Lab x001 - Génération pseudo aléatoire et simulation stochastique

Jairo Cugliari

M2 MALIA - UL2

Exo 1 – Générateur congruentiel linéaire

1. Coder un générateur congruentiel linéaire comme celui vu en cours. Vous écrirez une fonction prenant comme argument la graine, une période et les paramètres A et C du générateur.
2. Utiliser votre PRNG pour créer une suite de $n=1000$ valeurs entre 0 et 1, $u_1,...,u_n$ avec $X_0=1$, $a=43$, $b=0$ et $m=2048$.
3. Représenter graphiquement la suite sur le plan (u_n,n).
4. Obtenir un histogramme. Comment pourriez-vous identifier un mauvais générateur avec l'histogramme ?
5. Représenter graphiquement la suite sur le plan u_{n+1},u_n. Comment pourriez-vous identifier un mauvais générateur avec ce graphique ?
6. Que pouvez-vous conclure sur l'adéquation de cette suite aux propretés statistiques souhaités.

Pour aller plus loin : Répétez l'exercice en utilisant maintenant $X_0=1,a=1664525,b=1013904223$ et $m=2^{32}$. Laquelle de ces deux suites vous semble plus pertinente ?

Exo 2 – Simulation par la méthode de l'inverse de F

1. Rappeler la définition de densité de probabilité.
2. Prouver que la densité d'une v.a. exponentielle de paramètre $\lambda>0$ est une densité de probabilité.
3. Obtenir la fonction de répartition.
4. Obtenir l'inverse de la fonction de répartition.
5. Écrire une fonction qui simule des échantillons exponentiels à partir de la méthode d'inversion

Exo 3 – Simulation par la méthode du rejet

Vous devez simuler des échantillons à partir d'une distribution Beta avec des paramètres $(\alpha,\beta)=(2,5)$ en utilisant la méthode du rejet. La distribution de proposition $q(x)$ à utiliser sera la loi uniforme $\mathcal{U}(0,1)$.

1. Rappelez la fonction de densité de la loi Beta avec les paramètres donnés.
2. Proposez un facteur d'échelle M adapté pour la méthode du rejet.
3. Implémentez l'algorithme de la méthode du rejet en choisissant un nombre d'échantillons $n=1000$.
4. Tracez un histogramme des échantillons obtenus et comparez-le à la densité de la distribution cible.

Pour aller plus loin : Testez différentes valeurs de α et β pour observer l'impact sur l'efficacité de la méthode du rejet.

Exo 4 – Simulation de variables exponentielles par inversion et rejet

Vous utiliserez deux méthodes différentes pour simuler des échantillons d'une distribution exponentielle avec paramètre $\lambda=2$.

1. **Méthode d'inversion :** Rappelez la fonction de répartition de la loi exponentielle et dérivez sa fonction inverse. Utilisez cette méthode pour générer 1000 échantillons d'une distribution exponentielle.
2. **Méthode du rejet :** En utilisant une distribution de proposition uniforme $\mathcal{U}(0,1)$ et un facteur d'échelle approprié, implémentez la méthode du rejet pour générer des échantillons d'une distribution exponentielle.

3. Comparez les résultats obtenus avec les deux méthodes en traçant les histogrammes des échantillons. Discutez de l'efficacité des deux méthodes en termes de temps de calcul et de précision.

Pour aller plus loin : Modifiez le paramètre λ et observez son effet sur les deux méthodes de simulation.

Exo 5 – Comparaison de Générateurs Pseudo-Aléatoires (PRNG)

Vous allez comparer deux générateurs pseudo-aléatoires couramment utilisés, le générateur congruentiel linéaire (LCG) et le générateur Mersenne Twister.

1. Implémentez un **LCG** avec les paramètres suivants :
$$a = 1103515245, \quad c = 12345, \quad m = 2^{31}, \quad X_0 = 42.$$
Générez 10 000 nombres pseudo-aléatoires.
2. Utilisez un générateur **Mersenne Twister** (disponible dans les bibliothèques Python/NumPy ou autres) pour générer également 10 000 nombres pseudo-aléatoires.
3. Tracez deux histogrammes des nombres générés pour chaque générateur. Comparez visuellement la qualité des distributions.
4. Appliquez un test statistique (par exemple, le test χ^2 ou Kolmogorov-Smirnov) pour évaluer la qualité des deux PRNGs. Comparez les résultats.
5. Discutez de la différence de la période des deux générateurs et de leur impact potentiel dans les applications nécessitant des longues séquences.

Pour aller plus loin : Répétez l'expérience avec un autre générateur pseudo-aléatoire disponible (par exemple, un générateur cryptographique).

Exo 6 – PRNG dans les langages de programmation

Cherchez sur Google et avec ChatGPT comment générer des suites *iid* suivant une distribution $\mathcal{U}[0,1]$ avec deux langages informatiques de votre choix (`Python`, `R`, `Julia`, `C++`, `Matlab/Octave`). Apprenez à fixer la graine aléatoire. Quel algorithme de génération est utilisé par défaut ?