



# Chapitre 2

## Random numbers and Monte Carlo simulation

2025-2026

Niveau : L2-INFO

# Monte Carlo simulation

- La simulation de Monte Carlo repose sur des nombres aléatoires pour simuler le processus plusieurs fois et évaluer la variabilité.

- **Étapes de mise en œuvre :**

1. Définir le problème
2. Génération de nombres aléatoires
3. Simuler le processus
4. Analyser les résultats

# Random Number Generation (RNG's)

## 1. Introduction

Les nombres aléatoires sont largement utilisés dans les simulations, la cryptographie et l'échantillonnage statistique. Ils sont soit :

- True Random Numbers
- Pseudorandom Numbers

# Random Number Generation (RNG's)

## 2. Types de nombres aléatoires (True vs. Pseudorandom)

- True Random Numbers :
  - Générés à partir d'un phénomène physique (par exemple, bruit thermique).
  - Sont non déterministes → il n'existe aucune formule mathématique ou algorithme capable de prédire le prochain nombre de la séquence.

# Random Number Generation (RNG's)

## 2. Types de nombres aléatoires (True vs. Pseudorandom)

- Pseudorandom Numbers :
  - Générés par des algorithmes, et bien qu'ils paraissent aléatoires, ils sont déterministes en fonction d'une valeur initiale "graine" (seed).

# Random Number Generation (RNG's)

## 3. Exigences pour les nombres aléatoires

Distribution uniforme (continue) entre 0 et 1 :

Pourquoi 0 et 1 ? → ils peuvent être facilement transformés en nombres aléatoires dans n'importe quelle plage souhaitée ou même en nombres suivant d'autres distributions.

Les nombres dans la séquence sont indépendants les uns des autres.

# Random Number Generation (RNG's)

**Les générateurs de nombres aléatoires (RNG) dans les simulations informatiques sont pseudo-aléatoires.**

- Les nombres sont générés algorithmiquement et déterminés par leurs prédecesseurs (par exemple, le générateur linéaire congruentiel (LCG)).
- Des tests statistiques peuvent être utilisés pour vérifier la randomisation (uniformité et indépendance).

# Propriétés des nombres aléatoires

Deux propriétés statistiques importantes :

- Uniformité
- Indépendance

Fonction de densité de probabilité (PDF) pour une distribution uniforme

Les nombres aléatoires,  $x_1, x_2, x_3, \dots$  doivent être tirés indépendamment d'une distribution uniforme avec une fonction de densité de probabilité (PDF) :

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

# Desirable Properties of (RNGs)

- Uniformity
- Independence
- Reproducibility : RNGs should have the capability to reproduce a given sequence of random numbers when initialized with the same "seed" value.
  - Helps program debugging
  - Helpful when comparing alternative system design
- Should have provision to generate several streams of random numbers
- Computationally efficient

# Linear Congruential Generator (LCG)

- Un algorithme mathématique pour générer des séquences de nombres pseudo-aléatoires dans les programmes informatiques. Une sequence  $x_1, x_2, \dots$  entre 0 et  $m-1$  est générée en se basant sur :

$$X_{i+1} = (aX_i + c) \bmod m, \quad i = 0, 1, 2, \dots$$

The diagram shows the LCG formula  $X_{i+1} = (aX_i + c) \bmod m$ . Three callout boxes point to the components: 'The multiplier' points to  $a$ , 'The increment' points to  $c$ , and 'The modulus' points to  $m$ .

- Par exemple, mélanger un jeu de cartes dans un jeu informatique utilise des nombres pseudo-aléatoires pour réorganiser les cartes de manière aléatoire.
- Reproductibilité : La séquence générée par un LCG est déterministe.

# Linear Congruential Generator (LCG)

- Exemple :  $x_n = 5x_{n-1} + 1 \pmod{16}$

Then the first few random numbers would be computed as:

- $x_1 = (5 \cdot 5 + 1) \pmod{16} = 26 \pmod{16} = 10$
- $x_2 = (5 \cdot 10 + 1) \pmod{16} = 51 \pmod{16} = 3$
- $x_3 = (5 \cdot 3 + 1) \pmod{16} = 16 \pmod{16} = 0$
- and so on...

Les 32 premiers nombres obtenus par la procédure ci-dessus 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4

Divisé par 16 pour les normaliser dans l'intervalle [0, 1].

0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125, 0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125

# Linear Congruential Generator (LCG)

- $a=13$ ,  $c=0$ , et  $m=64$ .
- Changer  $X_0$  produirait une séquence différente, montrant que la graine est cruciale.

La période est-elle toujours égale à  $m$ ?

$i$	$X_i$ $X_0=1$	$X_i$ $X_0=2$	$X_i$ $X_0=3$	$X_i$ $X_0=4$
0	1	2	3	4
1	13	26	39	52
2	41	18	59	36
3	21	42	63	20
4	17	34	51	4
5	29	58	23	
6	57	50	43	
7	37	10	47	
8	33	2	35	
9	45		7	
10	9		27	
11	53		31	
12	49		19	
13	61		55	
14	25		11	
15	5		15	
16	1		3	

# Linear Congruential Generator (LCG)

- Génère des séquences déterministes basées sur la graine et les paramètres.
- **Périodicité** : La séquence finit par se répéter ; la période dépend de  $a$ ,  $c$ ,  $m$  et de la graine.
- **Efficace** : Simple et rapide pour générer des nombres aléatoires.
- **Non adapté à la cryptographie** : Séquences prévisibles, inadaptées aux applications de sécurité.
- **Application** : Utilisé couramment dans les simulations (ex. Monte Carlo) et les jeux pour une génération rapide de nombres aléatoires.

# Propriétés LCG

- Un  $m$  plus grand augmente la période maximale possible et procure une meilleure approximation d'une distribution uniforme continue.
- La qualité d'un LCG dépend fortement de ses paramètres  $a$ ,  $c$  et  $m$ .
- De mauvais choix peuvent entraîner des séquences avec des périodes très courtes ou des séquences présentant des motifs évidents, compromettant leur caractère aléatoire.
- L'approximation semble avoir peu d'importance.

# Caractéristiques d'un bon générateur

- **Densité Maximale** : Les valeurs sont uniformément réparties sur l'intervalle  $[0,1]$ .
- **Période Maximale** : Une longue période, signifiant qu'elle doit produire de nombreuses valeurs distinctes avant de répéter la séquence.
- **Évitements des Cycles.**
- **Efficacité avec les Systèmes Binaires:** La vitesse et l'efficacité sont améliorées en utilisant un modulus  $m$  qui est (ou proche de) une puissance de 2.

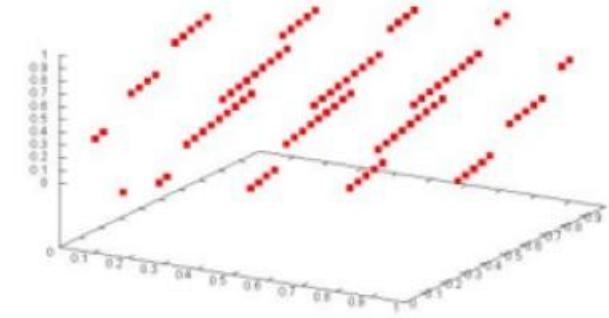
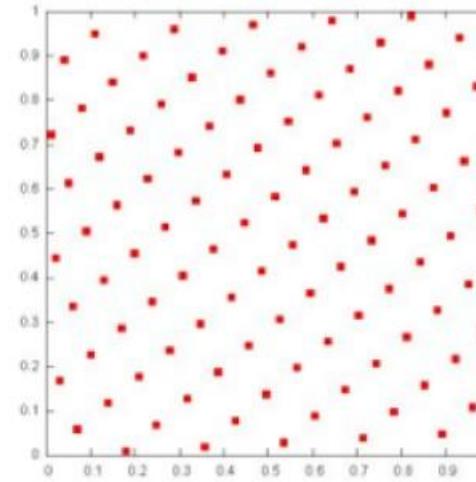
# Monte Carlo & LCG

Pour Monte Carlo, la graine n'a pas besoin d'être "parfaite", mais la suite doit être suffisamment longue et bien répartie pour simuler l'aleatoire.

# Monte Carlo & LCG

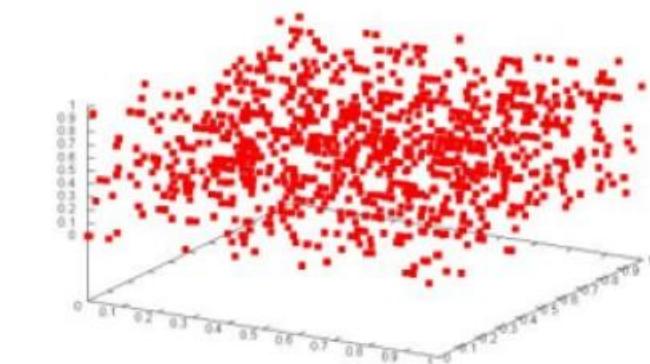
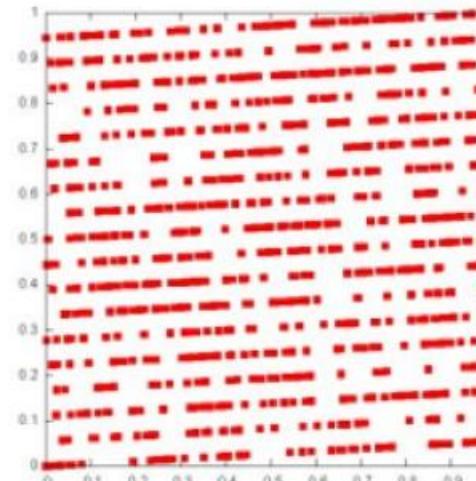
## Haut (Mauvais générateur) :

- Des alignements subtils ou des grilles, indiquant une corrélation ou une période courte.
- Le générateur produit des séquences prévisibles ou cycliques, avec des motifs évidents



## Bas (Bon générateur) :

- Les points semblent couvrir l'espace de manière homogène, ce qui reflète une bonne densité maximale.
- Des points dispersés sans alignements ou structures apparentes → une longue période et une absence de cycles précoces.



# Types des LCG

- **Mixed LCG (LCG Additif)**

- $c > 1$
- Les LCG mixtes peuvent avoir une période de  $m$  si tous les paramètres sont choisis correctement.

- **Multiplicative LCG (LCG Multiplicatif)**

- $c = 0$
- Il n'y a pas de terme additif, simplifiant les calculs, mais imposant des conditions plus strictes sur  $a$  pour garantir une bonne période.

→ Un bon générateur de nombres aléatoires doit trouver un équilibre entre : une période complète, une distribution uniforme, et des nombres aléatoires indépendants.

# Types des LCG

Un LCG multiplicatif actuellement populaire est donné :

$$x_n = 7^5 x_{n-1} \bmod (2^{31} - 1)$$

**Période complète** : Cela signifie que le générateur peut produire une séquence de nombres distincts avant que la séquence ne commence à se répéter.

Full period :  $2^{31}-2$

Ce générateur a été étudié, testé, et prouvé de donner de bons résultats pour de nombreuses applications.

# Principes clés pour le choix des graines

- Éviter les graines aléatoires (Random Seeds)
- Ne pas utiliser zéro
- Éviter les valeurs paires
- Ne pas utiliser des graines successives

# Exemple

- Le **carré** (côté 2, aire 4) représente l'espace total où les points sont générés aléatoirement (de -1 à 1 sur x et y ).
- Le **cercle** (rayon 1, aire  $\pi$ ) est inscrit dans ce carré, et son aire ( $\pi$ ) est la cible de l'estimation.

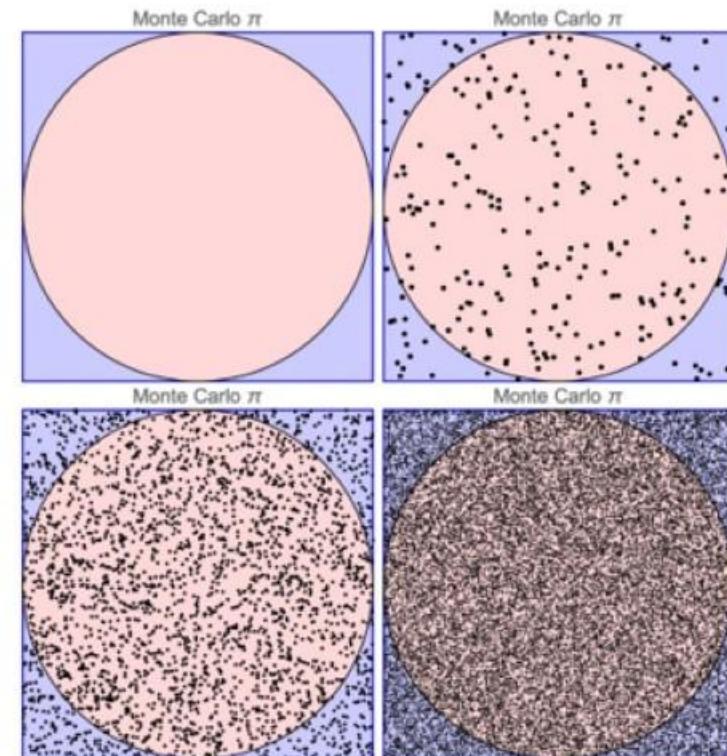
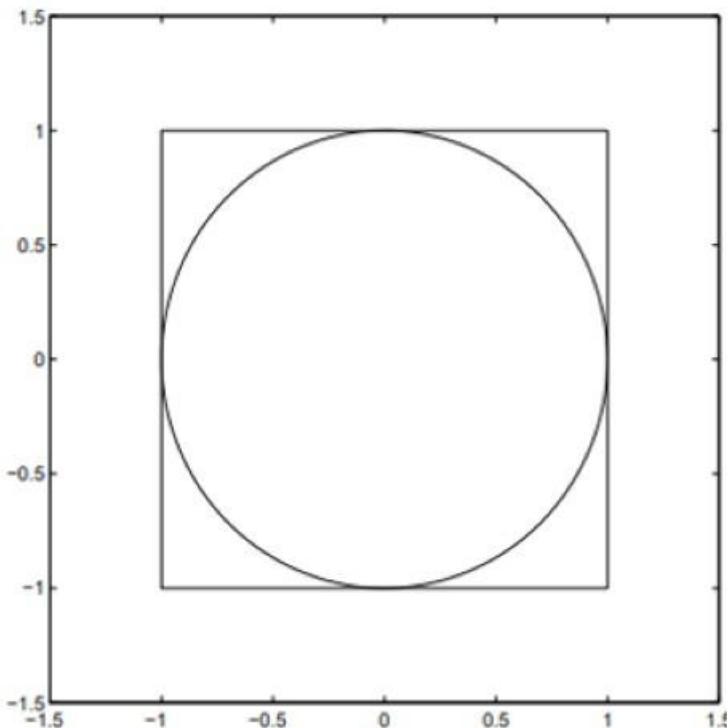


Figure 2: Squares with zero, 250, 2500 and 25000 points.

La proportion de points tombant dans le cercle par rapport au carré est  $\pi/4$  (car l'aire du cercle est  $\pi$ , et celle du carré est 4). Ainsi,  $\pi \approx 4 \cdot \frac{\text{points dans le cercle}}{\text{points totaux}}$ .

# Exemple

Résultats avec LCG :

N = 1000: Estimation de  $\pi$  = 3.148000 (Erreur = 0.006408)  
N = 10000: Estimation de  $\pi$  = 3.106800 (Erreur = 0.034792)  
N = 100000: Estimation de  $\pi$  = 3.133720 (Erreur = 0.007872)

Résultats avec random.random :

N = 1000: Estimation de  $\pi$  = 3.172000 (Erreur = 0.030408)  
N = 10000: Estimation de  $\pi$  = 3.122400 (Erreur = 0.019192)  
N = 100000: Estimation de  $\pi$  = 3.146640 (Erreur = 0.005048)

→ Autre algorithme : Mersenne Twister

random.random()

$2^{19937} \approx 10^{1800}$

# Principes clés pour le choix des graines

- Médecine nucléaire [1] : Simuler rayons X dans tissu, 10 000 trajectoires, dose moyenne, planification cancer.
- Finance : Estimer valeur action, 1 000 scénarios, moyenne, prédictions marchés.
- Jeux vidéo/IA : Générer environnements aléatoires, placement objets, mondes procéduraux.
- Optimisation : Trouver chemin labyrinthe, 500 parcours, meilleur chemin.
- Climatologie : Prédire température, 10 000 scénarios, moyenne prévisions.

[1] SEPEHRI, Fatemeh, HAJIVALIEI, Mahdi, et RAJABI, Hossein. Selection of random number generators in GATE Monte Carlo toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 2020, vol. 973, p. 164172.