Chapitre 2
Random numbers and Monte Carlo simulation

2025-2026

Niveau : L2-INFO

hajmessaoud.nada@gmail.com

Dr. Nada Haj Messaoud

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).

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 prédécesseurs (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, x1,x2,x3,… doivent être tirés indépendamment d'une distribution uniforme avec une fonction de densité de probabilité (PDF) :

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

Linear Congruential Generator (LCG)

- Un algorithme mathématique pour générer des séquences de nombres pseudo-aléatoires dans les programmes informatiques. Une sequence x1,x2,… entre 0 et m-1 est géneéé en se basant sur :

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

The multiplier — The increment — The modulus

- 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 :** $x_n = 5x_{n-1} + 1 \mod 16$

Then the first few random numbers would be computed as:

- $x_1 = (5 \cdot 5 + 1) \mod 16 = 26 \mod 16 = 10$
- $x_2 = (5 \cdot 10 + 1) \mod 16 = 51 \mod 16 = 3$
- $x_3 = (5 \cdot 3 + 1) \mod 16 = 16 \mod 16 = 0$
- 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 X0 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.
- **Évitement 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.

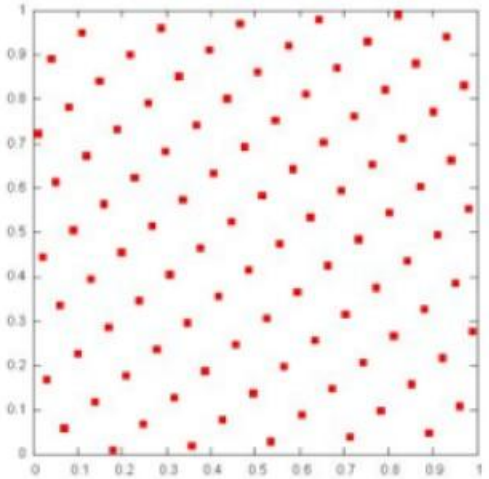
Monte Carlo & LCG

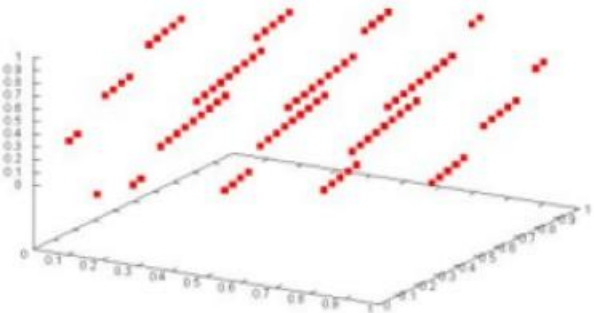
Pour Monte Carlo, la graine n'a pas besoin d'être "parfaite", mais la suite doit être suffisamment longue et bien répartie pour simuler l'aléatoire.

Monte Carlo & LCG

Haut (Mauvais générateur) :

- Des alignements subtils ou des grilles, indiquant une corrélation ou une période courte.

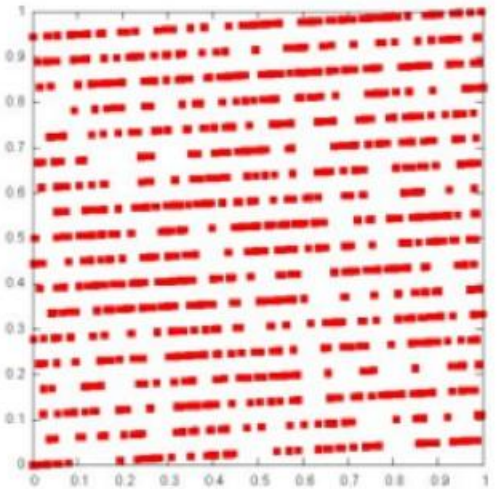


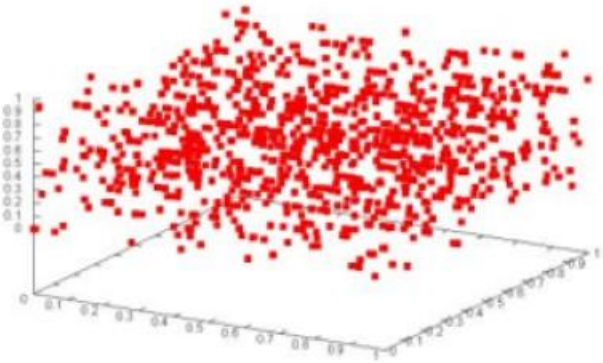


- Le générateur produit des séquences prévisibles ou cycliques, avec des motifs évidents

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.

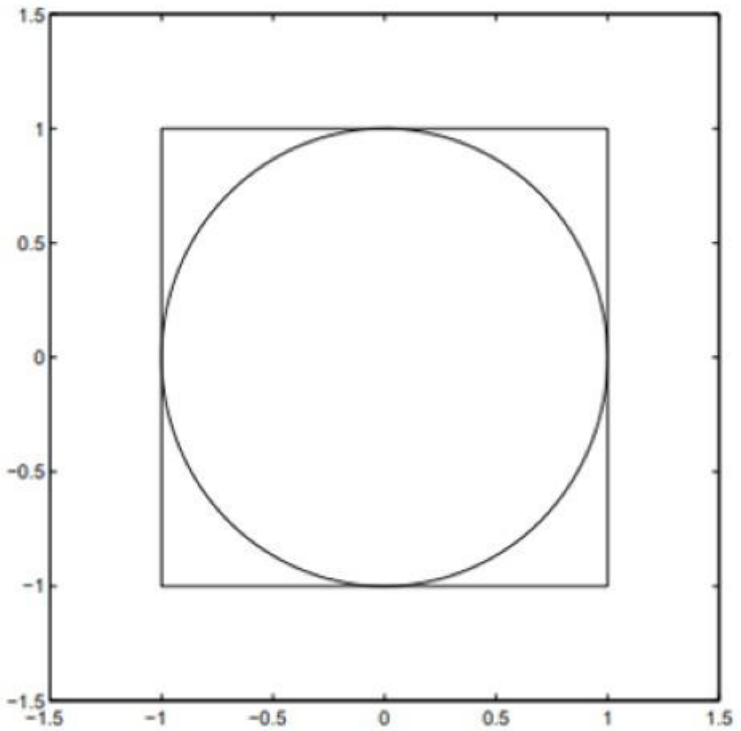
Principes clés pour le choix des graines

- 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 π) est inscrit dans ce carré, et son aire (π) est la cible de l'estimation.



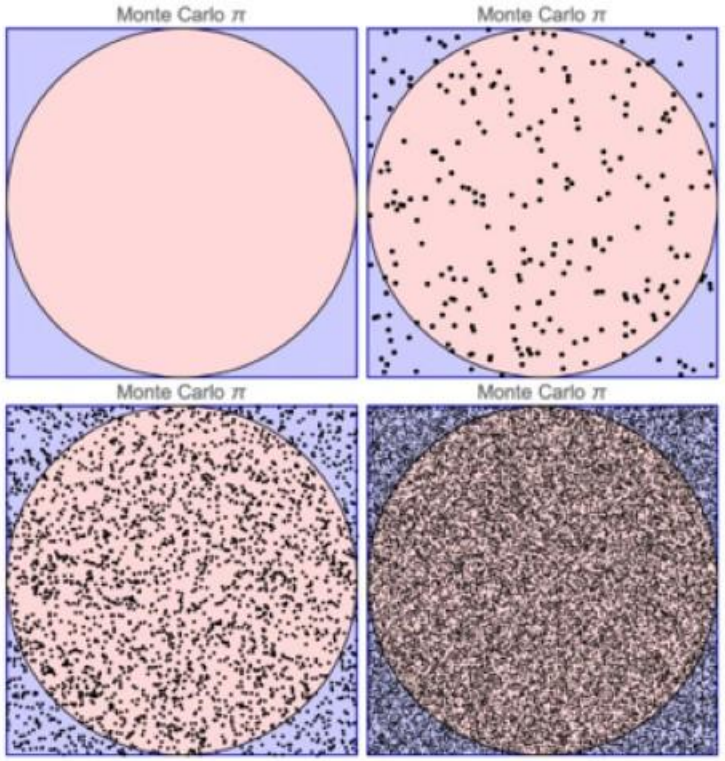


Figure 2: Squares with zero, 250, 2500 and 25000 points.

La proportion de points tombant dans le cercle par rapport au carré est $\pi/4$ (car l'aire du cercle est π, 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 π = 3.148000 (Erreur = 0.006408)
N = 10000: Estimation de π = 3.106800 (Erreur = 0.034792)
N = 100000: Estimation de π = 3.133720 (Erreur = 0.007872)

Résultats avec random.random :
N = 1000: Estimation de π = 3.172000 (Erreur = 0.030408)
N = 10000: Estimation de π = 3.122400 (Erreur = 0.019192)
N = 100000: Estimation de π = 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.

Test des Générateurs de Nombres Aléatoires (RNG)

- **Pourquoi tester les RNG ?**

  - Le test **n'est pas nécessaire** si :
    - On utilise un **package bien connu**
    - Un générateur **déjà validé**.
  - Échouer un test → le générateur est mauvais
  - Réussir un test → ne prouve pas qu'il est bon.

- **Deux catégories de tests :**
  - **Test d'uniformité** $H_0 : R_i \sim U[0,1]$ $H_1 : R_i \not\sim U[0,1]$ → **Ne pas rejeter** H_0 **= aucune preuve de non-uniformité**
  - **Test d'indépendance** H_0 : R_i indépendants H_1 : R_i non indépendants → **Ne pas rejeter** H_0 **= aucune preuve de dépendance**

- **Niveau de signification** α → Probabilité de **rejeter** H_0 alors qu'elle est **vraie** $\alpha = P(\text{rejeter } H_0 \mid H_0 \text{ vraie})$ → Exemple : $\alpha = 0.05$ → **5 % de risque d'erreur**

Test des Générateurs de Nombres Aléatoires (RNG)

- Deux tests sont couramment utilisés pour évaluer l'aléatoire :
  - Chi-Square Test
  - Kolmogorov-Smirnov Test (KS test)

Test des Générateurs de Nombres Aléatoires (RNG)

❑ Chi-Square Test

- **Objectif :** Vérifier si la répartition des nombres aléatoires correspond à une distribution théorique donnée (généralement uniforme).

- **Hypothèses :**

  - **H0 (hypothèse nulle) :** Les nombres aléatoires suivent la loi uniforme U[0,1]
  - **H1 (hypothèse alternative) :** Les nombres aléatoires ne suivent pas la loi uniforme.

Test des Générateurs de Nombres Aléatoires (RNG)

Chi-Square Test

Étapes du test du Chi²

1. **Diviser l'intervalle [0, 1] en k sous-intervalles** *(ex. : 10 intervalles de largeur 0,1)*
2. **Compter les fréquences observées (O_i)** → Nombre de valeurs aléatoires dans chaque intervalle
3. **Calculer les fréquences attendues (E_i)** → Sous hypothèse d'uniformité
4. **Calculer la statistique du Chi² :**

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

5. **Comparer la statistique calculée avec la valeur critique** → Table du Chi², avec $k - 1$ **degrés de liberté** et $\alpha = 0,05$

Test des Générateurs de Nombres Aléatoires (RNG)

❑ Chi-Square Test

Exemple :

Supposons que vous générez 1 000 nombres aléatoires et que vous souhaitez tester leur répartition.Pour cela, vous divisez l'intervalle [0, 1] en 5 sous-intervalles égaux et comptez combien de nombres tombent dans chaque intervalle.

- **Rejeter l'hypothèse nulle (H0)** : → Si la **statistique Chi² calculée** est **supérieure** à la **valeur critique** au seuil α choisi.

- **Accepter l'hypothèse nulle (H0)** *(ou plus précisément : ne pas rejeter)* : → Si la **statistique Chi² calculée** est **inférieure ou égale** à la **valeur critique** au seuil α choisi.

Test des Générateurs de Nombres Aléatoires (RNG)

- Exemple : Chi-Square reference table

Degrees of Freedom (df)	$\alpha = 0.10$	$\alpha = 0.05$	$\alpha = 0.01$	$\alpha = 0.001$
1	2.706	3.841	6.635	10.828
2	4.605	5.991	9.210	13.815
3	6.251	7.815	11.345	16.266
4	7.779	9.488	13.277	18.467
5	9.236	11.070	15.086	20.515
6	10.645	12.592	16.812	22.457
7	12.017	14.067	18.475	24.322
8	13.361	15.507	20.090	26.124
9	14.684	16.919	21.666	27.877
10	16.919	18.307	23.209	29.588

Test des Générateurs de Nombres Aléatoires (RNG)

Kolmogorov-Smirnov (KS) test

- Compare la fonction de répartition cumulée observée F0(x) avec la fonction de répartition cumulée attendue Fe(x) → pour voir à quel point les données correspondent à la distribution théorique.

- Le test KS évalue si la différence entre les deux CDF est suffisamment petite → formalisation du concept derrière le graphique Quantile-Quantile (Q-Q plot).

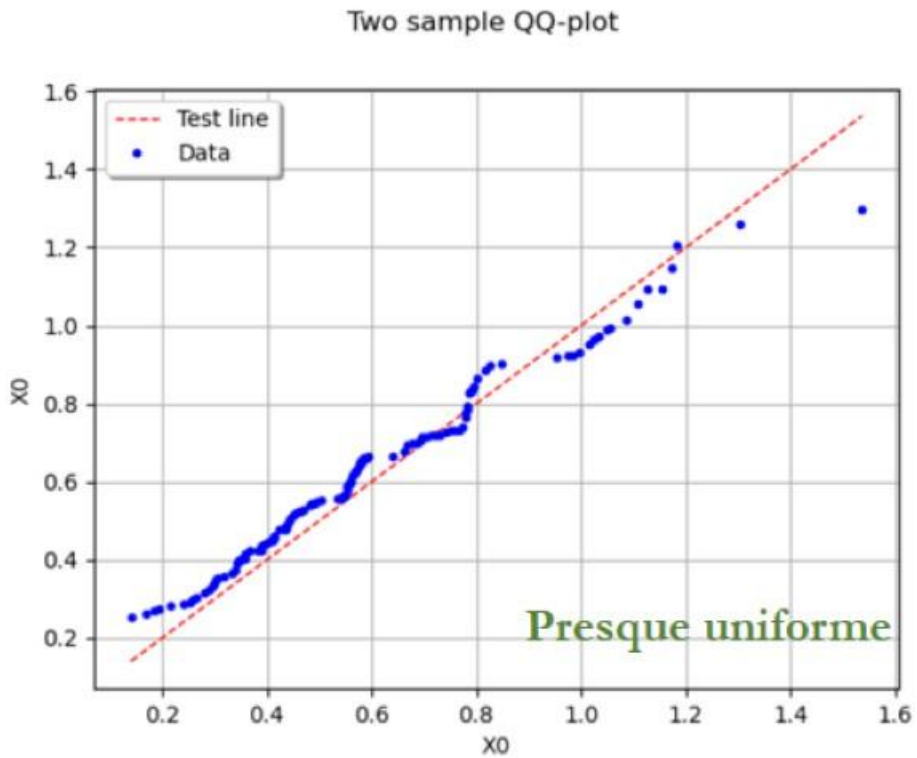
Q-Q Plot = "Compare les points de tes données triées avec les points attendus si c'était uniforme"

Test des Générateurs de Nombres Aléatoires (RNG)

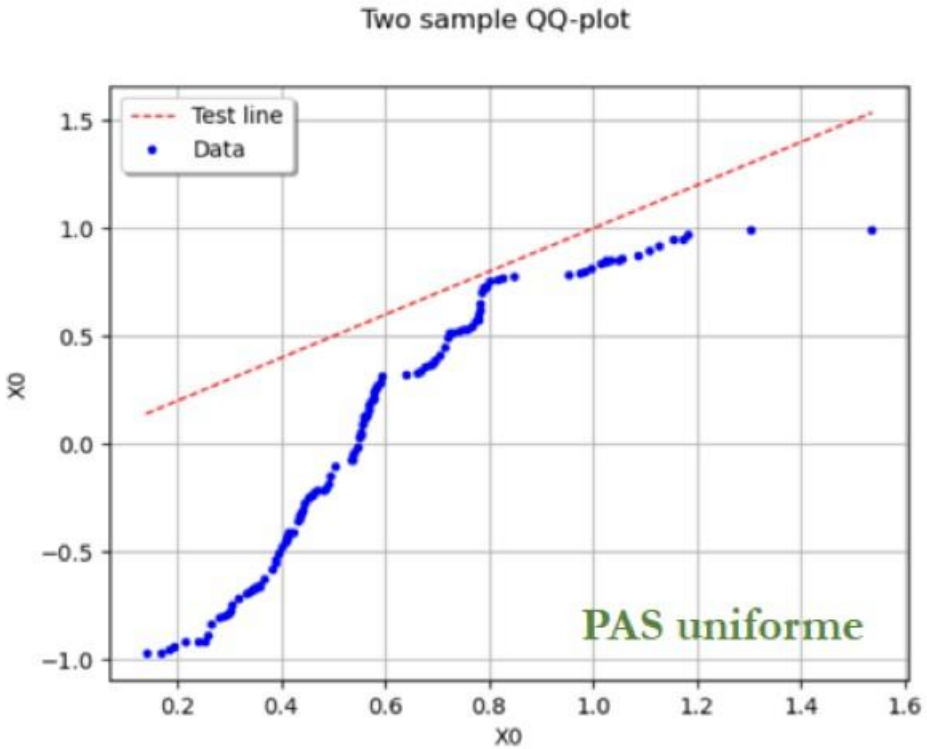
Le test KS = Q-Q plot + mesure de l'écart max (D_max)
→ Q-Q plot = visualisation
→ KS = décision automatique

Générer 3 échantillons : sample1 et sample2 proviennent de la même distribution.

Comparer visuellement sample1 et sample2 à l'aide du Q-Q plot



Comparer visuellement sample1 et sample3 à l'aide du Q-Q plot



Test des Générateurs de Nombres Aléatoires (RNG)

Kolmogorov-Smirnov (KS) test

Étape 1 : Classer les observations

- **Trier les données du plus petit au plus grand :** $Y_1 \leq Y_2 \leq Y_3 \leq \cdots \leq Y_n$

Étape 2 : Définir la CDF observée $F_0(x)$

- $F_0(x)$ = **proportion des échantillons** Y_i qui sont **inférieurs ou égaux à** x, divisée par n (nombre total d'échantillons).

$$F_0(x) = \frac{\#(Y_i \leq x)}{n}$$

Étape 3 : Calculer la statistique KS

- La statistique K est **l'écart absolu maximum** entre :
  - la **CDF observée**
  - et la **CDF attendue** → **sur tous les** x.

$$K = \max |F_0(x) - F_e(x)|$$

Test des Générateurs de Nombres Aléatoires (RNG)

Kolmogorov-Smirnov (KS) test

n \ α	0.001	0.01	0.02	0.05	0.1	0.15	0.2
1		0.99500	0.99000	0.97500	0.95000	0.92500	0.90000
2	0.97764	0.92930	0.90000	0.84189	0.77639	0.72614	0.68377
3	0.92063	0.82900	0.78456	0.70760	0.63604	0.59582	0.56481
4	0.85046	0.73421	0.68887	0.62394	0.56522	0.52476	0.49265
5	0.78137	0.66855	0.62718	0.56327	0.50945	0.47439	0.44697
6	0.72479	0.61660	0.57741	0.51926	0.46799	0.43526	0.41035
7	0.67930	0.57580	0.53844	0.48343	0.43607	0.40497	0.38145
8	0.64098	0.54180	0.50654	0.45427	0.40962	0.38062	0.35828
9	0.60846	0.51330	0.47960	0.43001	0.38746	0.36006	0.33907
10	0.58042	0.48895	0.45662	0.40925	0.36866	0.34250	0.32257
11	0.55588	0.46770	0.43670	0.39122	0.35242	0.32734	0.30826
12	0.53422	0.44905	0.41918	0.37543	0.33815	0.31408	0.29573
13	0.51490	0.43246	0.40362	0.36143	0.32548	0.30233	0.28466
14	0.49753	0.41760	0.38970	0.34890	0.31417	0.29181	0.27477
15	0.48182	0.40420	0.37713	0.33760	0.30397	0.28233	0.26585
16	0.46750	0.39200	0.36571	0.32733	0.29471	0.27372	0.25774
17	0.45440	0.38085	0.35528	0.31796	0.28627	0.26587	0.25035

Test des Générateurs de Nombres Aléatoires (RNG)

- **Kolmogorov-Smirnov (KS) test**

  - **Z-Table :** https://www.ztable.net/ (Distribution normal)
  - **KS Table :** https://real-statistics.com/statistics-tables/kolmogorov-smirnov-table/

Test des Générateurs de Nombres Aléatoires (RNG)

Vs.

Test KS	**Test du Chi²**
• Petits échantillons	• Grands échantillons
• Distributions continues	• Distributions discrètes
• Différences entre probabilités cumulées observées et attendues	• Différences entre probabilités observées et hypothétisées
• Utilise chaque observation sans regroupement	• Regroupe les observations en un petit nombre de cases
• La taille des cases n'est pas un problème	• La taille des cases affecte la conclusion
• Exact	• Approximatif
	• Pas de règle stricte