rejette donc pas H_0 avec ce test du χ^2.