Générateur de Nombres Aléatoires

Les générateurs de nombres aléatoires sont des dispositifs capables de produire une séquence de nombres dont on ne peut pas tirer facilement des propriétés déterministes.

Il existe depuis très longtemps des méthodes pour obtenir de tels nombres, notamment utilisées dans les jeux de hasard, comme par exemple le lancement de dés ou le jeu de la roulette.

Néanmoins, de nombreux progrès ont été réalisés et actuellement les meilleurs moyens de générer des suites de nombres aléatoires sont d'utiliser des méthodes physiques qui profitent du caractère apparemment aléatoire des phénomènes quantiques.

Il sera alors intéressant d'expliciter la notion de générateur aléatoire et d'en donner des exemples d'utilisation dans un premier temps, puis d'aborder plus en profondeur le problème des générateurs pseudo aléatoires, et finalement des comparer ces derniers générateurs et d'en tirer des conclusions quand à leur fiabilité.

I/ Définition d'un générateur de nombres aléatoires et utilisation

Pour donner une approche assez simple des générateurs, on pourrait dire qu'ils ont pour but de caractériser le hasard.

Mais le caractère aléatoire est une notion difficile à appréhender et il se pose alors la question de savoir comment le définir.

Il faut alors choisir un phénomène (d'ordre physique, mathématique ou bien d'autres...) en profitant du fait que certains paraissent aléatoires de par leur nature.

Malheureusement cette nature aléatoire n'est qu'une apparence. Par exemple, le lancement d'un dé (pour lequel on dirait que chaque résultat a une chance sur 6 d'apparaître) est en fait soumis à de nombreuses contraintes physiques telle que la gravitation.

En fait, nous avons plus affaire à des caractères imprévisible qu'aléatoires ; de plus le caractère non aléatoire des conditions initiales (gravitation...) fait que le phénomène présente un biais, c'est-à-dire qu'il y a possibilité de prévision du résultat.

Nous allons alors voir que l'on peut former différents types de générateurs de nombres aléatoires :

- les générateurs que l'ont peut qualifier de « manuels » comme le pile ou face, les dés, etc.
- les générateurs reposants sur des algorithmes, ce qui semble contradictoire avec ce que l'on veut obtenir, du fait du caractère déterministe d'un algorithme ; pourtant certaines opérations sont suffisamment imprévisibles pour avoir une impression de hasard.
- les générateurs reposant sur des phénomènes physiques, (qui sont comme nous l'avons déjà dit les plus performants) comme la radioactivité, les bruits thermiques ou électromagnétiques, la mécanique quantique (choix d'un électron de traverser ou non une lame semi-réfléchissante).
- les générateurs mixtes utilisant à la fois une source physique et des algorithmes pseudo-aléatoires, comme le système des lampes à lave développé en 1996 par le cryptologue Landon Noll.

Les générateurs de nombres aléatoires sont utilisés dans des domaines nombreux et variés comme dans les jeux de hasard, la simulation (physique, biologique), l'analyse (calcul de valeur numérique d'équa diff), la sécurité informatique, la cryptologie et même la parapsychologie.

II/Générateur de nombres pseudo-aléatoires

Il est en fait impossible de créer des générateurs complètement aléatoires reposant sur des algorithmes, et c'est pourquoi ces générateurs sont appelés pseudo aléatoires car ils produisent des séquences de nombres presque indépendants.

De plus il est parfois possible d'utiliser des nombres pseudo aléatoires à la place de vrais nombres aléatoires car ils sont plus faciles à implémenter en informatique et plus facilement utilisables.

Il existe plusieurs types de générateurs de nombres pseudo aléatoires, tels que la méthode de Von Neumann ($1111^2=1234321$, $3432^2=11778624$, $7786^2=...$), la méthode de Fibonacci ($xn=(xn-1 + xn-2) \bmod M$) , mais nous allons nous intéresser à la méthode de congruence linéaire qui est la plus courante. Le but est alors d'en écrire le code dans le langage de Scilab qui est particulièrement bien approprié.

Son algorithme est plutôt simple :

$$Xn+1 = (a \cdot Xn + c) \bmod m$$

Mais les caractéristique de ce générateurs varient en fonctions des paramètres a, c, m et X0 qui est la graine du générateur, qui est très importante (en effet, pour une même graine, les résultat seront identiques.

On va alors le programmer de trois manières différentes :

- méthode dite de RANDU où a=65539, c=0 et $m=2^{31}$
- générateur de Sedgewick où a=31415821, c=1 et $m=10^8$ (en particulier on prendra X0=0)
- standard minimal où a=16807, c=0 et $m=2^{31}-1$

Il est évident que le caractère aléatoire de la graine est très important dans l'obtention de résultats analysables, c'est pourquoi on utilisera le nombre de secondes dans la minute en cours grâce à la fonction getdate de Scilab.

On peut également programmer une autre congruence :

$Xn+1 = Xn(Xn + 1) \bmod 2^e$ avec $X0 \bmod 4 = 2$

Puis, grâce au générateur BBS (Blum Blum Shub), nous verrons à quel point le choix des conditions initiales influe sur la période d'apparitions des nombres :

$Xn+1 = (Xn)^2 \bmod M$ avec $M=pq$ tel que p et q soient deux grands nombres premiers égaux à 3 modulo 4.

III/Comparaison des générateurs

Il est maintenant intéressant de comparer les générateurs pour discuter de leur fiabilité. Pour cela on prendra comme référence la fonction rand de Scilab.

On compare d'abord le temps d'exécution des générateurs pour un différents nombres de valeurs générées :

- Pour 1000 valeurs, les générateurs à congruences mettent environ tous 0,43 secondes alors que rand le fait presque instantanément (0,01s)
- Pour 10000 valeurs, les générateurs à congruences mettent environ 4,3 secondes, ce qui est beaucoup en comparaison au rand qui prend encore très peu de temps (0.05s)
- A partir de 100000 valeurs, les générateurs à congruence mettent vraiment beaucoup trop de temps (43 secondes : il y a une évolution proportionnelle entre le nombre de valeur et le temps d'exécution) alors que rand commence à prendre un peu plus de temps (0.1s). Néanmoins, la fonction rand est beaucoup plus rapide que les générateurs à congruence.

On compare ensuite les probabilités d'apparition des nombres en fonction du nombre de tirage (100,1000,10000) et on obtient les graphiques suivants :
(Voir feuilles)

On remarque donc que les courbes sont à peu près identiques quel que soit le nombre de tirages.

Conclusion

On en conclue donc que la proportion d'apparition des nombres avec les générateurs à congruences linéaires est très bonne pour une utilisation assez basique, puisque pour un nombre élevé de tirages, les résultats sont presque identiques à ceux de la fonction rand de Scilab ce qui en fait un générateur assez fiable du point de vue du caractère aléatoire recherché.

Néanmoins, on a vu que pour un nombre de tirages relativement grand, le temps d'exécution est beaucoup trop important, ce qui n'en fait pas une très bonne fonction en comparaison de la fonction rand qui reste la référence.