Université Mohamed BOUDIAF
M'Sila

Faculté des Mathématiques et de l'Informatique.

Département de Mathématiques, 3ème Année Licence
Mathématiques Appliquées Semestre 6

Module : Simulation et pratique de logiciels

Responsable du module :

Dr. BOUHADJAR El Mountasar Billah

Chapitre 1 : Nombres aléatoires et pseudo aléatoires

Année Universitaire : 2024-2025

Table des matières

Chapitre 1

Nombres aléatoires et pseudo aléatoires

1.1 Introduction

Dans le monde moderne, le hasard joue un rôle essentiel dans la résolution de problèmes allant des jeux de hasard simples aux simulations scientifiques complexes. Que ce soit pour prédire les tendances des marchés financiers, modéliser des phénomènes météorologiques ou chiffrer des données sensibles, la capacité à générer des nombres imprévisibles est une pierre angulaire de l'informatique et de l'analyse.

Cependant, la véritable aléatoire, issue de phénomènes naturels ou de processus physiques, reste difficile à exploiter de manière cohérente. Pour répondre à ce défi, des méthodes computationnelles ont été développées afin de produire des séquences qui imitent le hasard. Ces nombres pseudo-aléatoires, bien qu'engendrés par des algorithmes déterministes, possèdent des propriétés statistiques proches de celles des séquences véritablement aléatoires.

1.2 Explication : Générateurs de nombres aléatoires

Un générateur de nombres aléatoires (GNA) est une construction mathématique ou physique utilisée pour produire des nombres sans modèle séquentiel ni corrélation, apparaissant donc aléatoires. Les générateurs de nombres aléatoires se répartissent en deux grandes catégories :

— **Les générateurs de nombres véritablement aléatoires**, basés sur des phénomènes physiques ou naturels (par exemple, le lancer de dés, le bruit thermique ou la radioactivité).
— **Les générateurs de nombres pseudo-aléatoires (GNPA)**, qui reposent sur des algorithmes déterministes pour produire des séquences de nombres qui simulent l'aléatoire.

Historiquement, les méthodes pour obtenir des nombres aléatoires utilisaient des processus physiques, comme les jeux de hasard (dés, roues de loterie) ou les tirages au sort. Ces méthodes ont permis de construire des tables de nombres aléatoires utilisées dans des calculs et simulations. Cependant, les séquences générées de cette manière étaient limitées en taille et en variété, ce qui les rendait inadaptées aux applications modernes nécessitant de grandes quantités de nombres aléatoires.

Aujourd'hui, la plupart des besoins en nombres aléatoires sont satisfaits par les GNPA, qui génèrent des séquences de nombres grâce à des algorithmes mathématiques. Bien qu'ils soient déterministes, ces générateurs offrent des propriétés statistiques proches de celles des séquences aléatoires véritables, ce qui les rend idéaux pour les simulations, la cryptographie, et d'autres applications. La simplicité et l'efficacité des GNPA en font un outil indispensable en informatique moderne.

1.3 Génération des nombres pseudo-aléatoires

La génération des nombres pseudo-aléatoires est une étape essentielle dans de nombreuses applications, notamment les simulations, la cryptographie et les modèles probabilistes. Contrairement aux nombres véritablement aléatoires, obtenus à partir de phénomènes physiques comme le bruit thermique ou quantique, les nombres pseudo-aléatoires sont générés par des algorithmes mathématiques.

Un générateur de nombres pseudo-aléatoires (GNPA) fonctionne de manière déterministe. Cela signifie qu'il produit une suite de nombres à partir d'une valeur initiale appelée *graine* (*seed*), en suivant une règle fixe ou un algorithme prédéfini. Bien que ces nombres soient produits de manière systématique, leurs propriétés statistiques les rendent semblables à des nombres véritablement aléa-

toires. Ces générateurs doivent répondre à deux critères importants :

— **Uniformité** : Les nombres générés doivent être répartis de manière uniforme sur l'intervalle $[0, 1]$ ou dans une plage spécifiée.

— **Apparente indépendance** : Les valeurs successives ne doivent pas montrer de corrélation perceptible, même si elles sont issues d'un processus déterministe.

Les GNPA sont largement utilisés en raison de leur rapidité, leur capacité à générer de grandes séquences, et leur reproductibilité, ce qui est utile pour des tests ou des simulations répétées. Cependant, ils ne sont pas adaptés aux applications nécessitant une véritable sécurité, comme la cryptographie, car leurs séquences peuvent être prédites si l'algorithme ou la graine est connu.

Algorithme général de récurrence

1. Choisir x_0 dans $M := \{0, 1, \ldots, m-1\}$.
2. Poser $x_{n+1} = f(x_n)$, pour $n > 0$ où f est une application adéquate de M dans M.

1.3.1 La méthode de Fibonacci

Cette méthode est basée sur une suite additive bien connue :

$$X_n = (X_{n-1} + X_{n-2}) \mod m$$

Elle appartient à la classe des **générateurs à congruence additive**. Plus généralement, avec un ensemble initial de k nombres $X_0, X_1, \ldots, X_{k-1}$, on définit la suite pour $n \geq k$ par :

$$X_n = (X_{n-1} + X_{n-k}) \mod m$$

Avantages

— Méthode simple à mettre en œuvre.

— Peut générer des suites pseudo-aléatoires dans certains contextes.

Inconvénients

— **Mémoire requise :** Il est nécessaire de stocker les k dernières valeurs générées, ce qui peut devenir coûteux pour des k élevés.

— **Corrélations pour k faible :** Lorsque k est petit, les nombres consécutifs de la suite présentent une forte dépendance, ce qui diminue leur qualité pseudo-aléatoire.

1.3.2 Méthode congruentielle simple

La méthode congruentielle est une technique utilisée pour générer une suite de nombres pseudo-aléatoires. Cette méthode repose sur une formule mathématique simple et répétitive, qui produit des nombres qui semblent aléatoires, bien qu'ils soient générés de manière entièrement déterministe.

La formule générale : La suite est définie par :

$$x_{n+1} = (a \cdot x_n + c) \mod m$$

Avec :

— x_0 : la graine (le premier nombre choisi pour commencer la suite).
— a : le multiplicateur.
— c : l'incrément.
— m : le module, un nombre entier positif qui détermine la plage des résultats.

Si $c > 0$ le générateur est dit mixte, sinon ($c = 0$) on parle de générateur multiplicatif.

Fonctionnement :

1. On commence avec un nombre initial x_0, appelé la graine.
2. En utilisant la formule ci-dessus, on calcule x_1, puis x_2, et ainsi de suite.
3. Les valeurs x_n générées sont comprises entre 0 et $m - 1$.

Exemple 1.3.1. *Si on choisit $x_0 = 5$, $a = 3$, $c = 7$, et $m = 10$, la suite se calcule comme suit :*

$$x_1 = (3 \cdot 5 + 7) \mod 10 = 22 \mod 10 = 2$$

$$x_2 = (3 \cdot 2 + 7) \mod 10 = 13 \mod 10 = 3$$

$$x_3 = (3 \cdot 3 + 7) \mod 10 = 16 \mod 10 = 6$$

Ainsi, la suite obtenue est : $5, 2, 3, 6, \ldots$

Propriétés importantes :

1. Uniformité : En divisant chaque x_n par m, on obtient une suite de nombres pseudo-aléatoires entre 0 et 1 .
2. Période "p" : le nombre de valeurs possibles de la suite.
3. Si p = m, on parle de générateur à période maximale.

Exemple 1.3.2. *Soit*

$$a = 4,\ c = 1,\ m = 9, \text{ et la graine initiale } x_0 = 3.$$

Calculons les étapes de génération :

$$\begin{aligned}
x_1 &= (4 \cdot 3 + 1) \mod 9 = 13 \mod 9 = 4,\\
x_2 &= (4 \cdot 4 + 1) \mod 9 = 17 \mod 9 = 8,\\
x_3 &= (4 \cdot 8 + 1) \mod 9 = 33 \mod 9 = 6,\\
x_4 &= (4 \cdot 6 + 1) \mod 9 = 25 \mod 9 = 7,\\
x_5 &= (4 \cdot 7 + 1) \mod 9 = 29 \mod 9 = 2,\\
x_6 &= (4 \cdot 2 + 1) \mod 9 = 9 \mod 9 = 0,\\
x_7 &= (4 \cdot 0 + 1) \mod 9 = 1,\\
x_8 &= (4 \cdot 1 + 1) \mod 9 = 5,\\
x_9 &= (4 \cdot 5 + 1) \mod 9 = 21 \mod 9 = 3.
\end{aligned}$$

La séquence générée est :

$$\{3, 4, 8, 6, 7, 2, 0, 1, 5, 3\}.$$

On remarque que la séquence recommence à partir de $x_9 = 3$, ce qui signifie que la période est 9. Ainsi, cette méthode produit une suite pseudo-aléatoire avec une période maximale de $M = 9$.

Remarque 1. *La méthode congruentielle est simple à mettre en œuvre et rapide, mais elle a des limites. Par exemple, si les paramètres a, c, et m ne sont pas choisis avec soin, la séquence peut être biaisée ou trop courte.*

Théorème 1.3.3. *(Hull-Dobell) : Un générateur congruentiel linéaire possède une période maximale (p = m) si, et seulement si, les trois conditions suivantes sont vérifiées :*

1) Les entiers c et m sont premiers entre eux.
2) Pour tout facteur premier $\hat{p}$ de m, $a - 1$ est un multiple de $\hat{p}$.
3) Si m est un multiple de 4, alors $a - 1$ est également un multiple de 4.

Remarque 2. *La situation est plus compliquée pour les générateurs multiplicatifs* ($c = 0$) *parce qu'il reste toujours la possibilité de blocage (si* $x_0 = 0$*). Dans ce cas, la période maximale ne peut être au mieux que* $m - 1$.

Exemple 1.3.4. *Considérons le générateur congruentiel suivant :*

$$x_{n+1} = (31415821x_n + 1) \bmod 100000000$$

Étapes de vérification

1. ***Condition 1 :*** *Ici,* $c = 1$ *et* $m = 100000000$*. Le plus grand commun diviseur est :*
$$PGCD(1, 100000000) = 1$$
2. ***Condition 2 :*** *Les facteurs premiers de* $m = 100000000$ *sont* 2 *et* 5*, car :*
$$m = 10^8 = 2^8 \cdot 5^8$$
Calculons $a - 1$ *:*
$$a - 1 = 31415821 - 1 = 31415820$$
C'est évident que a-1 est bien un multiple de 2 et 5.
3. ***Condition 3 :*** *Ici,* $m = 100000000$ *est un multiple de* 4*. Aussi* $a - 1$ *est un multiple de 4.*

Toutes les conditions du théorème de Hull-Dobell sont vérifiées. Par conséquent, le générateur congruentiel a une période maximale de $p = 100000000$.

Exposé : Chercher et bien expliquer les algorithmes suivants :

1. L'algorithme de Mersenne Twister (générateur de nombres pseudo-aléatoires).
2. L'algorithme de Berlekamp-Massey (pour les registres à décalage à rétroaction linéaire).
3. L'algorithme de Reed-Solomon (pour la correction d'erreurs).
4. L'algorithme RSA (cryptographie à clé publique).

Instructions :

Fournir une explication détaillée de chaque algorithme, y compris son fonctionnement, ses applications et son importance.

Inclure des exemples ou des pseudocodes si possible.

Préparer un exposé écrit (5-10 pages) et une présentation orale (10-15 minutes).

1.4 Tests des générateurs des nombres pseudo aléatoires

Les tests de conformité ont pour objectif d'évaluer si un échantillon de données est issu ou non d'une variable aléatoire suivant une loi de distribution théorique donnée $\mathbb{F}_0(x)$. Supposons que $\mathbb{F}(x)$ soit la fonction de répartition empirique basée sur l'échantillon. Le but est de vérifier l'hypothèse nulle :

$$\mathbb{H}_0 : \mathbb{F}(x) = \mathbb{F}_0(x)$$

contre l'hypothèse alternative :

$$\mathbb{H}_1 : \mathbb{F}(x) \neq \mathbb{F}_0(x).$$

Ces hypothèses permettent de déterminer si les données sont compatibles avec la distribution théorique proposée.

1.4.1 Test du Chi-deux χ^2

On cherche à tester l'uniformité d'une suite générée $\{u_1, u_2, \ldots, u_n\}$ à l'intervalle $[0, 1]$ où chaque $u_i = \frac{x_i}{m}$. Voici les étapes principales de ce test :

1. **Partitionnement de l'intervalle** $[0, 1]$ **:** On divise l'intervalle $[0, 1]$ en r sous-intervalles $I_1, I_2, \ldots, I_r$, où chaque sous-intervalle est défini par :

   $$I_i = [c_{i-1}, c_i], \quad c_0 = 0, \quad c_r = 1.$$

   En général, on choisit $r \approx \sqrt{n}$ pour une meilleure répartition.
2. **Effectifs observés :** Soit n_i l'effectif de la classe I_i, défini comme suit :

   $$n_i = \text{card}\{k : u_k \in I_i,\, k = 1, \ldots, n\}, \quad \forall i = 1, \ldots, r.$$

3. **Probabilités :** La probabilité est donnée par :

   $$p_i = \mathbb{F}(c_i) - \mathbb{F}(c_{i-1}),$$

   où $\mathbb{F}(x)$ est la fonction de répartition associée à la loi uniforme, donc $\mathbb{F}(x) = x\mathbf{1}_{[0,1[}(x) + \mathbf{1}_{[1,+\infty)}(x)$. Cela implique que $p_i = c_i - c_{i-1}$.

4. **Statistique de test de Pearson :** La statistique de test est donnée par :
$$k_n^2 = \sum_{i=1}^{r} \frac{(n_i - np_i)^2}{np_i},$$
où np_i représente l'effectif théorique attendu pour la classe I_i.
5. **Distribution asymptotique :** Pearson a démontré que la variable aléatoire k_n^2 suit asymptotiquement une loi du Chi-deux avec $(r-1)$ degrés de liberté sous l'hypothèse nulle $\mathbb{H}_0$ (la suite est uniforme).
6. **Règle de décision :**
   — Si $k_n^2 < \chi^2_{(r-1,\alpha)}$, on accepte l'ajustement, c'est-à-dire $\mathbb{H}_0$.
   — Si $k_n^2 \geq \chi^2_{(r-1,\alpha)}$, on rejette l'ajustement.

Remarque 3. *(Cas des intervalles équiprobables) Pour tester l'uniformité sur* $[0,1]$ *avec des intervalles équiprobables, les probabilités* p_i *sont toutes égales à* $\frac{1}{r}$ *c-à-d :* $p_1 = p_2 = ... = p_r = \frac{1}{r}$*. Les intervalles sont alors définis par :*
$$I_i = \left[\frac{i-1}{r}, \frac{i}{r}\right[, \quad \forall i = 1, \ldots, r.$$

Exemple 1.4.1. *1000 nombres aléatoires ont été échantillonnés à l'aide d'un générateur de nombres pseudo-aléatoires. Nous testons si ces nombres sont uniformément distribués sur* $[0,1]$*. L'intervalle est divisé en* $r = 10$ *sous-intervalles de largeur* $h = 0.1$*. Les fréquences observées* (n_i) *et les fréquences théoriques* (np_i) *sont données dans le tableau suivant :*

Intervalle	$[0,h)$	$[h,2h)$	$[2h,3h)$	$[3h,4h)$	$[4h,5h)$	$[5h,6h)$	$[6h,7h)$	$[7h,8h)$	$[8h,9h)$	$[9h,1)$
n_i	*99*	*94*	*95*	*108*	*108*	*88*	*111*	*92*	*111*	*94*
np_i	*100*	*100*	*100*	*100*	*100*	*100*	*100*	*100*	*100*	*100*

En effectuant les calculs, on trouve :
$$k_n^2 = \sum_{i=1}^{r} \frac{(n_i - np_i)^2}{np_i} = \frac{676}{100} = 6.76.$$

Le degré de liberté est $r - 1 = 9$*. La valeur critique de* χ^2 *au seuil* $\alpha = 0.05$ *et pour 9 degré de liberté est :*
$$\chi^2_{9,0.05} = 16.92.$$

La règle de décision est la suivante :

— *Si* $k_n^2 < \chi^2_{9,0.05}$*, on accepte l'hypothèse nulle* $\mathbb{H}_0$ *: la suite est uniformément distribuée sur* $[0, 1]$.
— *Sinon, on rejette* $\mathbb{H}_0$.

Dans ce cas, $k_n^2 = 6.76 < 16.92$*. Par conséquent, nous acceptons* $\mathbb{H}_0$*, ce qui signifie que la série* $\{u_1, \ldots, u_n\}$ *est uniformément distribuée sur* $[0, 1]$.

1.4.2 Test de Kolmogorov-Smirnov

Le test de Kolmogorov-Smirnov est non paramétrique, ce qui signifie qu'il peut être utilisé sans supposer une forme précise de la distribution des données. Voici les étapes principales pour effectuer ce test :

1. Un échantillon de n observations est généré à l'aide du générateur de nombres aléatoires ;
2. Les valeurs de l'échantillon sont triées dans un ordre croissant ;
3. La fonction de répartition empirique $\mathbb{F}_n(x)$, construite à partir des n observations, est comparée à la fonction de répartition théorique $\mathbb{F}(x)$. Le test repose sur le calcul de l'écart maximal entre les deux fonctions, défini comme :

$$D_n = \max_x |\mathbb{F}_n(x) - \mathbb{F}(x)|,$$

où :

$$\mathbb{F}_n(x) = \frac{\text{nombre d'observations}}{n}.$$

Un seuil de signification α est fixé, et la valeur critique $d(\alpha)$ est déterminée à partir des tables de Kolmogorov-Smirnov. Ce seuil est choisi pour que $P(D_n > d(\alpha)) = \alpha$, où α représente le risque d'erreur.

Règle de décision

La décision est prise selon les critères suivants :
— Si $D_n > d(\alpha)$, on rejette l'hypothèse $\mathbb{H}_0$, c'est-à-dire que la distribution empirique $\mathbb{F}_n(x)$ est significativement différente de la distribution théorique $\mathbb{F}(x)$.
— Sinon, $\mathbb{H}_0$ est accepté, ce qui indique que les données sont compatibles avec la distribution théorique.