# ReadMe - Générateurs

March 2024

## 1 LCG - Générateur congruentiel linéaire

Un générateur congruentiel linéaire (LCG) est un algorithme qui génère une séquence de nombres pseudo-aléatoires en utilisant une relation linéaire. La formule générale d'un LCG est:

$$X_{n+1} = (a \cdot X_n + c) \mod m \tag{1}$$

où :

- $X$ est la séquence des nombres pseudo-aléatoires,
- $a$ est le multiplicateur,
- $c$ est l'incrément,
- $m$ est le module,
- $X_0$ est la valeur initiale (graine).

### 1.1 Implémentation Python

Ce code définit une fonction `lcg` qui prend les mêmes paramètres ($a$, $c$, $m$, `seed`) et retourne un générateur Python. On peut ensuite utiliser `next(random_generator)` pour obtenir les nombres pseudo-aléatoires un par un. Cette approche permet de continuer à générer des nombres pseudo-aléatoires de manière séquentielle sans avoir besoin de la structure de classe.

Les valeurs de $a$, $c$, et $m$ sont choisies pour cet exemple mais peuvent être ajustées pour répondre à différents besoins ou propriétés statistiques.

## 2 Générateur basé sur une application d'algèbre linéaire sur $F_2^n$

Créer un générateur de nombres pseudo-aléatoires basé sur l'algèbre linéaire sur $F_2^n$ (l'espace vectoriel des séquences binaires de longueur $n$ sur le corps fini $F_2$) est une tâche intéressante qui s'appuie sur des concepts mathématiques avancés.

Un exemple courant de tel générateur est le registre à décalage à rétroaction linéaire (LFSR, Linear Feedback Shift Register), très utilisé dans les applications cryptographiques et pour la génération de séquences pseudo-aléatoires.

Un LFSR manipule une séquence de bits selon une règle fixe et produit une séquence de sortie qui peut sembler aléatoire. Pour un LFSR donné, la règle est définie par un polynôme de rétroaction, où les termes du polynôme indiquent quels bits de l'état actuel sont utilisés pour calculer le bit de feedback.

### 2.1 Concepts clés

- **Seed** : L'état initial du LFSR. Il est important que cet état ne soit pas nul.
- **Taps** : Définit les positions des bits (en commençant à 0) qui sont utilisés pour calculer le bit de feedback. Ces positions correspondent aux termes non nuls du polynôme de rétroaction, à l'exception du terme de plus haut degré qui est toujours présent.

Le bit de feedback est calculé comme le XOR des bits sélectionnés par les taps. Ensuite, l'état du registre est décalé d'un bit vers la droite, et le bit de feedback est inséré à gauche. Cette procédure produit une séquence de sortie qui peut être utilisée comme générateur de nombres pseudo-aléatoires.

## 3 Générateur de Matrice sur $F_2^n$

Il est possible de créer un générateur de nombres pseudo-aléatoires basé sur l'algèbre linéaire sur $F_2^n$ sans utiliser directement un LFSR. Une méthode alternative, souvent appelée Générateur de Matrice, implique l'utilisation de matrices et de vecteurs dans $F_2^n$. Ce type de générateur utilise les opérations d'algèbre linéaire pour transformer un état initial (vecteur) par multiplication avec une matrice pour produire l'état suivant, générant ainsi une séquence de vecteurs (ou de nombres pseudo-aléatoires).

### 3.1 Principe du Générateur de Matrice

L'idée de base est de choisir une matrice carrée $A$ de taille $n \times n$ sur $F_2$ (c'est-à-dire, une matrice dont les éléments sont 0 ou 1, et toutes les opérations sont effectuées modulo 2) qui possède certaines propriétés, comme être non singulière (pour éviter des cycles courts ou des états nuls). On commence ensuite avec un vecteur initial $x_0$ de taille $n$ sur $F_2$, et on génère la séquence par récurrence :

$$x_{k+1} = A \cdot x_k \mod 2$$

où $x_k$ est l'état courant du générateur.

Dans cet exemple, $A$ est une matrice de transformation qui définit comment chaque nouvel état est calculé à partir de l'état actuel. Le seed est l'état initial

du générateur. Cette méthode produit une séquence de vecteurs qui peuvent être interprétés comme des nombres pseudo-aléatoires.

Il est crucial de choisir la matrice $A$ soigneusement pour assurer de bonnes propriétés pseudo-aléatoires de la séquence générée. Une matrice avec une grande période et qui distribue uniformément les états possibles dans $F_2^n$ est idéale. La complexité de choisir une telle matrice dépend des applications spécifiques et des exigences en matière de sécurité et d'efficacité.

# 4 LFSR - Registre à décalage à rétroaction linéaire

Le LFSR (Linear Feedback Shift Register) est un outil puissant pour générer des séquences de nombres pseudo-aléatoires. Son fonctionnement repose sur quelques principes simples mais efficaces :

1. On commence avec un état initial (la "graine") qui est un nombre binaire de longueur $n$.
2. On choisit certains bits dans cet état (définis par les "taps") pour calculer un nouveau bit par une opération XOR.
3. L'état est décalé d'un bit vers la gauche, et le nouveau bit calculé est inséré à la position la plus basse (droite).

Dans cette configuration :

- **Seed** est la valeur initiale du LFSR. Il est crucial que cette valeur ne soit pas nulle pour éviter une séquence de sortie trivial.
- **Taps** sont les positions des bits utilisés pour calculer le bit de feedback. Ces positions sont définies par rapport au bit le plus à gauche, en commençant par 0.
- **Steps** détermine combien de bits seront générés par le LFSR, dictant ainsi la longueur de la séquence de sortie.

La fonctionnalité clé du LFSR réside dans son utilisation des taps pour calculer le bit de feedback via une opération XOR, produisant ainsi une séquence qui peut sembler aléatoire mais qui est en réalité déterminée par sa graine initiale et sa configuration de taps. Cette méthode est largement utilisée dans divers domaines, notamment en cryptographie et dans la simulation numérique, en raison de sa capacité à générer rapidement de grandes séquences de nombres pseudo-aléatoires avec une bonne propriété de distribution.

# 5 Générateur Geffe

Le générateur Geffe est une forme améliorée de générateur de nombres pseudo-aléatoires qui combine trois registres à décalage à rétroaction linéaire (LFSR) de

manière non linéaire pour produire une sortie plus complexe et moins prévisible. L'idée principale est d'utiliser une fonction de filtrage non linéaire, typiquement une combinaison des sorties des trois LFSRs, pour générer la séquence finale.

La fonction de filtrage la plus simple pour le générateur Geffe peut être décrite comme suit, où $L_1$, $L_2$, et $L_3$ sont les sorties des trois LFSRs respectifs :

$$G(x) = L_1 \cdot L_2 \oplus (\neg L_1) \cdot L_3$$

Dans cet exemple, chaque LFSR est initialisé avec ses propres semences (seed) et configurations (taps). La longueur de la séquence générée est déterminée par le paramètre *length*. La fonction *geffe_generator* génère la séquence finale en combinant les sorties des trois LFSRs selon la fonction de filtrage de Geffe.

Il est crucial de choisir soigneusement les paramètres des LFSRs pour assurer de bonnes propriétés pseudo-aléatoires de la séquence générée par le générateur Geffe. La combinaison non linéaire des sorties augmente la complexité et la période de la séquence, rendant le générateur Geffe plus résistant à certaines formes d'analyse et d'attaque par rapport à un LFSR standard.