# Evaluation des générateurs pseudo-aléatoire

Dans cette partie on va s’intéresser en détail à la manière de tester un générateur pseudo-aléatoire. On rappelle que l’objectif d’un tel générateur est de générer des nombres aléatoirement qui sont uniformément réparties dans un intervalle donné et indépendant entre eux. C’est-à-dire qu’on ne peut pas prévoir le nombre suivant en connaissant les nombres générés précédent. Ces nombres étant générés par des algorithmes déterministes, ils ne pourront jamais égaler de vrais générateurs aléatoires. Néanmoins leurs performances peuvent être jugés satisfaisantes s’ils arrivent à passer un certain nombre de tests avec une difficulté qui dépendra de l’exigence demandée.

On va avoir deux catégories de tests :

* Les tests d’indépendances qui vont mesurer l’imprévisibilité du nombre généré suivant
* Les tests d’uniformités qui vont mesurer la qualité de la répartition de ces nombres

Il existe de nombreux tests d’uniformité ou d’indépendance. On présentera dans ce chapitre les tests les plus connus afin de comprendre leurs fonctionnements et comparer les différents algorithmes présentés au chapitre précédent. En réalité, pour valider un algorithme PRNG il ne suffit pas qu’il valide un seul test mais une batterie de test avec des critères plus élevés pour avoir une qualité cryptographique. On peut citer dans les plus connus les batteries de test TestU01 et Diehard et NIST pour une qualité cryptographique.

Nous verrons dans une dernière partie des algorithmes qui sont capables de prédire la suite de nombre aléatoire à partir des x valeurs précédentes comme Berlekamp-Massey ou Reed-sloans. Bien entendu les générateurs de qualité cryptographique doivent résister à ce genre d’attaque.

## A. Protocole de test

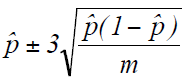
### 1. Les PRNG testés

### 2. Critère pour valider un test

Tous les tests statistiques sont basés sur le même principe, comparer le résultat observé à un résultat théorique. Chaque test calcule une valeur statistique qui ne doit pas dépasser un seuil critique pour valider les résultats observés qui varie en fonction de l’intervalle de confiance que l’on souhaite avoir. Par exemple si on veut un intervalle de confiance de 95%, on va récupérer la valeur correspond à 100 – 95 = 5% noté alpha qui sera notre seuil critique. Une façon équivalente de valider le test est de calculer la p-value qui se déduit directement de la valeur statistique du test. Si on choisit un alpha de 5%, alors le test est validé si p-value > 5%, en effet la p-value représente la probabilité d’obtenir les résultats observés sachant l’hypothèse théorique H0. Par contre si p-value < 5%, alors on peut rejeter l’hypothèse H0 et conclure que les résultats observés ne suivent pas les résultats théoriques.

Illustration avec un exemple concret : Si l’on cherche à déterminer l’efficacité d’un médicament sur un groupe de test, on va comparer les résultats par rapport à un groupe control qui n’aura pas reçu ce médicament qui sera les données attendues théorique. Si on choisit un seuil de tolérance de 5%, cela signifie qu’on a 5% de chance de rejeter l’hypothèse H0 avec un médicament sans effet. Rejeter l’hypothèse H0 implique soit qu’on a été très chanceux soit que le médicament à un effet significatif. On peut diminuer ce seuil de tolérance alpha pour augmenter la confiance dans le rejet de l’hypothèse H0 et donc que le médicament a un effet significatif. On peut remarquer qu’il y a toujours une possibilité de se tromper, et si on diminue fortement alpha pour s’assurer que le médicament a un effet significatif alors on risque de rater les médicaments ayants un effet moindre mais non négligeable. Pour pallier à ce problème on va voir comment effectuer plusieurs fois ce test aide à avoir une bien meilleure idée sur le résultat. Néanmoins dans un cas réel comme le test d’un médicament on n’a pas forcément la possibilité d’avoir autant de donnée que l’on souhaite. Ce qui n’est pas le cas pour la génération de nombre pseudo-aléatoire.

On va suivre le protocole établi pour les tests NIST. On choisit alpha de 1%, c’est-à-dire que le test se trompera 1% du temps si notre PRNG est bien uniforme et inversement réussira à 99%. On va reproduire le test 1000 fois avec de nouvelle donnée à chaque fois et estimer la proportion de test réussi. Le résultat qu’un test réussi ou non est binaire, ce qui permet de calculer l’intervalle de confiance pour la loi binomiale sachant que l’on connait la probabilité **p** de réussir le test et **m** le nombre de test effectué. Elle est donnée par :



Le coefficient 3 correspond à un intervalle de confiance de 99.7% (3 fois l’écart type pour une loi gaussienne). On notera que plus **m** est grand ou **p** petit, plus l’intervalle de confiance sera resserré et le résultat du test précis.

En choisissant alpha=1%, m=1000 et n=1000, on doit obtenir une proportion de test réussi supérieur à 98.06%.

## B. Les tests d’uniformités

### 1. Test du Chi-2

Le test du Chi-2 est très largement utilisé en statistique pour évaluer si une distribution d'échantillon suit une distribution théorique attendue. On peut l’utiliser pour montrer qu’on obtient bien la distribution attendue qui sera l’hypothèse H0 ou au contraire réfuter celle-ci avec un certain seuil de tolérance.

Dans notre cas, on veut montrer qu’un PRNG génère des nombres aléatoires uniformément. La distribution théorique attendue sera donc uniforme. On choisit de distribuer 1000 nombres aléatoires entre 0 et 1 dans 10 classes (0-0.1, 0.1-0.2, 0.2-0.3, …, 0.9-1.0), on s’attend à avoir 100 nombres pour chaque classe. On va ensuite calculer :

Avec Oi les effectifs observées, Ei les valeurs théoriques attendues pour chaque classe i

Puis comparer cette valeur à celle extraite de la table du Chi2 pour le seuil de tolérance choisi, ici 1% et un degré de liberté de 10 – 1 = 9.

En suivant le protocole de test on obtient pour nos trois PRNG :

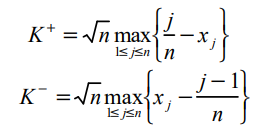
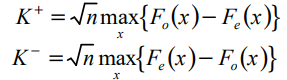
* Mersenne Twister : 0.991 > 0.9806
* LCG : 0.994 < 0.9806
* Blum Blum Shub : 0.991 > 0.9806

=> Nos trois PRNG ont passé le test

### 2. Test de Kolmogorov-Smirnov

Ce test permet de comparer la distribution de l’échantillon à une distribution théorique ou deux échantillons entre eux. Dans notre cas on va comparer notre échantillon à une distribution théorique uniforme. De même que pour le test du Chi2 et de façon plus générale, on calcule une statistique de test que l’on comparera par rapport à une table de valeur pour déterminer le seuil critique ou la p-value.

On va s’intéresser au test à un échantillon. L’objectif du test est de calculer la plus grande différence entre la distribution cumulative empirique et théorique noté respectivement Fo et Fe. On commence par trier les échantillons observés par ordre croissant qui seront noté Fo = xj avec j l’index de l’échantillon allant de 1 à n. On crée ensuite la série uniforme théorique allant de 1/n à n et allant de 0 à (n – 1) /n. On veut calculer K définit par :



On obtient comme résultats pour nos trois PRNG :

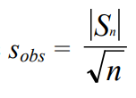
* Mersenne Twister : 0.992 > 0.9806
* LCG : 0.99 > 0.9806
* Blum Blum Shub : 0.992 > 0.9806

=> Nos trois PRNG ont passé le test

### 3. Test de fréquence monobit

L’objectif de ce test et de calculer la proportion de 0 et 1 en sortie d’un PRNG. Pour obtenir des bits au lieu de valeur continue entre 0 et 1, on ne divise plus la valeur de sortie par la période max du générateur mais on prend à la place le bit le plus faible. C’est-à-dire le plus le plus à droite, ex : 5 en binaire s’écrit 101, le bit le plus à droite et donc le plus faible est 1.

A partir de la série de taille n composée uniquement de 0 et 1, on va remplacer les 0 par -1 puis sommer la série entière et enfin diviser par la racine de n.



La p-value sera obtenue à partir de la fonction d’erreur complémentaire erfc. Avec un seuil de tolérance de 1%, on rejettera l’hypothèse d’une distribution uniforme si p-value < 0.01.

On obtient comme valeur de Q pour nos trois PRNG :

* Mersenne Twister : 0.992 > 0.9806
* LCG : 1.0 > 0.9806
* Blum Blum Shub : 0.985 > 0.9806

=> Nos trois PRNG ont passé le test

Néanmoins on peut remarquer que le LCG génère une série qui alterne toujours entre 0 et 1 c’est pourquoi on un score parfait de 1.0 pour le test mais qui n’est pas du tout aléatoire. Si l’on regarde les nombres générés, ils sont tous impairs. Les bits faible ne sont pas aléatoires, c’est pourquoi il est conseillé d’utiliser que les bits fort, par exemple les 16 premiers.

## C. Les tests d’indépendances

### 1. Test des paires en série

L’objectif du test est de former des paires de deux nombres aléatoires ex : x1, x2, x3 et x4 => (x1, x2), (x3, x4). Un peu à la manière du test d’uniformité du Chi2, on va découper un espace 2D en k\*k boites et on va compter le nombre de paire dans chacune d’entre elles. Ensuite on effectue le test du Chi2 en comparant l’observation à l’attendu. Le nombre de paire attendu est donnée par .

On obtient comme résultats pour nos trois PRNG :

* Mersenne Twister : 0.992 > 0.9806
* LCG : 0.989 > 0.9806
* Blum Blum Shub : 0.988 > 0.9806

=> Nos trois PRNG ont passé le test