

PAF Juin 2016

Rapport Synthétique de PAF

# Projet étudiant de 1ère année :

# « Peut-on être certain de l’incertitude ? »

**AZOULAY Nathan**

**BARAKAT Anas**

**COHEN-OLIVAR Adrien**

**Encadrant :** Lirida NAVINER et You WANG

Sommaire

[Objectifs 2](#_Toc454809650)

[Explications des Algorithmes 2](#_Toc454809651)

[Analyse des résultats 2](#_Toc454809652)

[Répartition du travail 2](#_Toc454809653)

# Objectifs

Les nombres aléatoires sont essentiels dans divers domaines. On peut citer par exemple la cryptographie où ils permettent de générer des mots de passes imprévisibles et donc sécurisés. Il est légitime de ce se demander comment savoir si un nombre binaire donnée est vraiment aléatoire.

L’objectif de ce projet est justement de mettre en œuvre un testeur de qualité à travers quinze algorithmes de nature statistique.

# Explications des Algorithmes

Frequency (Monobit) Test

Cette interprétation se fait grâce au théorème central limite. Si on note, Sn la somme des éléments de la séquence, où 0 est transformé en 1, on sait que Sn/n\*\*0.5 tend vers une v.a. suivant une loi normal d’espérance nulle et de variance 1.

P\_value étant la probabilité qu’un générateur de nombres aléatoires parfait pour ce test génère une séquence moins aléatoire que la séquence testée.

Frequency Test within a Block

Le test est dans la prolongation du précédent. Cette fois-ci la séquence est partagée en différent blocks où l’algorithme de Frequence test est appliqué sur chaque block. On applique ensuite un test chi-carre pour vérifier si les échantillons vérifient bien l’hypothèse nulle.

Binary Matrix Rank Test

Discrete Fourier Transform (Spectral) Test

Non-overlapping Template Matching Test

Le but de ce test est d’énumérer le nombre d’occurrence d’un motif donné dans la séquence. En effet, une séquence qui comporte un motif qui se répète plusieurs fois n’est certainement pas généré par un RNG.

Description : - Comme on fera un test chi\_carre, on partitionne la suite en N block de longueur M.

* Pour un motif B de longueur m donné, et pour un échantillon i donné on calcule Wi le nombre d’occurrence de B dans l’échantillon. Ce calcul se fait grâce à une fenêtre glissante qui a la particularité contrairement au test suivant de glisser directement de m bit une fois le motif trouvé.
* En théorie, W le nombre d’occurrence de B dans une séquence aléatoire de taille n vérifie : W =somme… Somme de bernouillis m-dépendante de proba ½\*\*m donc d’esperance p et de variance ½\*\*m(1-1/2\*\*m). D’après le théorème centrale limite, ie il faut n-m assez grands, on W suit une loi normale d’espérance n-m+1/2\*\*m et de vriance …
* On calcule grâce à ces valeurs théoriques le chi\_carré observé
* Chi\_carré est une v.a. suivant une chi\_carré distribution de degré de liberté N d’où :

P\_value = p(chi\_carre>chi\_carré(obs)) = …

Overlapping Template Matching Test

Ce test ressemble au précédent sauf que cette fois-ci la fenêtre passe au bit suivant au lieu de sauter m bits quand le motif est trouvé. Le traitement du nombre d’occurrence dans les échantillons est aussi différent. En effet, on peut ici détecter si un motif est trop peu présent ou présent de manière irrégulière.

Description  : - Comme on fera un test chi\_carre, on partitionne la suite en N block de longueur M.

* Pour un motif B de longueur m donné, et pour un échantillon i donné on calcule ni le nombre d’occurrence de B dans l’échantillon. Ce calcul se fait grâce à une fenêtre glissante qui glisse de 1 bits à chaque fois.
* Pour des raisons pratique, on se limite à un degré de liberté égale à 5, on introduite donc v =(v0, v6) et si ni =.. alors on incrément sinon..
* En théorie, si W compte le nombre de présence (avec chevauchement) d’un motif de longueur m dans une séquence aléatoire de taille M et où chaque élément suit une loi binomiale d’esperance ½, alors on a que W suit une loi de Poisson-Géométrique, ie : P(W=u)=…. En fixant M, m pour tout les tests, on a eta = on trouve avec l’expression dans le poly, ces valeurs sont légèrement différentes pour gagner en précision.
* On calcule ensuite le chi\_carre

Linear Complexity Test

Ce test consiste à mesurer la complexité linéaire d’une séquence pour en déterminer sa qualité aléatoire. En effet, on essaie de voir si il existe une équation de récurrence pouvant générer les éléments de la suite. Si tel est le cas, la séquence n’est pas du tout aléatoire. On introduit dans ce test une mesure LSFR (linear feedback shift register). Plus le LSFR est élevée plus la séquence est censé être aléatoire.

Description : - On partitionne en N block de M bits car la théorie ne permet pas de donner un comportement asymptotique valide pour n « trop » grand.

* Pour un échantillon i dans (1,N) donné, on calcule sa complexité Li linéaire grâce à l’algorithme de Berkelamp-Massey…
* On calcule ensuite l’esperance de cette compléxité mu =… . Dans ce calcul , on doit prendre en compte le cas M paire du cas M impaire afin de rectifier cette espérance qui aurait dû être M/2 si Li suivant une loi normal (ce qui n’est pas le cas).
* On calcule ensuite Ti=… pour l’échantillon concerné. Si la suite est aléatoire alors Ti converge vers une v.a. T à valeurs entières vérifiant …
* On se limite à 6 degré de liberté ie on calcule pi0=P(T<=-2.5) .. pi6=P(T>2.5) et on calcule chi\_carre le facteur d’adéquation des échatillons avec la théorie.
* Chi\_carré est une v.a. suivant une chi\_carré distribution de degré de liberté 6 d’où :

P\_value = p(chi\_carre>chi\_carré(obs)) = …

# Analyse des résultats

# Répartition du travail