Modélisation et validation des générateurs de nombres aléatoires cryptographiques pour les systèmes embarqués

Kevin Layat

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques**

Arrêté ministérial : 7 août 2006

Présentée par

Kevin LAYAT

Thèse dirigée par **Philippe ELBAZ-VINCENT**
et codirigée par **Cécile DUMAS**

préparée au sein **de l'Institut Fourier**
et de **l'Ecole Doctorale MSTII**

Modélisation et validation des générateurs de nombres aléatoires cryptographiques pour les systèmes embarqués

Thèse soutenue publiquement le **17 décembre 2015**,
devant le jury composé de :

Mme Cécile DUMAS
Ingénieure CEA-Leti Grenoble, Docteur, Co-Directeur de thèse
M. Philippe ELBAZ-VINCENT
Professeur à l'Université Grenoble Alpes, Directeur de thèse
M. Viktor FISCHER
Professeur à l'Université Jean Monnet, Saint-Etienne, Examinateur
M. Sylvain GUILLEY
Professeur à TELECOM PariTech, Rapporteur
M. David LUBICZ
Ingénieur DGA, Habilité à Diriger des Recherches, Université Rennes 1, Rapporteur
M. Didier PIAU
Professeur à l'Université Grenoble Alpes, Président
M. Emmanuel PROUFF
Expert sécurité des systèmes embarqués à l'ANSSI, Habilité à Diriger des Recherches, Examinateur
M. Yannick TEGLIA
Architecte sécurité et cryptologie à ST Microelectronics, Docteur, Rousset, Examinateur

Remerciements

J'aime les panoramas...
... et pourtant celui des personnes à remercier est particulièrement difficile à dresser.

Tout d'abord je tiens à remercier Philippe et Cécile sans qui ce projet n'aurait jamais pu voir le jour. Tout au long de mon master et de ma thèse ils ont su m'aiguiller, me conseiller, me corriger avec une complémentarité très appréciable.

Merci également aux membres du jury d'avoir accepté de rapporter et d'examiner cette thèse. En particulier David Lubicz et Sylvain Guilley qui, outre le temps passé à étudier la thèse et à rédiger leur rapport, m'ont permis par leurs conseils enrichissants d'avoir une meilleure vision de ce domaine de recherche. Je tiens aussi à faire une mention particulière à Alain Joye et Yannick Teglia pour toutes les questions auxquelles ils ont répondu et les lumières qu'ils m'ont apportées au cours de ces trois années.

Dans la liste des remerciements, je tiens à souligner le Labex Persyval-Lab (ANR-11-LABX-0025) qui a financé en intégralité ma bourse de thèse. Et tout particulièrement Anne-Laure qui, par sa disponibilité, sa réactivité et sa gentillesse, a grandement facilité une multitude de démarches administratives souvent très obscures pour un doctorant.

Je ne peux dissocier le bon déroulement de cette thèse avec l'extraordinaire ambiance qui a régné pendant ces trois années dans le bureau 34C. J'y associe aussi Pierro que j'ai rencontré en licence et qui, de dwish en kawa, m'a accompagné jusqu'au doctorat. S'ils n'avaient pas apprécié les fruits au sirop, les petits biscuits français ou si leur blanquette n'avait pas été si bonne, j'aurais surmonter avec plus de difficulté les aléas émotionnels de la thèse.

Enfin je tiens à remercier ma famille qui m'a toujours soutenue et qui m'a permis de faire un si long parcours dans les études supérieures. Mes amis qui ont toujours été là que ce soit en Haute-Savoie, dans l'Herault ou dans le Var. Et Jessica qui a accepté de la distance pendant ces trois années et qui accepte de me supporter à nouveau désormais !

Table des matières

Acronymes

ANSSI Agence Nationale pour la Sécurité des Systèmes d'Information. 20

BSI Bundesamt für Sicherheit in der Informationstechnik. 20

DRBG Deterministic Random Bit Generator. 3, 15, 17, 37, 38

DSA Digital Signature Algorithm. 16

ECDSA Elliptic Curve Digital Signature Algorithm. 16

FIPS Federal Information Processing Standard. 101

FPGA Field-Programmable Gate Array. 25

IID Indépendant Identiquement Distribué. 59

LFSR Linear Feedback Shift Register. 60, 79–81

META-RO Metastability based Ring Oscillator. 33

NDRBG Non-Deterministic Random Bit Generator. 3, 15, 17, 20, 21, 47, 76, 104, 111

NIST National Institute of Standards and Technology. 20

PLL Phase-Locked Loops. 9, 24

PRNG Pseudo Random Number Generator. 81

QRNG Quantum Random Number Generator. 37

RO Ring Oscillator. 22, 25, 33, 47

SHA1 Secure Hash Algorithm 1. 79

STR Self-Timed Ring. 9, 27–31

TERO Transition Effect Ring Oscillator. 3, 19, 25

TRNG True Random Number Generator. 35, 84

XOR eXclusive OR. 35

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

La protection des données et des communications est un enjeu majeur, tant dans le domaine militaire que dans les domaines industriels et privés. La sécurité de nombreux protocoles cryptographiques (chiffrement, signature, génération de clés, etc) ainsi que les contremesures utilisées pour ces protocoles dépendent de la qualité de séquences de nombres aléatoires. L'exemple de l'algorithme de signature DSA permet de s'en convaincre.

Prenons p et q deux nombres premiers tels que q divise $p-1$. Prenons également $g \in \mathbb{Z}\backslash\{0\}$ tel que $\alpha = (g^{\frac{p-1}{q}} \mod p) \neq 1$ c'est-à-dire que α soit un générateur du groupe cyclique d'ordre q dans $\mathbb{Z}/p\mathbb{Z}$.

Pour un message m, une clé privée $a \in (\mathbb{Z}/q\mathbb{Z})^\times$, un nombre aléatoire $k \in \{0, \ldots, q-1\}$ et une fonction de hachage H on crée alors la signature numérique (r, s) suivante :

$$\begin{aligned} r &= (\alpha^k \mod p) \mod q, \\ s &= (k^{-1}(H(m) + ar)) \mod q. \end{aligned}$$

En supposant que deux messages m_1 et m_2 différents soient signés avec la même graine aléatoire :

$$\begin{aligned} s_1 &= (k^{-1}(H(m_1) + ar)) \mod q, \\ s_2 &= (k^{-1}(H(m_2) + ar)) \mod q, \end{aligned}$$

il est possible pour un attaquant de calculer la valeur de la graine aléatoire :

$$\begin{aligned} s_1 - s_2 &= (k^{-1}(H(m_1) + ar - H(m_2) - ar)) \mod q, \\ &= (k^{-1}(H(m_1) - H(m_2))) \mod q. \\ \text{D'où } k &= (H(m_1) - H(m_2))(s_1 - s_2)^{-1} \mod q. \end{aligned}$$

Ce qui permet ensuite de calculer la clé privée :

$$a = (s_1 k - H(m_1))r^{-1} \mod q.$$

On suppose ici que $s_1 - s_2 \neq 0$, en effet si tel est le cas, $H(m_1) = H(m_2)$ ce qui signifie que m_1 et m_2 forment une collision pour H.

Une méthode similaire permet de retrouver la clé privée à partir de quelques bits de la graine aléatoire pour l'algorithme DSA [NS02] ainsi que pour la version ECDSA basée sur les courbes elliptiques [NS03].

D'autres travaux comme l'étude de la redondance des clés RSA [Hen12, LHA+12] ou le contournement de la protection de la PS3 [Ove10] montrent l'importance primordiale d'un

générateur de nombres aléatoires dans un système cryptographique. Il est alors déterminant de pouvoir analyser la structure ainsi que le comportement de ces dispositifs.

1.1 Structure des générateurs

1.1.1 Générateurs déterministes

Le premier type de générateur de nombres aléatoires est appelé DRBG, ce qui signifie générateur de bits aléatoires déterministe. Le principe de ces générateurs, décrits notamment dans les normes `AIS-31` [KS11] et `NIST-SP 800-90-A` [BKS12a], est basé sur une graine aléatoire qui, par le biais d'un traitement numérique (Fig. 1.1), fournie une séquence de bits entièrement déterminée par la graine initiale.

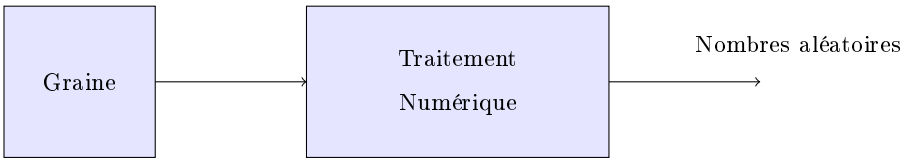


FIGURE 1.1: Principe de fonctionnement général d'un DRBG.

Il existe plusieurs méthodes pour générer des suites déterministes tout en respectant des propriétés d'imprédictibilité. Parmi les outils utilisés pour ce type de générateur, on peut citer les fonctions de hachages, les algorithmes de chiffrement par blocs ou encore les problèmes liés à la théorie des nombres.

Dans la suite nous expliciterons deux familles de DRBG qui seront utilisées au cours du manuscrit pour illustrer certains résultats.

Générateurs basés sur les fonctions de hachage

Le principe général des générateurs déterministes basés sur les fonctions de hachage est le suivant : on construit une suite $x_1, x_2, \ldots, x_n$ en utilisant la formule

$$x_j = H(f(x_{j-1}, \texttt{data})),$$

avec x_0 la graine aléatoire, H une fonction de hachage,f une fonction et `data` des données additionnelles par exemple un compteur.

Dans la suite nous utiliserons un générateur basé sur la fonction SHA1, respectant le schéma donné par le NIST [BKS12a] ainsi que la fonction `Rand` de la bibliothèque `OpenSSL` [Tea15b] qui est également basée sur les fonctions de hachage. Ces deux DRBG ont été

conçus et étudiés pour des applications cryptographiques contrairement aux générateurs que nous allons expliciter dans le paragraphe suivant.

Générateurs basés sur l'algèbre linéaire

Nous allons présenter deux générateurs dont le principe repose sur des mécanismes d'algèbre linéaire. Le premier est le générateur utilisé dans le système de calcul formel `PARI/GP`, basé sur le générateur `XorShift` de Marsaglia [Bre04].

Pour construire la séquence de nombres aléatoires on calcule :

$$x_j = x_{j-1}T,$$

où T est une matrice de taille $n \times n$ à coefficients dans F_2 et la graine x_0 est un vecteur de F_2.

La caractéristique de ce générateur réside dans la construction de la matrice T :

$$T = (I_n + L^a)(I_n + R^b)(I_n + L^c),$$

avec

$$L = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}, \qquad R = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix},$$

I_n la matrice identité de taille n et (a, b, c) un triplet d'entiers choisis de manière à maximiser la période du générateur.

Le second principe que nous allons présenter pour les générateurs déterministes basés sur l'algèbre linéaire est celui des registres à décalage à rétroaction linéaire (LFSR) [Sel66].

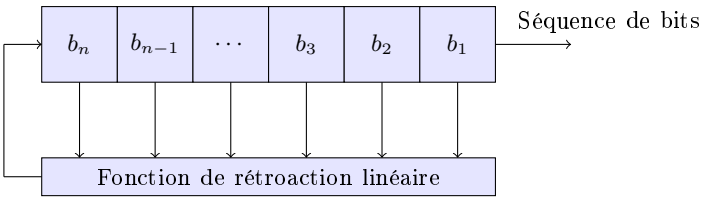


FIGURE 1.2: Principe de fonctionnement général d'un LFSR.

Pour un LFSR la graine constitue l'état initial $(b_1, \ldots, b_n)$, ensuite les bits b_j suivants sont calculés par la fonction de rétroaction linéaire évaluée en l'état $(b_{j-n}, b_{j-n+1}, \ldots, b_{j-1})$.

Le principe de construction des LFSR impose une périodicité aux séquences générées. En

effet, pour un registre de taille n il n'existe que $2^n - 1$ états du registre possibles (en omettant l'état composé de n zéros). Le bit de sortie b_j étant entièrement déterminé par les n bits du registre la période d'un LFSR peut varier entre 1 et $2^n - 1$. Nous étudierons dans la suite plusieurs LFSR de tailles et de périodes différentes.

1.1.2 Générateurs non déterministes

Le second type de générateur de nombres aléatoires est appelé NDRBG, ce qui signifie générateur non déterministe. A la différence des DRBG, la génération ne se fait pas entièrement via un traitement numérique mais à partir d'une source de bruit physique (appelée également source d'entropie) qui est ensuite numérisée pour subir un retraitement algorithmique (Fig. 1.3).

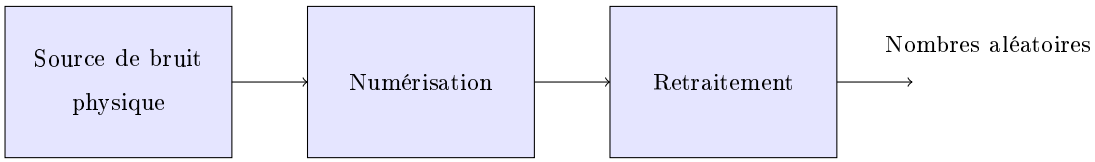


FIGURE 1.3: Principe de fonctionnement général d'un NDRBG.

Sources d'entropie

Il existe de nombreuses sources d'entropie exploitables dans le cadre de l'aléa cryptographique, les plus courantes seront explicitées dans le chapitre 2.

Retraitements algorithmiques

Une fois la séquence issue de la source de bruit physique numérisée, les propriétés statistiques recherchées pour une utilisation cryptographique de l'aléa sont rarement satisfaites. Ainsi, il est nécessaire de recourir à des algorithmes de retraitement pour corriger les défauts de la séquence.

Il est par exemple indiqué dans le Référentiel Général de Sécurité [ANS14] que : « *les règles et recommandations applicables aux générateurs d'aléa se fondent sur le constat qu'il est aujourd'hui très difficile de fournir une preuve convaincante concernant la qualité de l'aléa issu d'un générateur physique, alors qu'il est relativement aisé de se convaincre de la qualité d'un bon retraitement.* » ce qui révèle l'importance du retraitement concernant la certification des générateurs.

La méthode la plus simple pour garantir la qualité de la sorite du générateur est d'utiliser des primitives cryptographiques (par exemple AES) comme retraitement. Cependant le coût de cette opération est important et dans certains contextes comme celui des systèmes

embarqués, la gestion des ressources énergétiques et temporelles ne permet pas ce type de méthodes. Ainsi, des retraitements comme le correcteur de von Neumann [vN51] permettent d'améliorer les propriétés statistiques des séquences à faible coût.

La solution pour concilier un retraitement efficace et des performances raisonnables réside dans l'évaluation des générateurs. En effet, en détectant les défauts statistiques d'une source d'entropie, il serait possible d'adapter le retraitement à chaque instance de génération. Cependant, comme le fait remarquer le RGS, l'évaluation précise et la caractérisation des défauts d'un NDRBG est complexe.

1.2 Méthode d'évaluation de l'aléa

Comme nous l'avons vu précédemment, les nombres aléatoires jouent un rôle crucial dans la cryptographie moderne. Garantir des séquences de qualité est donc un enjeu majeur. Cependant, il n'est pas possible de définir un algorithme, une méthode permettant de distinguer parfaitement les séquences aléatoires des séquences déterministes. Les outils employés de nos jours définissent alors, pour les séquences aléatoires, des propriétés qui sont ensuite étudiées par des tests statistiques.

1.2.1 Tests statistiques

Soit $(\Omega, \mathcal{P}(\Omega))$ un espace probabilisable avec $\Omega = \{0, 1\}$. Notons la source $B = (B_1, \ldots, B_n)$, suite de variables aléatoires à valeur dans Ω et supposons que nous disposons d'un échantillon $b = (b_1, \ldots, b_n)$ de notre source B.

Pour construire un test statistique, on doit d'abord définir une hypothèse H_0 appelée hypothèse nulle et une hypothèse H_a appelée hypothèse alternative. On construit ensuite la statistique $S_B = S(B_1, \ldots, B_n)$; variable aléatoire dont on connaît le comportement sous l'hypothèse H_0 (distribution exacte ou asymptotique). Enfin on détermine une règle permettant de décider si le test est un échec ou un succès pour un échantillon donné. Il existe plusieurs types de règles de décision, la première consistant à définir une région de rejet R parmi les valeurs possibles de la statistique S_b. Ainsi le test sera considéré comme un échec si $S_b \in R$ et comme un succès si $S_b \notin R$.

La seconde règle est de définir un intervalle I ainsi que la p-valeur :

$$p = P(S_B < S_b | H_0) = F_{S_B}(S_b),$$

ce qui permet de quantifier la vraisemblance de la valeur S_b sous l'hypothèse H_0. L'intervalle que l'on définit est en fait :

$$I = [\frac{\alpha}{2}, 1 - \frac{\alpha}{2}],$$

où $\alpha = P[\text{echec}|H_0]$ est l'erreur de première espèce. Le test est donc un succès si $p \in I$ et un échec sinon. Pour les tests d'hypothèses en cryptographie, la valeur standard $\alpha = 10^{-4}$ est présente dans de nombreux tests et batteries.

Une des difficultés de ces tests statistiques est la caractérisation précise de l'hypothèse alternative. En effet, classiquement on définit l'hypothèse H_0 de manière à bien connaître le comportement de la variable S_B sous H_0. Néanmoins, plus le modèle imposé par H_0 est restrictif, plus l'hypothèse alternative est générale et donc difficile à caractériser.

Par exemple, en faisant l'hypothèse H_0 : « *La source B est une suite de variables aléatoires indépendantes et identiquement distribuées de loi Bernouilli de paramètre* $\frac{1}{2}$ », il est aisé de construire des statistiques dont le comportement est connu. Cependant l'hypothèse alternative comprend tous les modèles probabilistes autres que celui de l'hypothèse nulle. Ainsi, lors de l'échec d'un test statistique sous cette hypothèse H_0, aucune information n'est apportée suite à l'échec d'un échantillon.

1.2.2 Entropie

Un indicateur important pour qualifier l'aléa d'une source est l'entropie. Intuitivement, l'entropie représente la quantité d'information que contient un message, une source ou une séquence [Oll]. Par exemple, si l'on a besoin de l'intégralité d'un message pour en comprendre le contenu alors l'entropie est maximale. En revanche, si une partie du message suffit (répétitions) alors l'entropie est moindre. Dans le cadre d'un RBG, on peut voir l'entropie comme le nombre de bits nécessaire à un attaquant pour connaître l'intégralité de la séquence.

Outre l'évaluation d'une source d'aléa, une bonne estimation de l'entropie est également primordiale pour le retraitement algorithmique d'un générateur. En effet, les travaux sur les extracteurs d'entropie [Sha11, Sha04, Tre01] permettent de garantir les propriétés statistiques d'un RBG sous l'hypothèse d'une quantité suffisante d'entropie avant extraction.

La théorie de l'information développée par Claude Shannon [Sha01] donne une définition de l'entropie.

Définition 1.2.1. *Entropie de Shannon*

Soient $(\Omega, \mathcal{P}(\Omega))$ un espace probabilisable et S une source vue comme distribution sur cette espace. L'entropie de Shannon $H(S)$ est alors définie comme :

$$H(S) = -\sum_{x \in \Omega} P(S = x) \log(P(S = x)).$$

Lorsque la source est bien caractérisée, l'entropie de Shannon est facile à estimer et donne

une bonne indication du caractère aléatoire. En revanche lorsque le modèle de la source n'est pas bien identifié l'estimation que donne l'entropie de Shannon à tendance à surestimer l'entropie réelle. Ainsi les travaux de Renyi [Ré61] ont permis de définir toute une famille de mesures d'entropie.

Définition 1.2.2. *Entropie de Renyi*

Soient $(\Omega, \mathcal{P}(\Omega))$ *un espace probabilisable,* S *une source vue comme distribution sur cet espace et* $\alpha \in \mathbb{R}^+ \backslash \{1\}$*. L'entropie* α *de Renyi* $H_\alpha(S)$ *est alors définie comme :*

$$H_\alpha(S) = \frac{1}{1-\alpha} \log \left(\sum_{x \in \Omega} P(S = x)^\alpha \right).$$

Si on prend la limite $\alpha \to 1$ alors on obtient l'entropie de Shannon et en prenant la limite $\alpha \to \infty$ alors on définit l'entropie minimale (ou min-entropie) :

$$H_\infty(S) = -\log(\max_{x \in \Omega}(P(S = x))).$$

Dans la suite du manuscrit ces deux entropies seront calculées pour caractériser les séquences étudiées.

1.2.3 Source d'information

Dans la suite de ce manuscrit nous allons étudier les chaînes de Markov qui sont dans le cadre de la théorie de l'information des « sources d'information ».

Définition 1.2.3. *Source d'information [Ash90]*

Une source d'information est une séquence de variables aléatoires $X_0, X_1, \ldots$*, telle que :*

- *Chaque* X_i *prend des valeurs dans un ensemble fini* Γ*, appelé l'alphabet de la source.*
- *La séquence est stationnaire, c'est-à-dire,*

$$P(X_{i_1} = y_1, \ldots, X_{i_k} = y_k) = P(X_{i_1+h} = y_1, \ldots, X_{i_k+h} = y_k),$$

pour tout $i_1, \ldots, i_k, h \in \mathbb{N}$ *et tout* $y_1, \ldots, y_k \in \Gamma$*.*

Pour ce type de source il est possible de définir une notion d'entropie. De manière intuitive cela représente l'incertitude de l'apparition d'un symbole produit par la source sachant que l'on a observé les symboles précédents. Cette notion est fortement liée aux chaînes de Markov (définies dans le chapitre 3) dont l'état à un instant t est déterminé par un nombre fini d'états antérieurs.

Définition 1.2.4. *[Ash90]*

Soit $X = X_0, X_1, \ldots,$ une source d'information, l'entropie de la source, noté $H\{X\}$, est définie comme :

$$\lim_{n\to\infty} H(X_n|X_0, \ldots, X_{n-1}),$$

avec $H(A|B) = -\sum_{i,j} P(A = a_i, B = b_j) \log(P(A = a_i|B = b_j))$ l'entropie conditionnelle.

En pratique le calcul se fait par la formule donné par le théorème suivant dont la preuve est décrite dans le livre [Ash90] (p. 186).

Théorème 1.2.1. *Si $X_0, X_1, \ldots,$ est une source d'information alors :*

$$H\{X\} = \lim_{n\to\infty} \frac{H(X_0, X_1, \ldots, X_n)}{n+1}.$$

1.2.4 Méthodologies d'analyse actuelles

Les premières méthodologies d'analyse statistique des générateurs d'aléa sont parues dans les travaux de Knuth [Knu97] puis ont été utilisées par Marsaglia [Mar95, MT02] pour créer la suite de tests DieHard. Aujourd'hui, on trouve dans la littérature à la fois des suites de tests statistiques ainsi que des procédures d'évaluation fournies par les standards internationaux.

1.2.4.1 Suites de tests

Parmi les principales suites de tests nous allons détailler celle du FIPS-140 ainsi que celle des tests U01.

FIPS-140

Cette suite [NIS02] comporte quatre tests classiques inspirés des travaux de Knuth et elle permet l'étude des propriétés statistiques sur des échantillons de 20 000 bits (Tab. 1.1). Cet outil a été dans un premier temps amélioré et renommée SP800-22 [RSN+01] afin d'enrichir les tests proposés en incluant notamment les méthodes de Marsaglia puis dans un second temps remplacé par le SP-800-90 que nous présenterons dans la suite.

Nom du test	Description
Monobit	Test du biais
Poker	Test sur les motifs de 4 bits
Run	Test sur les séries de 0 et de 1
LongestRun	Test sur la plus longue série de 0 ou de 1

TABLE 1.1: Description des tests de la suite FIPS 140

Tests U01

Cet outil proposé par L'Ecuyer et Simard [LS07] fait partie d'une bibliothèque C qui fournit des tests statistiques, des batteries de tests ainsi que des implantations de générateurs déterministes. Parmi les batteries proposées, on distingue `Crush`, `SamllCrush` et `BigCrush` qui sont applicables à des séquences de nombres dans l'espace $[0, 1]$ et les batteries `Alphabit` et `Rabbit` qui proposent des tests pour les séquences binaires. Ces dernières, qui nous intéressent particulièrement pour l'analyse des NDRBG, se composent respectivement de 4 et 15 tests (avec plusieurs instances de chaque test) dont les résultats sont exprimés sous forme de p-valeurs contrairement à la batterie FIPS 140. Nous décrivons dans les tableaux suivants les tests des batteries `Alphabit` et `Rabbit`.

Nom du test	**Description**
MultinomialBitsOver	Test de la distribution multinomiale
ClosePairBitMatch	Test sur les points les plus proches en considérant les blocs de bits comme des points d'un espace de dimension t
AppearanceSpacings	Test de Maurer [Mau92]
LinearComp	Calcul de la taille du plus petit LFSR générant la séquence
LempelZiv	Test de compression
Fourier	Test sur les coefficients de Fourier
LongestHeadRun	Idem LongestRun FIPS
PeriodsInString	Test sur la corrélation entre les blocs de 32 bits
HammingWeight	Test sur la proportion de 1
HammingCorr	Test sur la corrélation entre les blocs de L bits
HammingIndep	Test d'indépendance des blocs de L bits
AutoCorr	Test d'auto-corrélation de décalage d
Run	Idem FIPS
MatrixRank	Test sur le rang des matrices carrées
RandomWalk	Test sur la marche aléatoire sur $\mathbb{Z}$ définie par la séquence

TABLE 1.2: Description des tests de la suite Rabbit

Nom du test	**Description**
MultinomialBitsOver	Test de la distribution multinomiale
HammingCorr	Test sur la corrélation entre les blocs de L bits
HammingIndep	Test d'indépendance des blocs de L bits
RandomWalk	Test sur la marche aléatoire sur $\mathbb{Z}$ définie par la séquence

TABLE 1.3: Description des tests de la suite Alphabit

1.2.4.2 Procédures d'évaluation

Les batteries présentées ci-dessus fournissent des tests indépendants pour analyser les générateurs. Or les organismes internationaux de standards et de certification comme le NIST ou le BSI requiert des méthodes d'analyse plus complètes. Ces organismes proposent ainsi des protocoles détaillés pour évaluer la qualité d'un générateur.

AIS-31

Ce protocole d'évaluation [KS11] est composé de deux procédures décrivant au total neuf tests.

Procédure	Nom du test	Description
P_1	T_0	Test sur les doublons
	T_1	Test biais (idem FIPS)
	T_2	Test sur les motifs de 4 bits (idem FIPS)
	T_3	Test sur les suites de 0 et de 1 (idem FIPS)
	T_4	Test sur la plus longue suite de 0 ou de 1 (idem FIPS)
	T_5	Test d'auto-corrélation
P_2	T_6	Test de distribution uniforme
	T_7	Test de distribution multinomiale
	T_8	Test de Maurer

TABLE 1.4: Description des tests du protocole AIS-31

La première procédure se déroule en plusieurs étapes :

— Réalisation du test T_0 sur $2^{16} \times 48$ bits.

— Réalisation des tests T_1 à T_5 sur 257 séquences de 20 000 bits.

La procédure n'est validée que si tous les tests sont réussis. Si une seule des 257 séquences de 20 000 bits échoue alors la procédure est relancée sur une nouvelle séquence de bits. Dans le cas où plusieurs séquences échouent aux tests, la séquence est rejetée.

Pour la seconde procédure P_2, les tests T_6 à T_8 sont réalisés sur la séquence entière et aucun échec n'est toléré pour la réussite de la procédure. En cas de succès, le résultat du test T_8 est utilisé comme estimation de l'entropie minimale.

SP 800-90

La norme SP 800-90 [BKS12b] a été introduite par le NIST pour remplacer la norme SP 800-22. Le principe de cette procédure n'est pas de tester les propriétés statistiques des séquences mais d'en estimer l'entropie.

Procédure	Nom du test	Description
P_1	χ^2 d'homogénéité	Test sur la stationnarité de la source
	χ^2 d'indépendance	Test sur l'indépendance des variables
P_2	Moments d'ordre 1	Estimation de l'entropie minimale
	Markov	Estimation de l'entropie minimale
	Fréquence	Estimation de l'entropie minimale
P_3	Test de santé	Évaluation de la qualité des estimations d'entropie
P_4	Test de proportion	Test en ligne
	Test de répartition	Test en ligne

TABLE 1.5: Description des tests du protocole SP 800-90

Pour plus de clarté nous avons défini 4 procédures expliquant le déroulement du protocole du NIST. La méthode est la suivante :

- Réalisation de P_1.
  - Si P_1 est validée, on calcule l'entropie minimale par la formule donnée par la définition 1.2.2.
  - Sinon on réalise la procédure P_2 afin d'estimer l'entropie minimale. Le résultat le plus pessimiste des trois estimations étant conservé.
- Réalisation de la procédure P_3 pour garantir la qualité des estimations ou du calcul de l'entropie.
- Réalisation de la procédure P_4 pendant la génération de l'aléa pour détecter les défauts en ligne.

1.3 Plan

Tout au long de ce chapitre introductif, nous avons mis en évidence l'importance capitale des nombres aléatoires dans la cryptologie, notamment dans le cadre des implantations matérielles pour lesquelles les protocoles ainsi que les contre-mesures (contre les attaques par canaux cachés par exemple) nécessitent des séquences d'aléa. Nous avons également souligné la difficulté de construire des générateurs aléatoires à partir de phénomènes physiques tout comme celle d'en évaluer les séquences. En outre, définir précisément un modèle stochastique pour un NDRBG, ainsi que des tests adaptés à ce modèle est complexe. Ceci implique que les méthodes d'évaluation génériques emploient, pour ce type de générateur, des hypothèses peu réalistes.

Ainsi, au cours du chapitre 2 nous feront un panorama des sources d'entropies et de leurs modèles stochastiques. Nous pourrons donc étudier les phénomènes physiques, électroniques à l'origine des générateurs non déterministes. Nous proposerons également de nou-

velles sources d'entropie basées sur des phénomènes de physique quantique, la miniaturisation des technologies rendant possible, à l'avenir, leur utilisation.

Dans le chapitre 3 nous proposerons une étude temporelle des séquences de nombres aléatoires basée sur les modèles de Markov. Ces outils mathématiques nous permettent d'établir des modèles plus proches de la réalité physique et d'analyser plus finement les générateurs d'aléa. Nous exposerons ainsi les résultats des algorithmes proposés pour des générateurs utilisés dans l'industrie.

Enfin dans le chapitre 4, en gardant l'objectif de réduire la frontière entre les modèles d'analyse et les modèles physiques, nous généraliserons les outils du chapitre 3 en étudiant les chaînes de Markov cachées. Nous développerons dans un premier temps la théorie liée à ces modèles puis nous analyserons dans un second temps les utilisations des chaînes de Markov cachées dans le cadre des NDRBG.

Chapitre 2

Panorama des sources d'entropie et modèles stochastiques

Sommaire

Les générateurs non-déterministes (présentés dans le paragraphe 1.1.2) sont construits à partir d'une source de bruit qui fournit l'entropie du système. Cet élément est primordial car en cas de défaillance aucun autre composant du système ne pourra, à long terme, compenser le manque d'entropie. Plus précisément, la séquence pourra conserver une imprédictibilité calculatoire grâce aux mécanismes de retraitement algorithmiques mais la notion d'imprédictibilité inconditionnelle ne pourra être vérifiée.

D'autre part, les organismes tels que l'ANSSI [1], le NIST [2] ou le BSI [3], dans leurs recommandations concernant la sécurité des systèmes d'informations, font état de la difficulté d'évaluer rigoureusement ce type de générateur [ANS14, BKS12b, KS11]. Il est ainsi important de donner une description précise de la source de bruit et de justifier ses propriétés probabilistes en explicitant le modèle stochastique associé.

Dans un premier temps, nous allons expliciter les sources d'entropie principales décrites dans la littérature puis dans un second temps nous proposerons de nouvelles sources qui, grâce aux avancées technologiques, pourront à l'avenir être utilisées dans le domaine des NDRBG.

2.1 Sources d'entropie dans la littérature

Dans cette partie nous allons décrire plusieurs sources de bruit étudiées dans la littérature et utilisées dans l'industrie pour la génération d'aléa cryptographique, en explicitant les modèles stochastiques de chacune des sources.

2.1.1 Bruit de phase (Jitter) dans les anneaux d'oscillateurs

Les oscillateurs sont des systèmes qui évoluent autour d'un état d'équilibre, ils sont présents partout dans les systèmes physiques, spécialement en électronique et en optique. On trouve également les oscillateurs dans les systèmes digitaux où la présence d'une référence temporelle est nécessaire afin de synchroniser les opérations.

Un des problèmes majeurs pour les oscillateurs est le bruit car une petite déviation peut entraîner d'importants changements dans le spectre de fréquence du signal périodique produit par l'oscillateur. Ce phénomène est connu sous le nom de bruit de phase ou encore jitter.

Dans un système avec un signal d'horloge on constate que les transitions d'un état à un autre ne sont pas parfaitement déterminées et le jitter est basé sur la variabilité de ces

1. Agence Nationale pour la Sécurité des Systèmes d'Information.
2. National Institute of Standards and Technology.
3. Bundesamt für Sicherheit in der Informationstechnik.

transitions.

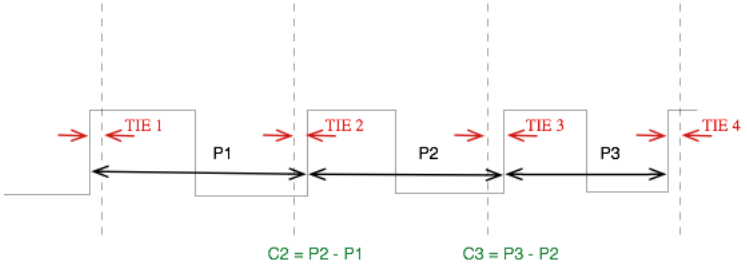


FIGURE 2.1: Différentes mesures du bruit de phase (Schéma tiré de : [CP05]).

Il existe différentes manières de mesurer ce bruit de phase :

— La première technique est le « Time Interval Error » qui s'intéresse à l'intervalle de temps entre le changement d'état théorique et le changement réel (notée TIE sur la Fig. 2.1).

— La seconde technique, « Period Jitter », mesure le temps entre deux périodes (le temps entre deux fronts montants $P1$ à $P3$ sur la Fig. 2.1).

— La troisième technique est le « Cycle to Cycle » qui compare les différences entre les mesures de périodes successives ($C2$ et $C3$ sur la Fig. 2.1).

— On peut également faire les mesures sur un nombre fixé de cycles ce qui permet d'accumuler les effets du jitter.

Ce phénomène de bruit de phase est en général problématique d'un point de vue électronique puisqu'il rend difficile la synchronisation des signaux. En revanche, dans le cadre d'un générateur d'aléa, il est étudié pour ses propriétés probabilistes, étant composé de différents bruits notamment de bruit blanc indépendant dont la distribution est supposée Gaussienne.

Design à base de jitter

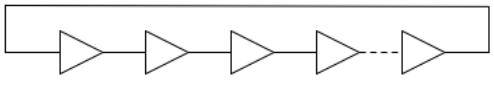


FIGURE 2.2: Anneau d'oscillateurs [CP05].

Un anneau d'oscillateur est composé d'un nombre impair d'inverseurs ce qui a pour effet de produire un signal carré périodique. Le jitter s'accumulant dans l'anneau, on peut alors échantillonner le signal par un signal d'horloge pour produire des séquences d'aléa. L'étude

sur cette catégorie de NDRBG réalisée par Baudet, Lubicz et al. [BLMT10] permet de déterminer un modèle stochastique associé à cette source d'entropie.

Modèle stochastique

Dans les approches classiques, les modèles étaient centrés sur le temps et considéraient les variables aléatoires définies par les durées P_k entre les transitions (voir Fig. 2.1) [CP05, Sch03]. Ces variables P_k étaient alors supposées indépendantes et identiquement distribuées.

Dans la nouvelle approche, le modèle est centré sur la phase du RO, dont l'évolution dans le temps est représentée par une variable aléatoire Gaussienne avec une variance qui croit linéairement ($\sigma^2(t) = ct$) ce qui est une caractéristique des processus de Wiener [DMR00]. D'autres études de modèles pour les anneaux d'oscillateurs existent dans la littérature, on peut citer les travaux de Y. Ma et al. [MLC+14] ainsi que les travaux de thèse de T. Amaki [Ama13, AHMO13] qui proposent une analyse basée sur les chaînes des Markov.

Définition 2.1.1 ([Bou08]). *Un processus stochastique $\{W_t \mid t \geqslant 0\}$ est un mouvement brownien (ou processus de Wiener) de dérive μ et de volatilité σ si :*

- $W_0 = 0$.
- *W_t suit une loi normale de moyenne μt et de variance $\sigma^2 t$.*
- *$\{W_t\}$ est un processus à accroissements stationnaires. Ainsi pour $s < t$, $W_t - W_s$ (qui à la même distribution que $W_{t-s} - W_0 = W_{t-s}$) suit une loi normale de moyenne $\mu(t-s)$ et de variance $\sigma^2(t-s)$.*
- *$\{W_t\}$ est un processus à accroissements indépendants, c'est-à-dire que pour toute séquence de temps $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_n$, les accroissements $W_{t_2} - W_{t_1}, \ldots, W_{t_n} - W_{t_{n-1}}$ sont des variables aléatoires indépendantes.*

Considérant un processus de Wiener $\varphi(t)$ avec μ la dérive et σ^2 la volatilité, ce nouveau modèle statistique permet de calculer :

- La probabilité d'apparition d'un bit à l'instant t :

$$Pr[s(t) = 1 | \varphi(0) = x] = \frac{1}{2} - \frac{2}{\pi}\sin(2\pi(\mu t + x))e^{-2\pi^2\sigma^2 t} + \mathcal{O}(e^{-4\pi^2\sigma^2 t}).$$

- La probabilité d'apparition d'un motif avec $\nu = \mu\Delta t$ et $Q = \sigma^2\Delta t$:

$$\begin{aligned} Pr(b) &= Pr[s(0) = b_1, \cdots, s((n-1)\Delta t) = b_n] \\ &= \frac{1}{2^n} + \frac{8}{2^n\pi^2}\left(\sum_{j=1}^{n-1}(-1)^{b_j + b_{j+1}}\right)\cos(2\pi\nu)e^{-2\pi^2 Q} + \mathcal{O}(e^{-4\pi^2 Q}). \end{aligned} \tag{2.1}$$

- L'entropie d'une séquence de taille fixée.

Grâce à ces calculs de probabilité il est possible de définir un test d'auto-corrélation adapté au modèle. Notons $c(b)$ le coefficient d'auto-corrélation :

$$c(b) = \frac{1}{n-1}\sum_{j=1}^{n-1}(-1)^{b_j+b_{j+1}}. \tag{2.2}$$

On voit que son espérance pour une source parfaite est nulle :

$$E[c(b)] = \sum_{b\in\{0,1\}^n} c(b)p(b) = 0. \tag{2.3}$$

Par un calcul de récurrence on voit également que :

$$\sum_{b\in\{0,1\}^n} c(b)^2 = \frac{2^n}{n-1}. \tag{2.4}$$

On peut alors calculer la formule pour une source non parfaite :

$$\begin{aligned}
E[c(b)] &= \sum_{b\in\{0,1\}^n} c(b)p(b) \\
&\overset{(1)}{=} \sum_{b\in\{0,1\}^n} c(b)\left[\frac{1}{2^n} + \frac{8}{2^n\pi^2}\left(\sum_{j=1}^{n-1}(-1)^{b_j+b_{j+1}}\right)\cos(2\pi\nu)e^{-2\pi^2Q} + \mathcal{O}(e^{-4\pi^2Q})\right], \\
&\overset{(2)}{=} \sum_{b\in\{0,1\}^n} \frac{c(b)}{2^n} + \frac{8(n-1)}{2^n\pi^2}\cos(2\pi\nu)e^{-2\pi^2Q}\sum_{b\in\{0,1\}^n} c(b)^2 + \mathcal{O}(e^{-4\pi^2Q}), \\
&\overset{(3)(4)}{=} \frac{8(n-1)}{2^n\pi^2}\cos(2\pi\nu)e^{-2\pi^2Q}\frac{2^n}{n-1} + \mathcal{O}(e^{-4\pi^2Q}), \\
E[c(b)] &= \frac{8}{\pi^2}\cos(2\pi\nu)e^{-2\pi^2Q} + \mathcal{O}(e^{-4\pi^2Q}).
\end{aligned}$$

2.1.2 Bruit de phase dans les boucles à verrouillage de phase

Les boucles à verrouillage de phase ou boucles à asservissement de phase en anglais Phase-Locked Loops (PLL) ont été inventées en 1932 par Henri De Bellescize. Ce dispositif électronique est un système bouclé permettant d'asservir la différence de phase entre deux signaux de même fréquence.

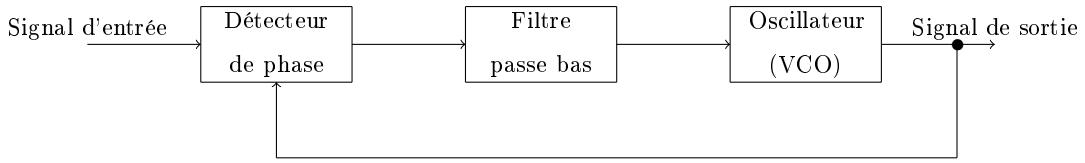


FIGURE 2.3: Schéma fonctionnel d'une boucle à verrouillage de phase.

Dans le schéma présenté, l'asservissement est fait entre le signal d'entrée et celui généré par la boucle. En théorie, la différence de phase asservie est déterministe cependant dans la réalité les signaux sont soumis au bruit de phase, ce qui permet une utilisation pour les générateurs aléatoires.

Dans le cadre des NDRBG, on utilise en pratique un synthétiseur de fréquence basé sur les boucles à verrouillage de phase, ce qui permet d'asservir des signaux de fréquences différentes.

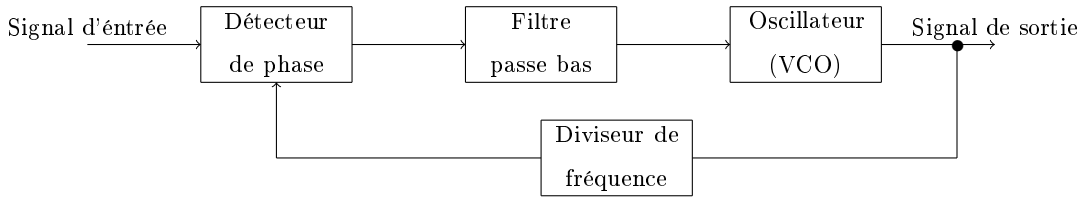


FIGURE 2.4: Schéma fonctionnel d'un synthétiseur de fréquence basé sur une PLL.

Design de générateur à base de PLL

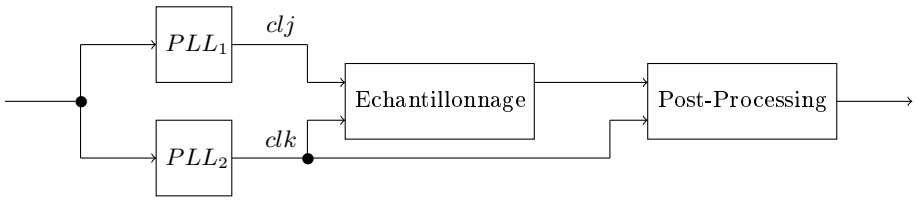


FIGURE 2.5: NDRBG basé sur l'échantillonnage cohérent utilisant des PLL.

Ce design de générateur présenté dans les travaux [BFV10, SDF11] est basé sur l'échantillonnage du signal bruité (jitter) clj d'une première boucle à verrouillage de phase PLL_1 par le signal clk d'une seconde boucle PLL_2. Le signal clj possède une période T_{clj} alors que le signal clk possède une période T, les fréquences des signaux étant reliées de par l'utilisation du synthétiseur de fréquence (Fig. 2.4).

Les travaux [BFV10] mettent en évidence un modèle mathématique permettant, à partir des paramètres des PLLs ainsi que des fréquences des signaux, de calculer la probabilité

d'apparition d'un bit pour la méthode d'échantillonnage cohérent en présence de bruit de phase.

2.1.3 Transition Effect Ring Oscillator TERO

Afin de répondre aux problématiques des anneaux d'oscillateurs, des travaux récents proposent une source d'entropie dérivée des anneaux d'oscillateurs, adaptée pour des architectures FPGA [VD10, DM13, Har11, VDM].

Le principe du circuit repose sur un anneau d'oscillateurs (RO) que l'on peut, par un signal de contrôle, modifier pour créer des perturbations aléatoires. C'est la transition du mode RO au mode perturbé qui crée l'entropie, d'où la dénomination Transition Effect Ring Oscillator.

Design d'un générateur TERO

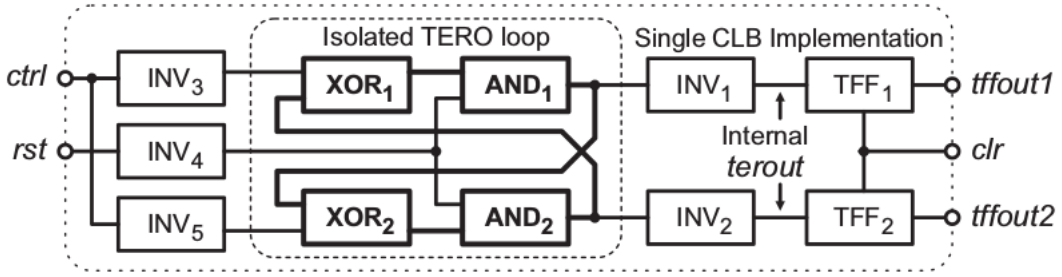


FIGURE 2.6: Exemple de circuit utilisant le phénomène TERO [VD10].

En fonction de la valeur du signal `ctrl` la boucle isolée TERO peut être dans deux modes différents :

— Lorsque `ctrl` vaut 0, la boucle est composée de deux inverseurs (XOR_1 et XOR_2) et de deux tampons (AND_1 et AND_2), le circuit est dans le mode RO.

— Lorsque `ctrl` vaut 1, la boucle est composée de quatre tampons (XOR_1, XOR_2, AND_1 et AND_2), le circuit est dans le mode TERO.

Dans les deux cas la boucle est composée d'un nombre pair d'inverseurs, elle se trouve donc dans un état stable, sans oscillation.

Cependant, quand le signal de contrôle change d'état, les nouvelles valeurs de XOR_1 et de XOR_2 se propagent dans la boucle, ce qui crée une perturbation qui disparaît après quelques oscillations (entre une dizaine et quelques centaines). La parité du nombre d'oscillations pour une période du signal `ctrl` permet alors de produire un bit d'aléa.

Modèle stochastique

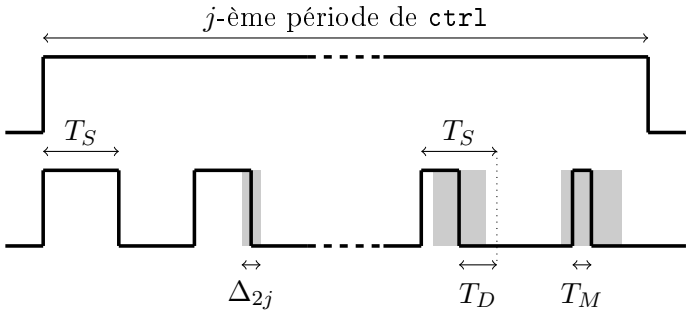


FIGURE 2.7: Interprétation graphique du modèle stochastique des générateurs TERO [VDM].

Lorsque la valeur du signal de contrôle passe à 1 et que le système entre dans le mode TERO, on note T_S le temps du premier pic du signal de sortie.

A chaque oscillation de la boucle TERO, la durée de ce pic diminue d'un temps T_D du fait de l'asymétrie du système.

Cependant, ce temps T_D est soumis au bruit de phase (supposé blanc gaussien) et au bruit de scintillement. Ainsi, en notant Δ_{ij} le bruit pour la i-ème oscillation de la j-ème période du signal `ctrl` et T_M le temps du dernier pic de cette période, l'équation du modèle est :

$$T_S - T_M = \sum_{i=1}^{Y_j}(T_D + \Delta_{ij}) = Y_j T_D + \sum_{i=1}^{Y_j} \Delta_{ij},$$

avec Y_j le nombre d'oscillations pour la période j.

Le modèle prend en compte le bruit de phase et de scintillement (de la famille des bruits roses ou bruits en $\frac{1}{F}$ [Ald94]), en revanche les mêmes interrogations que pour les RO apparaissent concernant la prédominance du bruit de scintillement dans les nouvelles technologies.

Dans les travaux de P. Haddad [Had15], le modèle stochastique proposé pour le générateur de type TERO ne prend pas en compte le bruit de scintillement.

Dans le cas du générateur RO, ce bruit rose pose un problème d'indépendance entre les bits or dans le cas du générateur TERO, le changement de mode entre deux bits de sortie permet de rendre la séquence indépendante. Cependant, pour TERO, la prédominance du bruit de scintillement peut perturber la génération d'aléa en entraînant un biais qui n'est pas prise en compte dans le modèle présenté sur la Fig. 2.7.

2.1.4 Bruit de phase dans les anneaux d'oscillateurs asynchrones

Un oscillateur en anneau asynchrone ou Self-Timed Ring (STR) est composé de L étages connectés, comprenant chacun une porte de Muller et un inverseur [Iss11] (p. 43). Il est également muni d'un protocole de poignée de main (handshaking), pour permettre le transfert de données locales, qui est un des points cruciaux concernant les circuits asynchrones.

Design à partir d'anneaux d'oscillateurs asynchrones

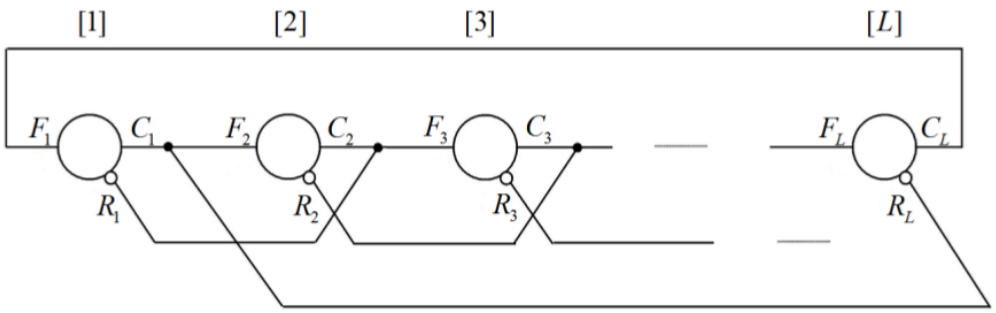


FIGURE 2.8: Structure de générateur basé sur un anneau d'oscillateurs asynchrones [CFAF13a, CFAF13b].

Pour chacun des L étages, F_i représente l'entrée reliée à l'étage précédent (F pour « Forward »), R_i l'entrée reliée à l'étage suivant (R pour « reverse ») et C_i la sortie. La table de vérité correspondant à la porte de Muller suivie d'un inverseur est la suivante :

F	R	C
0	0	C_{t-1}
0	1	0
1	0	1
1	1	C_{t-1}

TABLE 2.1: Table de vérité d'un étage d'anneau d'oscillateur asynchrone

L'entrée F_i est écrite dans la sortie si les deux entrées sont différentes. Dans le cas contraire la sortie n'est pas modifiée.

Introduisons le concept de bulle et de jeton afin de modéliser la propagation des données dans l'anneau. On appelle bulle un enchaînement de deux étages où $C_i = C_{i-1}$ et jeton un enchaînement de deux étages où $C_i \neq C_{i-1}$.

Connaissant la table de vérité et le concept de bulle/jeton, on dit qu'un jeton se propage de l'état i à l'état $i+1$ si et seulement si il est suivi par une bulle. Ainsi la bulle s'est elle même propagée de l'état $i+1$ à l'état i.

Par exemple, prenons la situation d'un jeton suivi d'une bulle, $(C_1, C_2, C_3) = (1, 0, 0)$. On a, par la construction du circuit (fig. 2.8) et l'état de départ :

$$F_2 = C_1 = 1 \text{ et } R_2 = C_3 = 0 \;\; \rightarrow \;\; F_2 \neq R_2.$$

La table de vérité nous permet alors de déduire le changement d'état du circuit :

$$C_2 = F_2 = 1,$$

ce qui donne le nouvel état $(C_1, C_2, C_3) = (1, 1, 0)$. On a ainsi propagé le jeton à l'étage supérieur (car $C_2 \neq C_3$) et la bulle à l'étage inférieur (car $C_1 = C_2$).

En fonction du nombre d'événements initiaux, le STR peut avoir deux modes d'oscillation :

— Mode uniformément espacé ou mode régulier.

— Mode rafale.

Dans le mode uniformément espacé un STR fournit L signaux synchronisés $(C_i)_{1 \leqslant i \leqslant L}$, sujets au bruit de phase, ayant la même période T et une différence de phase moyenne $\Delta\varphi = \frac{T}{2L}$.

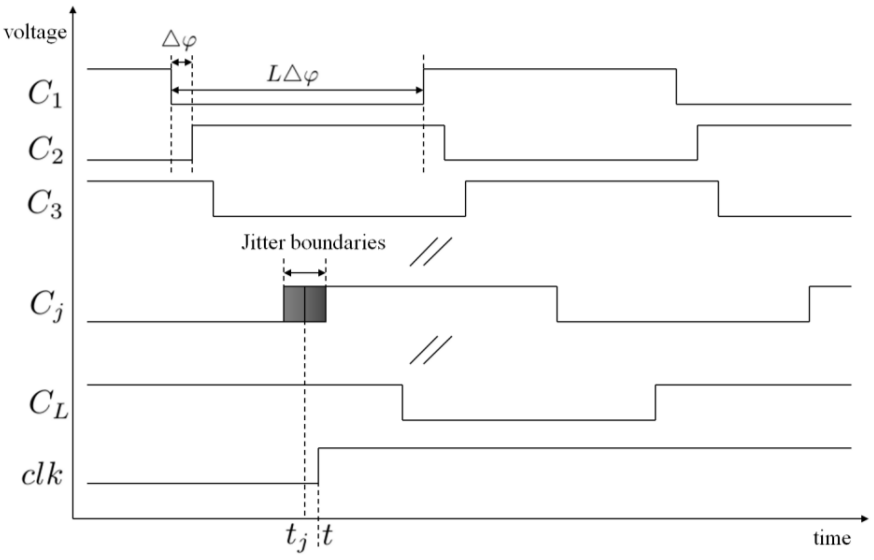


FIGURE 2.9: Échantillonnage des L étages du STR [CFAF13a, CFAF13b].

Une horloge échantillonne les sorties de chaque étage en utilisant une bascule flip-flop. On calcule ensuite la valeur ψ en fonction des résultats $s_i \in \{0,1\}$ de l'échantillonnage des signaux C_i :

$$\psi = \bigoplus_{i=1}^{L} s_i.$$

Quand les jetons sont uniformément espacés, chaque signal étant échantillonné avec la même horloge, pour tout instant t il existe toujours un j tel que :

$$|t - t_j| \leqslant \frac{\Delta\varphi}{2},$$

où t_j représente le temps où le signal C_j change de valeur (Fig. 2.9). Ainsi, si l'intervalle autour de t_j dans lequel le jitter agit sur le signal C_j est plus grand que $\Delta\varphi$, alors C_j est échantillonné dans cet intervalle ce qui implique que s_j est aléatoire et par conséquent ψ l'est également.

Le mode uniformément réparti dépend de plusieurs éléments :

- L'effet Charlie et l'effet d'élaboration [Ham09].
- Les temps de propagation dans un étage de l'anneau :
  - D_{ff} le temps de propagation entre l'entrée avant et la sortie.
  - D_{rr} le temps de propagation entre l'entrée arrière et la sortie.
- Le taux d'occupation de l'anneau $\frac{N_J}{N_B}$ où N_J est le nombre de jetons et N_B le nombre de bulles.

On atteint le mode uniforme lorsque

$$\frac{N_J}{N_B} \approx \frac{D_{ff}}{D_{rr}},$$

c'est-à-dire quand le rapport entre les temps de propagation et le rapport entre le nombre de jetons et de bulles sont similaires.

Modèle stochastique

Dans le mode régulier, les temps moyens $(t_{m_i})_{1\leqslant i\leqslant L}$ de changements d'états des signaux de sortie $(C_i)_{1\leqslant i\leqslant L}$ sont distribués uniformément sur la moitié d'une période d'oscillation (Fig. 2.9). Quitte à les réordonner, on peut supposer sans perte de généralité que $t_{m_1} \leqslant t_{m_2} \leqslant \cdots \leqslant t_{m_L}$. Ainsi, on peut modéliser le temps effectif du changement d'états du signal C_i par une variable aléatoire gaussienne X_i de moyenne t_{m_i} et d'écart type σ correspondant à l'amplitude du bruit de phase.

On considère les variables aléatoires $(X_i)_{1\leqslant i\leqslant L}$ indépendantes car elles sont relatives à des événements distincts situés à deux étages différents du STR.

On peut décomposer $\psi = \mu + \omega$ avec

$$\omega = s_j \oplus s_{j-1} \quad \text{et} \quad \mu = \bigoplus_{\substack{i \neq j \\ i \neq j-1}} s_i.$$

A part dans le cas où $\Delta\varphi \ll \sigma$, μ est entièrement déterminé donc $H(\psi) \approx H(\omega)$ avec $H(\omega)$ qui dépend de X_j et X_{j-1}. En prenant comme origine du temps $\frac{t_{m_j} - t_{m_{j-1}}}{2}$, l'entropie du système est donnée par les variables :

$$X_j = \mathcal{N}(\frac{\Delta\varphi}{2}, \sigma^2) \quad \text{et} \quad X_{j-1} = \mathcal{N}(\frac{-\Delta\varphi}{2}, \sigma^2).$$

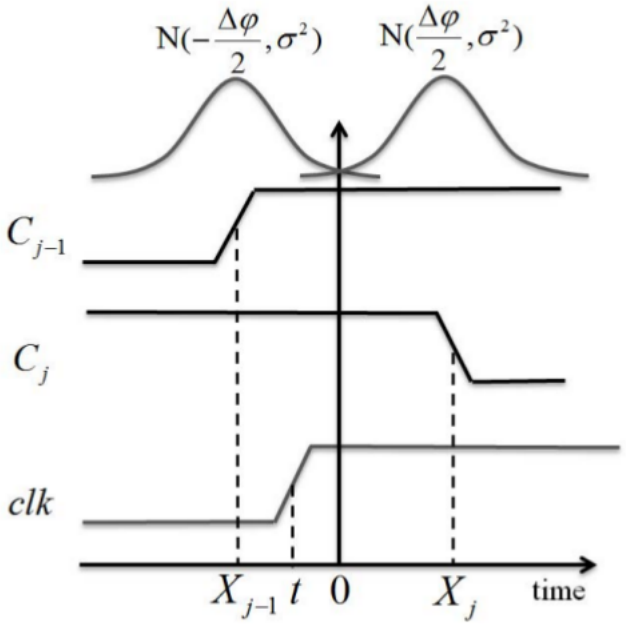


FIGURE 2.10: Extraction d'entropie du STR [CFAF13a, CFAF13b].

μ étant une valeur non aléatoire, en fonction des temps effectifs de changement d'états des signaux C_j et C_{j-1}, la valeur de ψ varie selon la table :

$(X_{j-1} \leqslant t)$	$(X_j \leqslant t)$	ω	ψ
Faux	Faux	1	$\overline{\mu}$
Faux	Vrai	0	μ
Vrai	Faux	0	μ
Vrai	Vrai	1	$\overline{\mu}$

TABLE 2.2: Valeur de ψ en fonction du temps d'échantillonnage

Ainsi en notant $p' = P(X_{j-1} \leqslant t)$ et $p = P(X_j \leqslant t)$ on calcule :

$$\begin{aligned} P(\psi = \mu) &= p(1-p') + p'(1-p), \\ &= p + p' - 2pp'. \end{aligned}$$

Or les distributions des variables X_j et X_{j-1} sont :

$$p = P(X_j \leqslant t) = \Phi(\frac{t - \frac{\Delta\varphi}{2}}{\sigma}) \quad \text{et} \quad p' = P(X_{j-1} \leqslant t) = \Phi(\frac{t + \frac{\Delta\varphi}{2}}{\sigma}).$$

On détermine alors la probabilité d'apparition d'un bit :

$$P(\psi = \mu) = \Phi(\frac{t - \frac{\Delta\varphi}{2}}{\sigma}) + \Phi(\frac{t + \frac{\Delta\varphi}{2}}{\sigma}) - 2\Phi(\frac{t - \frac{\Delta\varphi}{2}}{\sigma})\Phi(\frac{t + \frac{\Delta\varphi}{2}}{\sigma}).$$

L'analyse du modèle stochastique de cette construction de générateur basée sur les oscillateurs en anneaux asynchrones permet de calculer la probabilité d'apparition d'un bit. Par une étude plus approfondie [CFAF13b], il est également possible de déterminer des bornes sur le taux d'entropie des séquences générées.

Conclusion sur les sources d'entropie à base de bruit de phase

Les paragraphes précédents nous ont permis de montrer que le bruit de phase est un phénomène physique très étudié dans le domaine des générateurs de nombres aléatoires et qu'ils existe de nombreux designs et modèles pour ce phénomène. Dans la plupart de ces modèles, le bruit de phase est supposé composé majoritairement de bruit blanc gaussien.

Or les avancées technologiques dans le domaine des transistors tendent vers une miniaturisation des circuits ce qui a pour effet d'augmenter la présence du bruit de scintillement [Had15]. Ainsi, les hypothèses sur lesquelles se base ce type de modèles tout comme les mesures effectuées pour certains designs peuvent en être affectés. Des études faisant suites aux travaux [Had15] sont en cours, afin de déterminer l'influence du bruit de scintillement sur le jitter et son impact sur la qualité de la source d'entropie notamment dans le cadre des générateurs de type RO et TERO.

2.1.5 Métastabilité

En électronique, le phénomène de métastabilité est le fait qu'un système numérique persiste, pour une durée indéterminée, dans un état d'équilibre instable (ou état métastable). Lorsqu'un circuit se trouve dans cet état il agit de manière imprévisible.

Design à base d'inverseurs

Lorsqu'un inverseur boucle sur lui-même, il converge vers un état métastable et oscille autour de cet état.

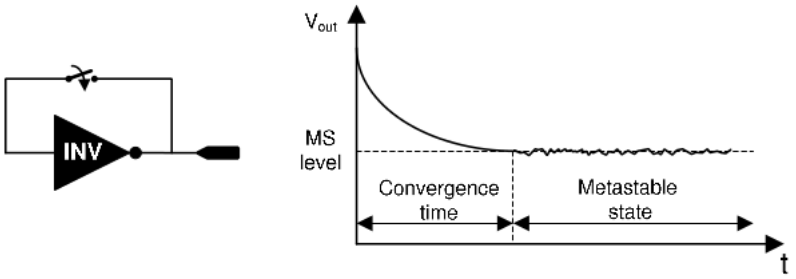


FIGURE 2.11: Phénomène de métastabilité basé sur un inverseur [VHYSK08].

En mettant plusieurs inverseurs en série avec la possibilité pour chaque inverseur de boucler sur lui-même, on crée un anneau d'oscillateur qui exploite le phénomène de métastabilité [VHYSK08].

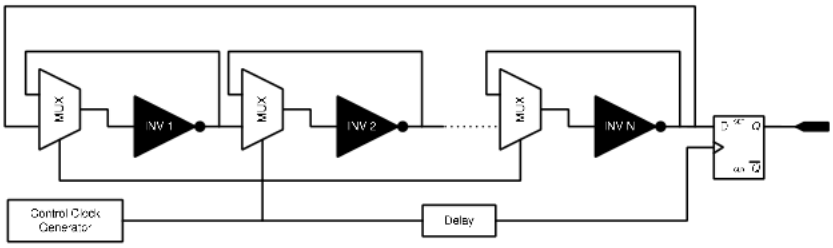


FIGURE 2.12: Anneau d'oscillateur exploitant la métastabilité [VHYSK08].

La génération d'aléa se déroule alors en trois étapes :

1. Une première phase d'initialisation dans laquelle tous les inverseurs bouclent sur eux-mêmes et convergent ainsi vers l'état métastable.
2. Une deuxième phase de transition durant laquelle on reconnecte les inverseurs pour en faire un anneau d'oscillateur. Bien que les états initiaux des inverseurs soient impré-

visibles à cause de la métastabilité, l'amplitude du signal n'est pas assez importante pour réaliser l'échantillonnage. Ainsi cette phase sert à amplifier le signal.

3. Une troisième phase de stabilisation dans laquelle on réalise l'échantillonnage et un retour au mode « métastable ».

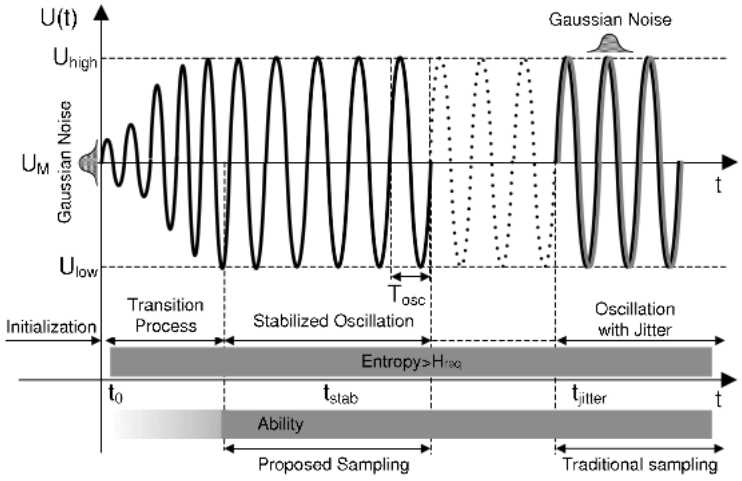


FIGURE 2.13: Signal de sortie pendant les différentes phases du processus [VHYSK08].

Comme on peut le voir sur le graphique la phase d'oscillations stables arrive rapidement alors que le phénomène de bruit de phase met plus de temps à apparaître. Ainsi le générateur basé sur le phénomène de métastabilité (META-RO) a un débit beaucoup plus important que celui basé sur les anneaux d'oscillateurs (RO). A titre de comparaison :

RO	1 Mb / sec.
META-RO	140 Mb / sec.
META-RO + Von Neumann	35 Mb / sec.

TABLE 2.3: Débit des différents générateurs basés sur les anneaux d'oscillateurs

Bien que ce design permette une génération d'aléa rapide, il n'existe pas de modèle stochastique associé dans la littérature. Néanmoins il existe d'autres processus basés sur la métastabilité dont les études stochastiques sont plus détaillées.

Design avec contrôle de qualité basé sur la métastabilité

En électronique, le bon fonctionnement des composants, comme les bascules, dépend de la synchronisation entre les signaux d'entrée et l'horloge. Si cette synchronisation n'est pas

effective, le comportement du système n'est pas prévisible et dépend du bruit environnant, le système est dans un état métastable. L'état final du composant (0 ou 1) est alors utilisable comme bit aléatoire [BRGD13] et le temps de résolution (temps que le composant met pour passer de l'état métastable à l'état logique final) permet de contrôler la qualité de l'aléa [TBM08].

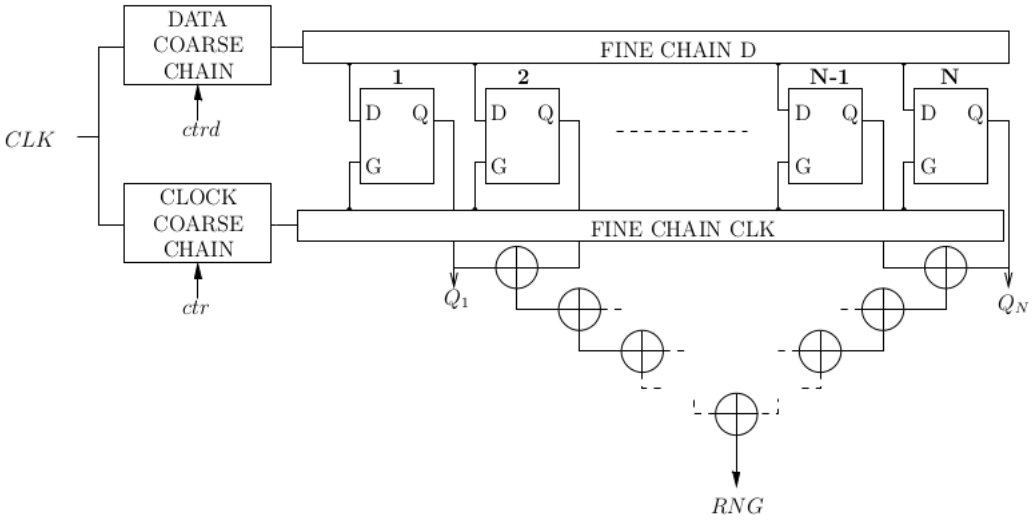


FIGURE 2.14: Structure du générateur basé sur la métastabilité des bascules D [BRGD13].

Dans le design présenté sur la Fig. **??**, N bascules vont être placée dans un état métastable et la sortie du générateur va être une fonction des sorties de chaque bascules (sur la figure 2.14 la fonction est un XOR des sorties deux à deux).

Modèle stochastique

On appelle D le signal d'entrée et G le signal d'horloge de la structure, composée de N bascules (latchs). On note δt_{DG_i} l'écart entre les signaux D et G pour la $i^{\text{ème}}$ bascule. Cet écart est incrémenté entre deux bascules par un différentiel δt :

$$\delta t_{DG_{i+1}} = \delta t + \delta t_{DG_i}.$$

On peut aussi exprimer cet écart comme la somme d'un écart déterministe qui correspond à la propagation des signaux et un écart aléatoire correspondant à l'impact du bruit :

$$\delta t_{DG_i} = \Delta D_0 - i\delta t + \mathcal{N}(\delta t),$$

avec ΔD_0 l'écart initial entre D et G et $\mathcal{N}(\sigma)$ la distribution normale. La bascule i va

échantillonner une valeur logique 1 si l'écart δt_{DG_i} est plus petit que T_{setup0} une valeur expérimentale asymptotique.

Notons P_{q_i} la probabilité que la valeur de sortie Q_i de la bascule i soit égale à 1 et P_S la probabilité que le bit de sortie du générateur soit 1. On a donc :

$$\begin{aligned} p_{q_i} &= P(\delta t_{DG_i} < T_{\text{setup0}}), \\ p_{q_i} &= \frac{1}{2}\left[1 - \text{erf}\left(\frac{\delta t_{DG_i} - T_{\text{setup0}}}{\sigma\sqrt{2}}\right)\right]. \end{aligned}$$

Comme la sortie du TRNG correspond au XOR des valeurs des N bascules, la probabilité que le TRNG émette une valeur égale à 1 est la probabilité qu'un nombre impair de bascules émettent la valeur logique 1. Afin de réaliser l'analyse de probabilité on distingue deux cas :

— Cas 1 : l'influence du bruit sur chaque bascule est indépendante.

— Cas 2 : la valeur de Q_i influe sur Q_{i+1}.

Cas 1 :

$$\begin{aligned} P_{q_1 \oplus q_2} &= P_{q_1}\overline{P_{q_2}} + P_{q_2}\overline{P_{q_1}}, \\ &= P_{q_1}(1 - P_{q_2}) + P_{q_2}(1 - P_{q_1}), \\ &= P_{q_1} + P_{q_2} - 2P_{q_1}P_{q_2}. \end{aligned} \tag{2.5}$$

Ce qui nous donne :

$$\begin{aligned} 1 - P_{q_1 \oplus q_2} &\overset{(2.5)}{=} 1 - 2P_{q_1} - 2P_{q_2} + 4P_{q_1}P_{q_2}, \\ &= (1 - 2P_{q_1})(1 - 2P_{Q_2}). \end{aligned}$$

En généralisant :

$$1 - 2P\left[\bigoplus_{i=1}^{N} Q_i = 1\right] = \prod_{i=1}^{N}(1 - 2P_{q_i}).$$

D'où

$$P_S = \frac{1}{2}\left[1 - \prod_{i=1}^{N}(1 - 2P_{q_i})\right].$$

Cas 2 :

Dans ce cas on considère que la valeur de sortie Q_i de la bascule i influe sur la valeur Q_{i+1} de la bascule $i+1$.

Supposons la dépendance suivant : si $Q_i = 1$ alors $Q_{i+1} \neq 0$. Par exemple avec les règles du premier cas on avait :

$$P_{q_1 \oplus q_2 \oplus q_3} = P_{q_1}\overline{P_{q_2}P_{q_3}} + \overline{P_{q_1}}P_{q_2}\overline{P_{q_3}} + \overline{P_{q_1}P_{q_2}}P_{q_3} + P_{q_1}P_{q_2}P_{q_3}.$$

Alors que dans le deuxième cas on obtient :

$$P_{q_1 \oplus q_2 \oplus q_3} = \overline{P_{q_1}P_{q_2}}P_{q_3} + P_{q_1}P_{q_2}P_{q_3},$$

les 3-uplets (1,0,0) et (0,1,0) ne pouvant pas se produire. Ainsi, on peut généraliser les formules en distinguant la parité de N, pour N pair :

$$P_S = \sum_{i=1}^{\frac{N}{2}} \prod_{j=1}^{2i-1} (1 - P_{q_j}) \prod_{j=2i}^{N} P_{q_j},$$

et pour N impair :

$$P_S = \sum_{i=1}^{\frac{N+1}{2}} \prod_{j=1}^{2i-2} (1 - P_{q_j}) \prod_{j=2i-1}^{N} P_{q_j}.$$

Dans le cadre de cette structure de générateur, l'étude du modèle stochastique [BRGD13] met en évidence la probabilité d'apparition des bits de sortie avec la possibilité de prendre en compte la dépendance entre les différents composants du système.

2.1.6 Propriétés quantiques des photons

La physique quantique, largement étudiée au XX$^{\text{e}}$ siècle, repose en partie sur le principe qu'on ne peut pas prédire le comportement exact de particules comme les photons ou les électrons mais qu'on peut simplement prédire les probabilités associées à ces comportements [Fey70]. Cette branche de la physique est donc très intéressante du point de vue de génération d'aléa, cependant la difficulté de miniaturiser ce genre d'expériences fait que les générateurs quantiques (QRNG) ne sont pas, pour l'instant, très exploités.

Afin d'illustrer les phénomènes quantiques dans le domaine des DRBG, nous allons présenter une expérience d'optique, dans laquelle des photons sont envoyés sur un miroir semi-réfléchissant. Cette expérience, étudiée dans [JAW+00, SGG+00], est l'idée à l'origine du générateur `Quantis` [Qua10a, Qua10b] développé par `ID Quantique`.

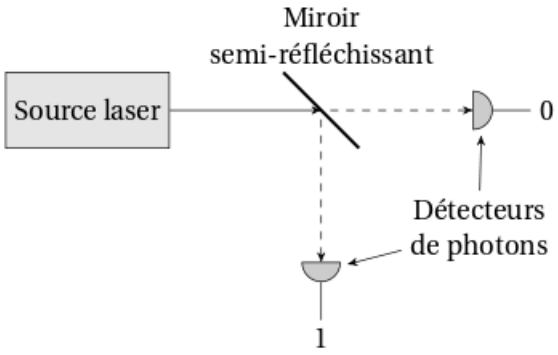


FIGURE 2.15: Trajectoire d'un photon atteignant un miroir semi-réfléchissant [Sou12].

Dans cette expérience, une source laser envoie des photons uniques en direction d'un miroir semi-réfléchissant. Les photons ont, en théorie, une probabilité identique d'être réfléchis ou de traverser le miroir. Ainsi en plaçant deux détecteurs, un dans l'axe de la source laser et un dans l'axe de réflexion du miroir, on construit la séquence de bits en attribuant les valeurs 0 et 1 à l'un ou l'autre des détecteurs (Fig. 2.15).

Le modèle stochastique dépend des paramètres du système. Il faut tenir compte de la qualité source de photons uniques, des propriétés du miroir (qui comporte en pratique un léger biais) ainsi que des défauts des détecteurs (comme les « dark counts » qui représente une détéction sans présence de photon).

Cependant, le générateur `Quantis` n'est pas construit exactement selon ce modèle, mais utilise la probabilité de détection des photons [RG09] :

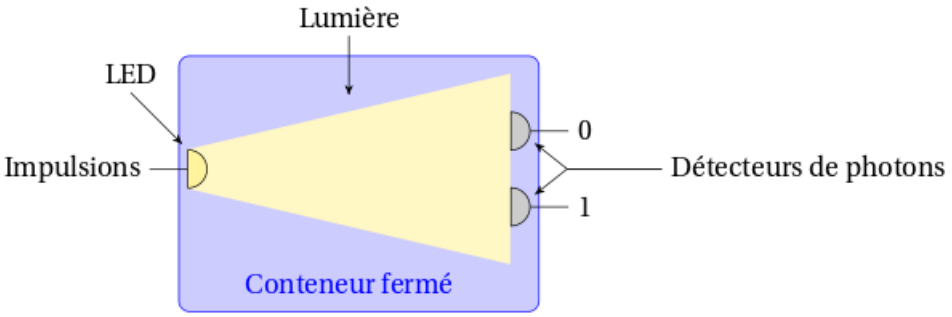


FIGURE 2.16: Trajectoire de photons générés par une LED [RG09, Sou12].

Le principe repose ainsi sur une LED qui fournit un faisceau lumineux et deux photodiodes

à avalanche permettant la détection des photons. En fonction du détecteur activé, un bit 0 ou 1 sera généré, le système assurant selon la documentation [Qua10a] un débit de 4 Mbit/s.

Il existe d'autres phénomènes quantiques étudiés dans la littérature, par exemple une variante de l'expérience précédente, décrite dans [SGG+00], met en jeu un photon qui peut emprunter deux branches de longueur différentes dans une fibre, le temps de détection du photon permettant ainsi de déterminer la valeur du bit de sortie.

On trouve également des expériences basée sur le comptage de photons dans les travaux [FWN+10, WG09].

2.2 Nouvelles sources d'entropie

Il existe dans la nature de nombreux phénomènes présentant un caractère aléatoire, cependant l'aspect crucial pour la génération d'aléa cryptographique est la qualité du modèle stochastique que l'on peut associer à ces phénomènes. En effet, il est important d'avoir des outils mathématiques qui caractérisent fidèlement le comportement de la source dans le temps mais la qualité du modèle dépend également des méthodes d'évaluation (tests adaptés) ainsi que des estimateurs d'entropie que l'on peut associé à ce modèle.

La physique quantique est un domaine propice pour ce genre de processus, en effet, les particules mises en jeu dans les expériences quantiques possèdent des comportements probabilistes et l'étude de l'évolution spatiale et temporelle de ces particules définit des modèles stochastiques précis pour ces expériences [Fey70, CTDRG12a, AFG12].

L'inconvénient de l'utilisation de processus quantiques comme DRBG réside dans la taille des outils nécessaires pour les mettre en œuvre. Cependant, les études récentes, notamment dans le cadre de l'informatique quantique, tendent à miniaturiser ce genre de procédés.

2.2.1 Interactions entre atomes et lumière

Nous allons expliciter dans ce paragraphe des notions de physique quantique résultants de l'interaction entre les atomes et la lumière. L'objectif est de mettre en exergue des phénomènes physiques probabilistes pouvant être utilisés dans le cadre de la génération d'aléa.

Pour un système physique ayant N degrés de liberté, la donnée des N coordonnées $x_1, \ldots, x_N$ ainsi que des vitesses correspondantes $\dot{x_1}, \ldots, \dot{x_N}$ à un instant donné détermine entièrement le mouvement ultérieur. Cette approche, locale en temps, est basée sur les équations du mouvement.

Il est également possible, d'étudier l'évolution d'un système au moyen d'un principe variationnel, global en temps, appelé le principe de moindre action [CTDRG12b, CTDRG12a]. En effet, dans cette approche appelée Lagrangienne, on fait le postulat qu'il existe une fonction $L(x_j, \dot{x_j}, t)$ appelée lagrangien, dépendant des coordonnées, des vitesses et du temps telle que l'intégrale de L entre deux instants t_1 et t_2 soit extrémale quand $x_j(t)$ correspond à l'évolution réelle du système entre t_1 et t_2.

Dans cette approche, les variables doivent vérifier des relations équivalentes aux équations du mouvement :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x_j}}\right) = \frac{\partial L}{\partial x_j},$$

appelée les équations de Lagrange.

Cependant, définir l'impulsion p_j et l'hamiltonien H comme :

$$p_j = \frac{\partial L}{\partial \dot{x_j}}, \qquad H(x_j, p_j) = \sum_{j=1}^{N} \dot{x_j} p_j - L,$$

permet de modifier ce formalisme.

En effet, les équations de Lagrange nous permettant de déduire que $\dot{p_j} = \frac{\partial L}{\partial x_j}$, on peut, en dérivant H, définir les équations de Hamilton comme :

$$\begin{cases} \dot{x_j} = \frac{\partial H}{\partial p_j}, \\ \\ \dot{p_j} = -\frac{\partial H}{\partial x_j}. \end{cases}$$

Ainsi, ces deux formalismes variationnels équivalents (lagrangien et hamiltonien) permettent de décrire l'évolution d'un système physique, le premier établissant N équations d'ordre 2 et le second $2N$ équations à l'ordre 1.

Dans la suite nous considérons des systèmes quantiques décrits par un hamiltonien $\hat{H_0}$ indépendant du temps, $\hat{H_0}$ étant représenté par une matrice de vecteurs propres et de valeurs propres correspondant aux états propres $|n\rangle$ et à leurs énergies E_n.

Pour désigner un système, à l'instant initial $t = 0$, dans l'état le plus général on note :

$$|\psi(0)\rangle = \sum_n \gamma_n |n\rangle \qquad \text{où pour tout } n,\ \gamma_n \in \mathbb{C}.$$

Par les équations de Schrödinger, on connaît l'état du système à un instant t ultérieur [CTDRG12b] :

$$|\psi(t)\rangle = \sum_n \gamma_n e^{-iE_n t/\hbar} |n\rangle\,, \tag{2.6}$$

ce qui nous donne la probabilité de transition de l'état $|\psi(0)\rangle$ vers un état $|\varphi\rangle$ entre 0 et t :

$$P_\varphi(t) = |\langle\varphi|\psi(t)\rangle|^2.$$

Prenons le cas où le système est, à l'instant initial, dans l'état propre $|n\rangle$ de $\hat{H}_0$. Par l'équation 2.6, à l'instant ultérieur t, le système sera dans l'état $|\psi(t)\rangle = e^{-iE_nt/\hbar}|n\rangle$, ce qui implique que la probabilité de trouver le système dans un état propre $|m\rangle$ de $\hat{H}_0$ différent de $|n\rangle$ est nulle.

Dans la réalité, le système est couplé au monde extérieur ce qui permet les transitions vers d'autres états propres. Les interactions en question sont alors modélisées par un terme $\hat{W}$ que l'on ajoute à l'hamiltonien du système isolé $\hat{H}_0$. En d'autres termes on ajoute une perturbation $\hat{W}$ à la description du système qui nous est donnée par $\hat{H}_0$. En général, il n'est pas toujours possible de connaître l'état d'un système de manière exacte à tout instant.

Oscillations de Rabi :

Dans le cadre du système le plus simple, c'est-à-dire un système à deux états $|a\rangle$ et $|b\rangle$ soumis à une interaction $\hat{W}$ constante appliquée à partir de $t = 0$, on connaît la probabilité de transition de $|a\rangle$ vers $|b\rangle$ exacte.

En effet, par la détermination des états propres de l'hamiltonien total $\hat{H}_0 + \hat{W}$ ainsi que le calcul de leurs énergies [AFG12], on trouve la probabilité de transition de l'état initial $|a\rangle$ à l'état $|b\rangle$ en un temps t :

$$P_b(t) = \frac{4\hat{W_{ab}}^2}{(E_a - E_b)^2 + 4\hat{W_{ab}}^2} \sin^2\left(\sqrt{(E_a - E_b)^2 + 4\hat{W_{ab}}^2}\ \frac{t}{2\hbar}\right),$$

où E_a, E_b sont les énergies des états $|a\rangle$ et $|b\rangle$ et $\hat{W_{ab}}$ l'élément non diagonal de la matrice d'interaction $\hat{W}$.

Il en résulte un comportement oscillant de la probabilité de transition que l'on nomme oscillation de Rabi.

Développement perturbatif

Néanmoins, pour des systèmes plus complexes, il n'est pas possible d'avoir de solution exacte. On doit alors utiliser, pour étudier le comportement d'un système, un développement perturbatif.

Pour connaître l'évolution d'un processus physique sous l'effet d'un hamiltonien total $\hat{H} = \hat{H}_0 + \hat{H}_1(t)$, il faut résoudre l'équation de Schrödinger :

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle.$$

Pour cela, on fait l'hypothèse que les éléments de $\hat{H_1}(t)$ sont petits devant ceux de $\hat{H_0}$, ainsi on note :

$$\hat{H_1}(t) = \lambda \hat{H_1'}(t),$$

où les éléments de $\hat{H_0}$ et de $\hat{H_1'}(t)$ sont de même ordre de grandeur et λ est un réel petit devant 1.

En projetant sur un état propre $|k\rangle$ de $\hat{H_0}$ et en utilisant l'expression 2.6 de $|\psi(t)\rangle$ [AFG12], on ramène l'équation de Schrödinger à :

$$i\hbar \frac{d}{dt}\gamma_k(t) = \lambda \sum_n \langle k|\hat{H_1'}(t)|n\rangle \, e^{i(E_k - E_n)t/\hbar}\gamma_n(t), \tag{2.7}$$

qui est un système exact (éventuellement infini) d'équations différentielles.

Pour résoudre ce système, on développe les $\gamma_k(t)$ en série entière de λ :

$$\gamma_k(t) = \gamma_k^{(0)}(t) + \lambda\gamma_k^{(1)} + \lambda^2\gamma_k^{(2)} + \dots \ .$$

Ainsi, en identifiant les termes de même puissance en λ dans l'équation 2.7 on obtient le développement perturbatif :

$$\begin{aligned}
&\text{A l'ordre 0 :} && i\hbar \frac{d}{dt}\gamma_k^{(0)}(t) = 0. \\
&\text{A l'ordre 1 :} && i\hbar \frac{d}{dt}\gamma_k^{(1)}(t) = \sum_n \langle k|\hat{H_1'}(t)|n\rangle \, e^{i(E_k - E_n)t/\hbar}\gamma_n^{(0)}(t). \\
&\text{A l'ordre r :} && i\hbar \frac{d}{dt}\gamma_k^{(r)}(t) = \sum_n \langle k|\hat{H_1'}(t)|n\rangle \, e^{i(E_k - E_n)t/\hbar}\gamma_n^{(r-1)}(t).
\end{aligned}$$

Ces équations différentielles permettent, après intégration sur le temps, de décrire l'évolution du système ayant pour hamiltonien total $\hat{H}$ et d'en calculer les probabilités de transitions. Ainsi, il est possible d'étudier différents types de perturbations $\hat{H_1}(t)$ (constantes, sinusoïdales, ...).

Prenons l'exemple d'un système à deux états soumis à une perturbation.

Transition entre deux états discrets sous l'effet d'une perturbation dépendant du temps :

Supposons le système d'hamiltonien total $\hat{H} = \hat{H_0} + \hat{H_1}(t)$ dans l'état propre $|l\rangle$ de $\hat{H_0}$ à l'instant $t = 0$.

Il vient alors que tous les $\gamma_k^{(0)}(0)$ sont nuls sauf $\gamma_l^{(0)}(0)$ qui vaut 1, c'est-à-dire :

$$\text{pour tout } k, \ \gamma_k^{(0)}(0) = \delta_{kl},$$

où δ est le symbole de Kronecker.

Cette donnée initiale associée à la théorie des perturbations, nous permet de calculer la probabilité de transition de l'état $|l\rangle$ vers l'état $|k\rangle$ entre les instants $t = 0$ et t_1 à l'ordre 1 [AFG12] :

$$P_k(t_1) = |\lambda\gamma_k^{(1)}(t_1)|^2 = \left| \frac{1}{i\hbar} \int_0^{t_1} \langle k|\hat{H}_1(t)|l\rangle \, e^{i(E_k - E_l)t/\hbar} dt \right|^2 . \tag{2.8}$$

Dans le cadre de cet exemple, nous avons déterminé la probabilité de transition en utilisant l'ordre 1, cependant pour des systèmes plus complexes il est nécessaire développer les calculs à des ordres plus élevés.

Parmi les processus dont on peut déterminer le comportement par cette méthode, les phénomènes d'émission et d'absorption induite paraissent adaptés à une utilisation de génération d'aléa.

Émission et absorption induite

Considérons un atome dont l'hamiltonien $\hat{H}_0$ a pour états propres $|a\rangle$ et $|b\rangle$ d'énergies respectivement E_a et E_b avec $E_a < E_b$.

Sous l'action d'une onde électromagnétique de fréquence proche de $\frac{E_b - E_a}{\hbar}$ l'atome dans l'état fondamental $|a\rangle$ peut être porté vers l'état excité $|b\rangle$, c'est le processus d'absorption. L'amplitude de l'onde électromagnétique est alors diminuée.

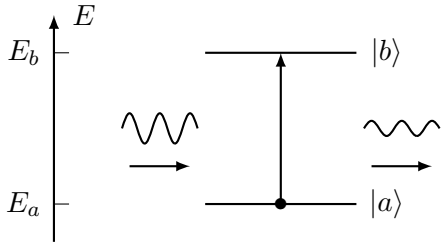


FIGURE 2.17: Processus d'absorption

A l'inverse, lorsque l'atome ce trouve dans l'état excité $|b\rangle$, sous l'effet d'une onde électromagnétique de fréquence proche de $\frac{E_b - E_a}{\hbar}$, l'atome peut être désexcité vers l'état $|a\rangle$, ce que l'on nomme émission induite. L'atome va émettre un photon qui va amplifier l'onde ayant provoqué ce processus.

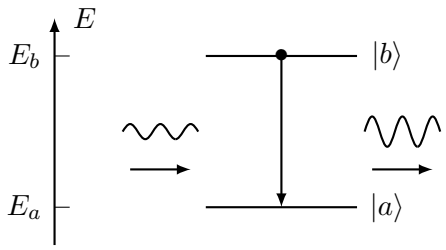


FIGURE 2.18: Processus d'émission induite

Ces deux expériences, impliquant des atomes à deux états, illustrent parfaitement les possibilités d'utilisation cryptographique de phénomènes quantiques.

Dans la suite, nous étudierons un processus plus complexe, faisant intervenir un atome à trois états.

2.2.2 Métastabilité quantique

En physique, on appelle saut quantique un changement brutal de l'état d'un système quantique. On peut citer comme exemple l'interaction entre un atome et le rayonnement. En effet, Bohr suggéra que cette interaction procède par émission et absorption de photons, l'électron de l'atome effectuant des sauts d'une orbite à l'autre. Ce changement d'état de l'atome appelé transition est un saut quantique. Ce genre de phénomènes est notamment étudié pour la construction de q-bits et de sources de photons uniques dans le domaines de l'informatique quantique.

Si on étudie les atomes sous forme de vapeur atomique, le phénomène de saut quantique est inobservable car les transitions individuelles sont noyées dans la masse et la fluorescence observée n'est qu'une valeur moyenne. En revanche on peut piéger des ions et les refroidir grâce à des lasers (technique développée dans les années 80) afin de les étudier séparément. On peut alors suivre le protocole suivant [CK85] :

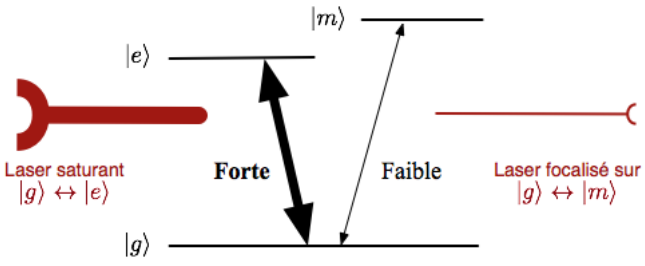


FIGURE 2.19: Sauts quantiques pour une particule à trois états.

Dans ce protocole on étudie un atome à trois états :

— Un état fondamental $|g\rangle$.

— Un état excité $|e\rangle$.

— Un état métastable $|m\rangle$.

La transition $|g\rangle \leftrightarrow |e\rangle$ est très courte ($10^{-8}s$) et très fréquente alors que la transition $|g\rangle \leftrightarrow |m\rangle$ est longue ($\approx 1s$) et très rare.

Le laser intense induit un va-et-vient entre $|g\rangle$ et $|e\rangle$ ce qui a pour effet de produire une fluorescence intense. Le laser qui est focalisé sur la transition $|g\rangle \leftrightarrow |m\rangle$ peut quant à lui porter l'atome à l'état $|m\rangle$ à tout moment. Dès lors la fluorescence est stoppée tant que le l'atome ne revient pas dans le sous-ensemble $\{|g\rangle, |e\rangle\}$. On peut donc savoir si l'atome est dans l'état $|m\rangle$ ou dans l'ensemble $\{|g\rangle, |e\rangle\}$ simplement en observant la fluorescence.

Équations cinétiques

Afin d'écrire les équations cinétiques on introduit les coefficients suivants :

— A_a coefficient d'Einstein d'émission spontanée de l'état $|a\rangle$.

— B_{cb} coefficient d'Einstein d'émission induite de photon entre les états $|c\rangle$ et $|b\rangle$.

— W_i densité spectrale du laser i.

$$\begin{cases} \dfrac{dP_g}{dt} = -(B_{eg}W_1 + B_{mg}W_2)P_g \ + \ (A_e + B_{eg}W_1)P_e \ + \ (A_m + B_{mg}W_2)P_m, & \text{(2.9a)} \\ \dfrac{dP_e}{dt} = -(A_e + B_{eg}W_1)P_e \ + \ B_{eg}W_1P_g, & \text{(2.9b)} \\ \dfrac{dP_m}{dt} = -(A_m + B_{mg}W_2)P_m \ + \ B_{mg}W_2P_g. & \text{(2.9c)} \end{cases}$$

Quand le laser 1 est saturé ($W_1 \to \infty$) on a $P_g = P_e$. Il est alors plus agréable d'utiliser les notations :

$$P_+ = P_m \quad \text{et} \quad P_- = P_g + P_e \qquad \text{d'où} \quad P_g = P_e = \frac{1}{2}P_-.$$

Notons également $R_+ = \frac{1}{2}B_{mg}W_2$ et $R_- = A_m + B_{mg}W_2$. On peut alors écrire l'équation 2.9c comme cela :

$$\frac{dP_+}{dt} = -R_-P_+ \ + \ R_+P_-. \tag{2.10}$$

D'après l'égalité $P_g + P_e + P_m = P_- + P_+ = 1$ on déduit :

$$\frac{dP_-}{dt} = -R_+P_- \ + \ R_-P_+. \tag{2.11}$$

Ainsi, on peut voir la situation comme si R_- allumait la fluorescence en faisant passer l'atome dans le sous-groupe $\{|g\rangle, |e\rangle\}$ et R_+ l'éteignait en faisant passer l'atome dans l'état $|m\rangle$.

Étude probabiliste adaptée des travaux [CK85]

Afin de réaliser une étude précise du processus stochastique on doit introduire les probabilités suivantes :

— $P_{n+}(t)$ la probabilité qu'en un temps t il y ait n transitions entre + et − et que l'état final soit +.

— $P_{n-}(t)$ la probabilité qu'en un temps t il y ait n transitions entre + et − et que l'état final soit −.

Faisons un bilan des événements possibles pour $P_{n+}(t+h)$:

— A : il ne se passe rien entre t et $t+h$. $Pr(A) = 1 - R_-h$.

— B : il y a eu n-1 transitions entre les temps 0 et t et il y a une transition de − à + entre t et $t+h$. $Pr(B) = R_+h$.

— C : il y a eu plus d'une transition entre les temps t et $t+h$. $Pr(C) \in \mathcal{O}(h^{k\geqslant 2})$.

On peut résumer cela par les équations :

$$\begin{cases} P_{n+}(t+h) = (1-R_-h)P_{n+}(t) + R_+hP_{n-1-}(t) + \mathcal{O}(h^{k\geqslant 2}), & \text{(2.12a)} \\ P_{n-}(t+h) = (1-R_+h)P_{n-}(t) + R_-hP_{n-1+}(t) + \mathcal{O}(h^{k\geqslant 2}). & \text{(2.12b)} \end{cases}$$

On peut désormais calculer le taux d'accroissement

$$\begin{aligned} \frac{P_{n+}(t+h) - P_{n+}(t)}{h} &= \frac{1}{h}[(1-R_-h)P_{n+}(t) + R_+hP_{n-1-}(t) + \mathcal{O}(h^{k\geqslant 2}) - P_{n+}(t)], \\ &= \frac{1}{h}[R_-hP_{n+}(t) + R_+hP_{n-1-}(t) + \mathcal{O}(h^{k\geqslant 2})], \\ &= -R_-P_{n+}(t) + R_+P_{n-1-}(t) + \mathcal{O}(h^{k\geqslant 1}). \qquad (2.13) \\ \lim_{h\to 0} \frac{P_{n+}(t+h) - P_{n+}(t)}{h} &= -R_-P_{n+}(t) + R_+P_{n-1-}(t). \qquad (2.14) \end{aligned}$$

Ce que l'on généralise à P_{n-} et résume par les équations :

$$\begin{cases} \dfrac{dP_{n+}}{dt} = -R_- P_{n+} + R_+ P_{n-1-}, & \text{(2.15a)} \\ \dfrac{dP_{n-}}{dt} = -R_+ P_{n-} + R_- P_{n-1+}. & \text{(2.15b)} \end{cases}$$

Comme $P_{-1+} = 0$ et $P_{-1-} = 0$ on a par (13a) et (13b) :

$$\begin{cases} \frac{dP_{0+}}{dt} = -R_- P_{0+} \Leftrightarrow P_{0+}(t) = e^{-R_- t}, \\ \frac{dP_{0-}}{dt} = -R_+ P_{0-} \Leftrightarrow P_{0-}(t) = e^{-R_+ t}. \end{cases}$$

Si on note T_n la durée de la période noire (sans fluorescence) et $O+$ l'événement « À t_0 l'état est $|m\rangle$ et il n'y a pas de transitions jusqu'à $t + t_0$ » on a :

$$P_{0+}(t) = \begin{cases} Pr[O+] = Pr(T_n > t) = 1 - F(T_n), \\ e^{-R_- t}. \end{cases}$$

En suivant le même raisonnement pour T_b la durée des périodes blanches (avec fluorescence) on obtient les fonctions de répartitions suivantes :

$$\begin{cases} F(T_n) = 1 - e^{-R_- t}, \\ F(T_b) = 1 - e^{-R_+ t}. \end{cases}$$

D'où les densités suivantes :

$$\begin{cases} f_n(t) = R_- e^{-R_- t}, \\ f_b(t) = R_+ e^{-R_+ t}. \end{cases}$$

On voit donc que les durées des périodes noires et des périodes blanches sont distribuées selon des lois exponentielles de paramètres $\lambda_n = R_-$ et $\lambda_b = R_+$.

On définit alors la sortie s du générateur en fonction de la variable T_n :

$$\text{Si} \begin{cases} T_n > \frac{\ln(2)}{R_-} & \text{alors} \quad s = 1. \\ T_n < \frac{\ln(2)}{R_-} & \text{alors} \quad s = 0. \end{cases}$$

Ce qui permet d'avoir équidistribution des bits de sorties :

$$\begin{aligned} Pr(s=1) &= Pr(T_n > \frac{\ln(2)}{R_-}), \\ &= e^{-R_- \frac{\ln(2)}{R_-}}, \\ &= \frac{1}{2}. \end{aligned}$$

2.3 Conclusion

Dans ce chapitre nous avons étudié de nombreuses sources d'entropie présentes dans la littérature, qui diffèrent selon les phénomènes physiques mis en jeu, selon l'exploitation de ces phénomènes, selon les modèles stochastiques associés, etc. Nous avons également mis en évidence d'autres sources exploitables, à l'avenir, pour des utilisations cryptographiques notamment grâce à la miniaturisation des technologies ainsi que proposé un modèle stochastique pour la métastabilité quantique.

Cependant, cette miniaturisation peut avoir des effets néfastes sur les modèles, comme nous l'avons vu dans le cadre des RO où le bruit de scintillement modifie le comportement de la source d'entropie. Ainsi, les modèles stochastiques dont nous disposons pour certains générateurs utilisés dans l'industrie ne sont pas entièrement satisfaisants, en particulier l'impact de bruits roses comme le bruit de scintillement reste à étudier précisément.

Cette étape de modélisation des générateurs est d'autant plus cruciale qu'elle est désormais nécessaire pour garantir la certification des NDRBG par les standards [ANS14, KS11, BKS12b]. C'est dans ce cadre de certification qu'il est également primordial de disposer de moyens rigoureux pour justifier l'utilisation d'un modèle pour une séquence ou un générateur donné ainsi que pour garantir le bon comportement d'un NDRBG au cours du temps. Dans la suite nous allons de ce fait étudier différents outils mathématiques permettant d'analyser les modèles et les séquences de générateurs cryptographiques.

Chapitre 3

Chaînes de Markov

Sommaire

Les chaînes de Markov, processus stochastiques introduits par Andrey Markov au début du vingtième siècle, représentent l'une des classes d'objets les plus étudiées dans le domaine des probabilités [Ste09], [Bé13]. Ainsi, ces processus fournissent des modèles très intéressants dans le cadre des générateurs de nombres aléatoires.

En effet, les modèles basés sur les chaînes de Markov intègrent de la dépendance ce qui offre une alternative réaliste (au sens de la réalité physique, électronique du modèle) au modèle classique d'une suite de variables aléatoires indépendantes et identiquement distribuées.

L'objectif de cette partie est donc de présenter les chaînes de Markov, quelques outils probabilistes qui en découlent et de proposer des outils d'analyse de générateurs basés sur ce type de modèle.

3.1 Présentation mathématique des chaînes de Markov

Dans cette section sont présentés les rudiments sur les chaînes de Markov, pour plus de détails on se référera au livre [Ste09].

Définition 3.1.1. *Soit T un sous-ensemble de $\mathbb{R}$, que l'on appelle l'espace de temps. Un processus stochastique d'espace de temps T est alors une famille de variables aléatoires $\{X(t), t \in T\}$. Les variables aléatoires sont à valeurs dans un ensemble Q que l'on appelle l'espace d'états.*

Afin d'illustrer les notions d'espaces de temps et d'états, voici quelques exemples :

- Lors de l'étude du déplacement d'un objet dans l'espace on peut considérer un espace de temps continu $T \subset \mathbb{R}$ et un espace d'états continu $Q \subset \mathbb{R}^3$.
- Une mesure périodique de courant électrique est modélisable comme un processus à espace de temps discret $T \subset \mathbb{N}$ et à espace d'états continu $Q \subset \mathbb{R}$.
- Pour modéliser le nombre de clients à un guichet on construit un processus stochastique à espace de temps continu $T \subset \mathbb{R}$ et à espace d'états discret $Q \subset \mathbb{N}$.
- Pour étudier une langue dont l'alphabet est $\mathcal{A}$, il est possible de considérer un processus à espace de temps discret $T \subset \mathbb{N}$ et à espace d'états discret $Q = \mathcal{A}$. En supposant que l'alphabet comporte n symboles on peut, sans perte de généralité, considérer $Q = \{1, \ldots, n\} \subset \mathbb{N}$.

Définition 3.1.2.

(1) Un processus stochastique est stationnaire, s'il est invariant par décalage du temps. Mathématiquement, pour toute constante $\alpha \in \mathbb{R}$ telle que pour tout $t \in T$, $t + \alpha \in T$ on a :

$$P(X(t_1) = x_1, \ldots, X(t_n) = x_n) = P(X(t_1 + \alpha) = x_1, \ldots, X(t_n + \alpha) = x_n),$$

pour tout $x_i \in Q$ avec $i \in \{1, \ldots, n\}$.

(2) On appelle probabilité de transition d'un processus stochastique la probabilité que le processus passe d'un état x_i au temps t à un état x_j au temps $t + \alpha$, $\alpha \in \mathbb{R}$ tel que $t + \alpha \in T$.

(3) Un processus stochastique est homogène si les probabilités de transitions sont indépendantes du temps écoulé.

Si on a un processus stochastique stationnaire et non homogène, son évolution change au cours du temps mais cette évolution sera la même relativement à l'origine temporelle du processus.

Définition 3.1.3.

(1) Soit $\{X(t), t \in T\}$ un processus stochastique. Si pour tout nombre naturel n et tout état x_n,

$$P(X(n+1) = x_{n+1} | X(n) = x_n, \ldots, X(0) = x_0) = P(X(n+1) = x_{n+1} | X(n) = x_n),$$

on dit que le processus vérifie la **propriété de Markov**.

(2) On appelle chaîne de Markov (respectivement processus de Markov) un processus stochastique à espace d'états discret (respectivement continu) satisfaisant la propriété de Markov.

Pour une chaîne de Markov à espace de temps discret, on peut sans perte de généralité remplacer l'ensemble T par l'ensemble des entiers naturels (on utilise alors la notation X_i pour représenter $X(t_i)$). Dans la suite, sauf mention explicite, les processus considérés seront à espace de temps et espace d'états discrets.

Propriété 3.1.1. *Une chaîne de Markov est homogène si pour tout état i et tout état j :*

$$P(X_{n+1} = j | X_n = i) = P(X_{n+m+1} = j | X_{n+m} = i),$$

pour $n = 0, 1, 2, \ldots$ et $m \geqslant 0$. On note alors :

$$P(X_{n+1} = j | X_n = i) = p_{ij}.$$

Définition 3.1.4. *On appelle temps de séjour le nombre de périodes de temps consécutives où la chaîne de Markov reste dans un état donné. On note R_i les temps de séjour dans l'état i.*

Propriété 3.1.2. *Le temps de séjour dans l'état i pour une chaîne de Markov homogène*

à espace d'états discret, suit une loi géométrique de paramètre $1 - p_{ii}$. *On a donc :*

$$\mathrm{E}[R_i] = \frac{1}{1-p_{ii}} \; et \; \mathrm{Var}[R_i] = \frac{p_{ii}}{(1-p_{ii})^2}.$$

En effet pour une chaîne homogène la probabilité de quitter l'état i est $1 - p_{ii}$. Ce qui implique :

$$P(R_i = k) = (1-p_{ii})p_{ii}^{k-1} \;, \;\; k = 1, 2, \ldots \; .$$

3.2 Méthodologie statistique pour l'étude de modèles markoviens

Il existe dans la littérature des tests statistiques utilisant des modèles markoviens. Le plus célèbre étant le test universel de Maurer [Mau92] qui fait l'hypothèse que la source est modélisée par une chaîne de Markov homogène. En revanche il est plus difficile de trouver des tests étudiant la propriété d'homogénéité du modèle markovien. Cette hypothèse d'homogénéité est pourtant faite par de nombreuses méthodes d'analyse, en effet outre le test de Maurer, les tests classiques comme le Monobit, le Poker, etc font l'hypothèse de variables identiquement distribuées ce qui implique l'homogénéité de la séquence.

Dans les travaux de Donald Knuth [Knu97] on trouve une présentation de tests basés sur les statistiques du χ^2 et de Kolmogorov-Smirnov. Les notions d'anomalies globales et locales de séquences d'aléas sont discutées et l'auteur montre, pour ces statistiques, les différences de détections. Pour illustrer cela, il prend l'exemple d'une séquence non-homogène composée de 100 sous séquences.

— Lorsque les déviations se compensent entre le premier bloc et le second bloc de 50 sous-séquences, le test de χ^2 par effet de moyenne ne détecte pas d'anomalie.

— En revanche, lorsque les déviations s'alternent (déviation positive pour les sous-séquences d'indice pair et négative pour les sous-séquences d'indice impair) alors le test de χ^2 donnera de meilleurs résultats que le test de Kolmogorv-Smirnov.

Cet exemple permet d'appréhender la difficulté d'étudier à la fois les anomalies locales et globales d'une séquence d'aléa.

Dans la suite, nous allons ainsi proposer des outils statistiques d'analyse de séquences dans le but d'étudier l'homogénéité des source d'entropie. Dans la section suivante nous proposerons des outils basés sur la méthodologie de test actuelle puis en étudierons les limites.

3.2.1 Notations et séquences d'illustration

Dans la suite du chapitre nous identifierons la séquence S, composée de N motifs de M bits, à la chaîne de Markov :

$$\xrightarrow{\pi} s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} s_2 \xrightarrow{A_2} \dots \xrightarrow{A_{N-2}} s_{N-1}$$

d'espace de temps $T = \{0, \dots, N-1\}$, d'espace d'états $Q = \{0, \dots, 2^M - 1\}$, de distribution initiale $\pi = (\pi_0, \dots, \pi_{2^M-1})$ et de matrice de transition de taille $2^M \times 2^M$, $A_t = \left(a_{ik}^t\right)_{0 \leqslant i,k \leqslant 2^M-1}$ pour $t \in T$.

On introduit de plus les notations suivantes :

- P le nombre de sous-séquences de S.
- S_j la sous-séquence d'indice $j \in \{0, \dots, P-1\}$.
- $A_S = \left(a_{ij}^S\right)_{0 \leqslant i,j \leqslant 2^M-1}$ et $A_{S_j} = \left(a_{ik}^j\right)_{0 \leqslant i,k \leqslant 2^M-1}$ les matrices de transitions empiriques respectivement calculées sur la séquence S et la sous-séquence S_j. C'est-à-dire le calcul des fréquences de transitions de l'état i à l'état k pour tout i, k pour la séquence S (respectivement S_j).
- $\mathcal{S}(t,i)$ et $\mathcal{S}_j(t,i)$ les nombres de séjours de durée $t \in \mathbb{N}$ en l'élément $i \in Q$ respectivement dans la séquence S et la sous-séquence S_j.
- $\mu(i) = \sum_{t=0}^{\infty} \mathcal{S}(t,i)$ et $\mu_j(i) = \sum_{t=0}^{\infty} \mathcal{S}_j(t,i)$.

Afin d'illustrer les algorithmes présentés dans le chapitre, nous utiliserons des séquences jouets construites à partir de modèles Markoviens dont les matrices de transitions sont de taille $2^M \times 2^M = 16 \times 16$ en considérant l'espace de temps $T = 80\,000\,000$ et l'espace d'états $Q = \{0, \dots, 15\}$.

Les modèles considérés sont entièrement définis par leur matrice de transition et la distribution initiale. Pour construire une séquence on définit donc les 256 valeurs a_{ij}^t de la matrice de transition associée à la chaîne de Markov pour tout $t \in T$ ainsi que la distribution initiale qui est, dans notre cas $\pi = (\pi_0, \dots, \pi_{15}) = (\frac{1}{16}, \dots, \frac{1}{16})$.

Afin de déterminer s_1, le premier élément de la séquence, on tire aléatoirement un nombre réel r_1 entre $[0,1]$. Si on considère la suite d'intervalles :

$$\mathcal{I}_0 = [0, \pi_0],\ \mathcal{I}_1 = [\pi_0, \pi_0 + \pi_1],\ \dots,\ \mathcal{I}_{15} = [\pi_0 + \pi_1 + \dots + \pi_{14}, 1],$$

alors en notant k l'entier tel que $r_1 \in \mathcal{I}_k$ on a $s_1 = k$. Pour construire l'élément s_{n+1} le principe est semblable ; on tire aléatoirement $r_{n+1} \in [0,1]$, puis en considérant $i = s_n$ on construit la suite d'intervalles :

$$\mathcal{I}_0 = [0, a_{i0}^{n+1}],\ \mathcal{I}_1 = [a_{i0}^{n+1}, a_{i0}^{n+1} + a_{i1}^{n+1}],\ \dots,\ \mathcal{I}_{15} = [a_{i0}^{n+1} + a_{i1}^{n+1} + \dots + a_{i\,14}^{n+1}, 1].$$

Si on note k l'entier tel que $r_{n+1} \in \mathcal{I}_k$ alors $s_{n+1} = k$.

Pour créer des séquences non-homogènes on va modifier la matrice de transition au cours du processus de construction de la séquence. Pour que les séquences jouets aient des propriétés différentes on influe sur plusieurs paramètres :

— La valeur des a_{ij}^t pour t fixé : uniforme, fixée (différente de l'uniforme) ou tirée aléatoirement.

— L'évolution au cours de la construction de la séquence des a_{ij}^t.

Voici les valeurs associées aux séquences construites ainsi que l'entropie de Shannon (E1) et l'entropie minimale (E2) par motif (de 4 bits pour ces séquences). Pour simplifier le tableau on indique uniquement à quel pourcentage de la séquence les a_{ij}^t sont modifiés.

Nom	Taille	Valeurs de a_{ij}^t	Changements des a_{ij}^t	E1	E2
`CM_Uni`	40 Mo	$\frac{1}{16}$	aucun	4.000	3.988
`CM_Rand_50`	40 Mo	Aléatoires	1 (à 50%)	3.992	3.750
`CM_Rand_25_50_75`	40 Mo	Aléatoires	3 (à 25%, 50% et 75%)	3.992	3.736
`CM_Fix_25_50_75`	40 Mo	Fixées	3 (à 25%, 50% et 75%)	3.999	3.901

TABLE 3.1: Constructions des séquences jouets basées sur le modèle des chaînes de Markov.

3.2.2 Test d'hypothèse sur l'homogénéité

Les propriétés des chaînes de Markov exposées dans le paragraphe 3.1 nous permettent de déduire un test d'hypothèse vérifiant l'homogénéité des séquences étudiées.

Le test se construit de la façon suivante :

— On fait l'hypothèse que la suite $S = s_1, s_2, \ldots, s_N$ est modélisée par une chaîne de Markov dont la matrice de transition A est connue. On distingue le cas où cette matrice ne dépend pas de la séquence et celui où elle est calculée à partir de S (on spécifie alors en indice de la matrice A la dépendance à la séquence).

— On calcule les temps de séjours pour chaque état de la chaîne de Markov.

— Pour chaque état i, on compare les temps de séjours avec la loi géométrique de paramètres a_{ii} donnés par la matrice A.

On s'intéresse principalement dans le cadre de notre étude aux temps de séjours car on connaît leur comportement théorique pour les chaînes de Markov, ce qui nous permet de construire des tests statistiques. Cependant d'autres caractéristiques des chaînes de

Markov peuvent être étudiés, ainsi nous proposerons dans la suite des outils pour l'étude de la matrice de transition.

La structure de test exposée est très générale, elle peut ainsi conduire à différents tests en fonction de l'explicitation de certains paramètres.

Dans un premier temps, il est nécessaire de préciser la méthodologie de calcul de la matrice A.

En effet, il est possible de définir une matrice A théorique, sans aucune information sur la séquence et sa méthode de génération, par exemple la matrice uniforme où chaque a_{ij} vaut $\frac{1}{2^M}$. Une autre méthode est de calculer une matrice A_{S_j} empirique en utilisant une partie de la séquence étudiée (dans ce cas la sous-séquence S_j), le test se fera alors sur le reste de la source. Enfin, il est possible d'utiliser toute la séquence pour calculer la matrice A_S et vérifier l'homogénéité sur des sous-séquences extraites de la source.

Dans un second temps, il faut expliciter la méthode de comparaison entre la distribution géométrique et les résultats empiriques.

La méthode la plus simple est d'afficher conjointement les résultats empiriques et la distribution théorique dans un graphique. Cette méthode est certes simple mais ne permet pas d'automatiser la comparaison et implique la subjectivité du testeur. On peut alors utiliser un test d'adéquation dont le résultat est une p-valeur. Pour cela il faut disposer d'un test d'adéquation ne faisant pas d'hypothèse d'indépendance de la source. En effet le modèle de la source est supposé Markovien ce qui implique que les variables ne sont pas deux à deux indépendantes. Dans la suite nous expliciterons différents protocoles permettant de tester l'homogénéité d'une source d'aléa.

3.2.3 Test sur les erreurs relatives

Pour ce test on considère les temps de séjours $t \in \{0, 1, 2\}$ pour des raisons de quantité de données. Dans un premier temps on calcule la matrice de transition empirique de la séquence A_S ainsi que les valeurs $\mathcal{S}(t, i)$ pour $t \in \{0, 1, 2\}$ et $i \in Q$. Dans un second temps on détermine l'erreur relative entre les valeurs des temps de séjours observées et les valeurs théoriques :

$$\frac{\mathcal{S}(t,i) - \mu(i)a_{ii}^t(1-a_{ii})}{\mu(i)a_{ii}^t(1-a_{ii})}.$$

Voici un exemple avec les séquences jouets `CM_Uni` et `CM_Rand_50` :

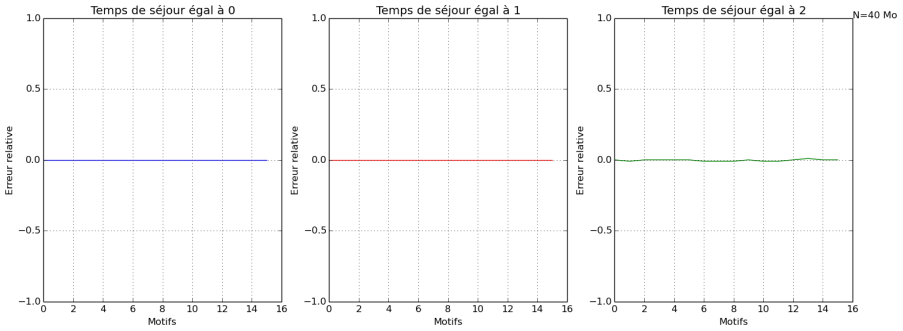


FIGURE 3.1: Erreurs relatives des temps de séjour pour la séquence `CM_Uni`

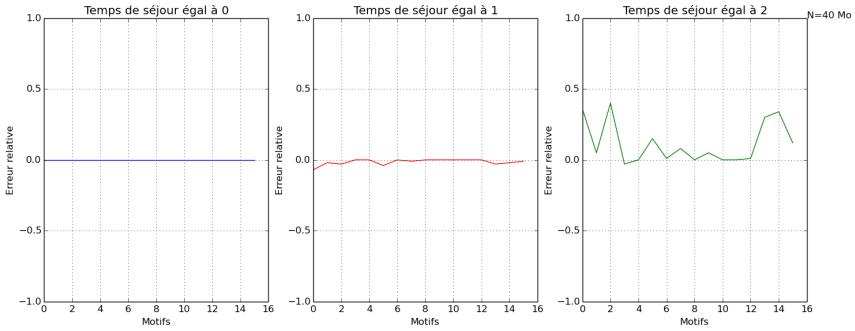


FIGURE 3.2: Erreurs relatives des temps de séjour pour la séquence `CM_Rand_50`

La séquence `CM_Uni` possède des erreurs relatives proches de 0, ce qui confirme l'homogénéité de la séquence alors que la séquence `CM_Rand_50` possède d'importantes déviations notamment pour les temps de séjours $t = 2$ ce qui dévoile un comportement temporel anormal.

Cette interprétation graphique n'est en revanche pas satisfaisante, il est en effet préférable d'avoir des critères plus objectifs pour caractériser l'homogénéité.

3.2.4 Test du χ^2

Le test de χ^2 peut être appliqué de deux manières différentes en fonction du résultat désiré. En effet si on privilégie un résultat graphique on peut calculer plusieurs statistiques sur des sous-séquences afin de les comparer à la distribution de probabilité de la loi de χ^2. Si en revanche on désire un résultat numérique on peut calculer une statistique sur toute la séquence et en déduire une p-valeur. Nous présenterons d'abord la méthode graphique puis la méthode numérique.

Pour la méthode graphique, on calcule tout d'abord la matrice de transition A_S sur toute la séquence. Ensuite, on réalise le test du χ^2 sur les P sous-séquences de S.

Pour toutes les sous-séquences S_j, on calcule pour $t \in \mathbb{N}$ et pour $i \in Q$ les nombres de séjours de durée t en i, $\mathcal{S}_j(t,i)$. Une fois les valeurs connues, on va les regrouper, selon la variable t, de manière identique pour tout i et tout j et de sorte que chaque classe ait une valeur supérieure à 10.

Pour illustrer cela, prenons un exemple numérique avec $j = 2$ et $i \in \{0,1\}$ avec les $\mathcal{S}_j(t,i)$ suivants :

$j=0$				
$i \setminus t$	0	1	2	3
0	62	20	11	0
1	64	24	9	0

$j=1$				
$i \setminus t$	0	1	2	3
0	65	30	19	5
1	53	27	18	11

TABLE 3.2: Effectifs des séjours de durée t.

Si on s'intéresse à la ligne $\mathcal{S}_1(t,1)$, on voit que toutes les valeurs sont supérieures à 10 alors que pour la ligne $\mathcal{S}_0(t,1)$ les valeurs pour $t = 2$ et $t = 3$ sont inférieures. Ainsi, pour que les classes soient les mêmes pour tout i et tout j, on doit définir 2 classes :

$j=0$		
$i \setminus t$	0	1-2-3
0	62	31
1	64	33

$j=1$		
$i \setminus t$	0	1-2-3
0	65	54
1	53	56

TABLE 3.3: Illustration du regroupement en classes des séjours de durée t.

Dans le cas général, en supposant que l'on ait pour tout $i \in Q$, pour tout $j \in \{0,\dots,P-1\}$, d classes $c_{i0}, c_{i1}, \dots, c_{id-1}$ de cardinaux $|c_{i0}|, |c_{i1}|, \dots, |c_{id-1}|$ le regroupement se fait comme suit :

	c_{i0}	c_{i1}	$\cdots$	c_{id-1}
$\mathrm{Emp}_j(c_{ik})$	$\sum_{t=0}^{\lvert c_{i0}\rvert-1} \mathcal{S}_j(t,i)$	$\sum_{t=\lvert c_{i0}\rvert}^{\lvert c_{i1}\rvert-1} \mathcal{S}_j(t,i)$	$\cdots$	$\sum_{t=\lvert c_{id-2}\rvert}^{\lvert c_{id-1}\rvert-1} \mathcal{S}_j(t,i)$
$\mathrm{Th}_j(c_{ik})$	$\mu_j(i)(1-a_{ii}) \sum_{t=0}^{\lvert c_{i0}\rvert-1} a_{ii}^t$	$\mu_j(i)(1-a_{ii}) \sum_{t=\lvert c_{i0}\rvert}^{\lvert c_{i1}\rvert-1} a_{ii}^t$	$\cdots$	$\mu_j(i)(1-a_{ii}) \sum_{t=\lvert c_{id-2}\rvert}^{\infty} a_{ii}^t$

TABLE 3.4: Effectifs empiriques et théoriques des classes de séjours pour la sous-séquence S_j.

Pour la dernière classe, on peut voir que la somme est finie dans le cas des effectifs empiriques alors qu'elle est infinie dans le cas des effectifs théoriques. En effet la séquence étant de taille finie, il existe un entier n tel que pour tout $m > n$, $\mathcal{S}(m,i) = 0$ alors qu'en théorie la distribution géométrique se calcule pour tout $t \in \mathbb{N}$.

Une fois les effectifs empiriques et théoriques des classes obtenus, on calcule la statistique du χ^2 pour chaque $j \in \{0, \dots, P-1\}$:

$$X_j^2 = \sum_{i=0}^{2^M-1} \sum_{k=0}^{d-1} \frac{(\mathrm{Emp}_j(c_{ik}) - \mathrm{Th}_j(c_{ik}))^2}{\mathrm{Th}_j(c_{ik})}.$$

qui suit une loi de χ^2 à $i(d-1)$ degrés de liberté.

On compare alors la courbe théorique de la loi de χ^2 à $i(d-1)$ degrés de libertés avec la courbe de la fonction :

$$\begin{array}{ccc} \mathbb{N} & \longrightarrow & \mathbb{R} \\ n & \longmapsto & \dfrac{\#\{X_j^2 \mid n - \frac{1}{2} \leqslant X_j^2 \leqslant n + \frac{1}{2}\}}{\#X_j^2}. \end{array}$$

Voici un exemple avec les séquences jouets `CM_Uni` et `CM_Rand_50` :

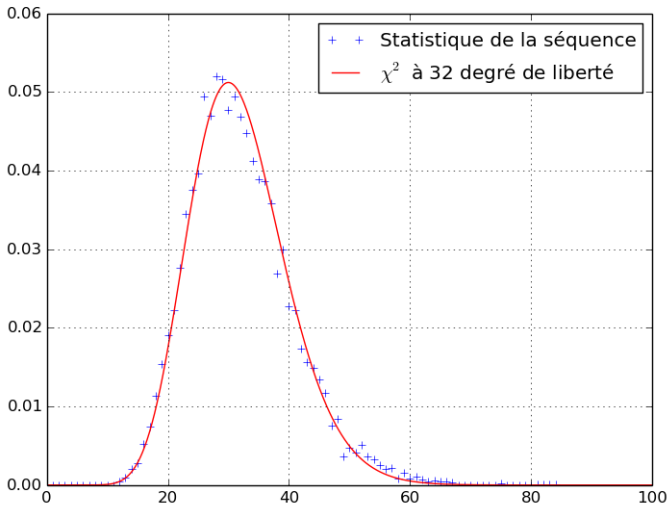


FIGURE 3.3: Test du χ^2 sur les temps de séjour pour la séquence `CM_Uni`

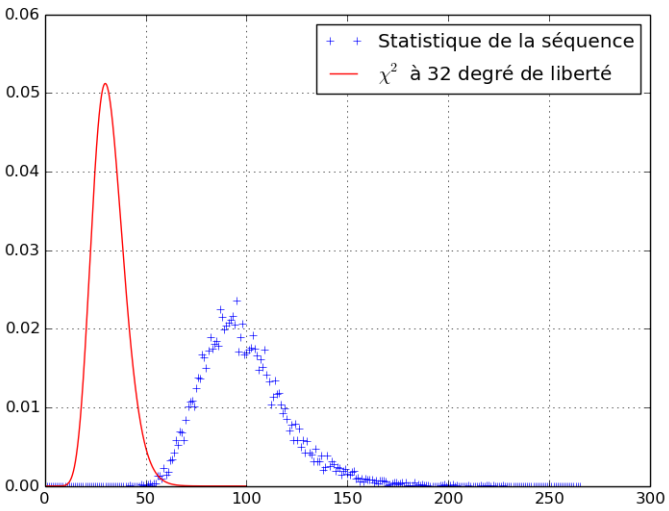


FIGURE 3.4: Test du χ^2 sur les temps de séjour pour la séquence `CM_Rand_50`

Les résultats du test nous permettent de différencier la séquence homogène CM_Uni et la séquence CM_Rand_50 qui possède deux blocs homogènes. En revanche pour CM_Uni, on distingue au niveau de la moyenne, un écart avec la distribution théorique. Cela peut être imputable à une erreur liée à l'hypothèse d'indépendance faite par le test du χ^2.

Pour la méthode numérique, le procédé est similaire avec la différence que les temps de séjours sont calculés sur la séquence globale. En effet, on va calculer la matrice de transition A_S de la séquence et les valeurs $\mathcal{S}(t,i)$ pour $t \in \mathbb{N}$ et $i \in Q$.

Ensuite, on va regrouper les effectifs en classes qui devront être identiques pour tout i. On construit ainsi le tableau analogue à celui de la méthode graphique (Table 3.4) pour la séquence S.

A partir de ce tableau on peut calculer la valeur de la statistique du χ^2 :

$$X^2 = \sum_{i=0}^{2^M-1} \sum_{k=0}^{d-1} \frac{(\mathrm{Emp}(c_{ik}) - \mathrm{Th}(c_{ik}))^2}{\mathrm{Th}(c_{ik})},$$

qui suit une loi de χ^2 à $i(d-1)$ degrés de liberté. Le résultat numérique est alors :

$$p = P(\chi^2(i(d-1)) \leqslant X^2) = F_{\chi^2(i(d-1))}(X^2),$$

où F désigne la fonction de répartition de la loi de χ^2 à $i(d-1)$ degrés de liberté.

Voici les résultats pour les séquences jouets :

Nom	Valeur de X^2	p-valeur
`CM_Uni`	14.3	0.356
`CM_Rand_50`	10883.7	1.000
`CM_Rand_25_50_75`	20775.9	1.000
`CM_Fix_25_50_75`	727775.9	1.000

TABLE 3.5: Test du χ^2 pour les séquences jouets.

Ces deux tests permettent de manière subjective de déterminer si la séquence est globalement homogène ou non. Il est ainsi impossible à partir du résultat, qu'il soit graphique ou sous-forme de p-valeur, d'affiner l'analyse et notamment de se prononcer sur l'homogénéité par morceaux de la séquence.

3.2.5 Modulation des tests et limites de la méthodologie statistique

Dans la méthodologie actuelle d'analyse statistique le principe est de comparer le comportement de séquences d'aléa à celui d'une suite de variables aléatoires IID de loi uniforme. Cela signifie que la source est considérée comme une suite de bits brute sans considération pour les modèles qui sont à l'origine de la génération. Nous allons illustrer cette limite par deux exemples simples.

Supposons dans un premier temps que notre source génère des nombres premiers sur 4 bits et que l'on teste la répartition des motifs de 8 bits. Avec la méthodologie classique, le test va échouer car aucun motif pair autre que 2 ne sera représenté ce qui contredit l'hypothèse d'uniformité. Or ce comportement est normal si l'on tient compte du modèle du générateur. Considérons dans un second temps un générateur déterministe basé sur un registre à décalage à rétroaction linéaire (LFSR) de taille 32. Tester la répartition des motifs de 4 bits sur un échantillon de 20 000 bits ne montrera aucune anomalie alors qu'un test de complexité linéaire n'aura besoin que de 64 bits pour détecter le caractère déterministe du générateur.

Ces deux exemples montrent l'importance de la différenciation des méthodes statistiques en fonction des connaissances théoriques sur les modèles de générateurs. Sans être dans le cas extrême où un test n'est utilisable que pour un générateur spécifique, il faut pouvoir moduler les outils d'analyse pour intégrer les informations théoriques si elles sont disponibles tout en conservant la pertinence des résultats si le modèle de la source est inconnu.

Un autre inconvénient des méthodes présentées précédemment est le manque de clarté des règles de décision. En effet, que ce soit graphiquement ou sous forme de p-valeur, les résultats sont interprétés de manière subjectives par des considérations telles que « les points sont proches de la courbe » ou encore « cette p-valeur est plus grande que le seuil

s ». De plus, les hypothèses faites sur les distributions théoriques des statistiques calculées ne sont pas toujours vérifiées, par exemple dans le cadre de problèmes de convergence ou dans le cadre de source intégrant de la dépendance.

Enfin, l'absence de temporalité de la plupart des tests des batteries actuelles ne permet pas une analyse fine des défauts d'un générateur. Par exemple, pour une séquence composée de deux sous-séquences de même taille, l'une avec 40% de biais et l'autre avec 60% de biais, un effet de moyenne va se produire et les biais vont se compenser ce qui va masquer l'anomalie.

De même, considérons une séquence qui possède une déviation importante dans la répartition des motifs de 4 bits sur son premier quart alors que le reste de la séquence possède une répartition uniforme. Les méthodes statistiques vont détecter cette anomalie mais ne vont pas permettre d'affiner l'analyse en la localisant sur le début de la séquence.

Ces limitations sont d'autant plus importantes à prendre en compte lors de l'étude des générateurs physiques dont les sources sont influencées, au cours du temps, par la modifications des paramètres environnementaux.

3.3 Analyse temporelle de l'homogénéité

Dans la suite, nous allons proposer des méthodes d'analyse de l'homogénéité en tenant compte des inconvénients mis en évidence dans le cadre de la méthodologie actuelle.

3.3.1 Étude de l'homogénéité par morceaux

Les tests d'homogénéité sur la totalité de la séquence sont intéressants mais ne permettent pas de savoir s'il existe à l'intérieur de la séquence des sous-séquences homogènes. Le principe de contrôler l'homogénéité dans le temps est de pouvoir déceler si la séquence est homogène par morceaux et de pouvoir étudier ces blocs homogènes indépendamment.

Pour une première analyse il est intéressant d'étudier l'évolution des a_{ii} en divisant en sous-séquences un échantillon.

Pour cela on va comparer les valeurs de A_S et de A_{S_j} pour $j \in \{0, \ldots, P-1\}$. On s'intéresse principalement ici aux valeurs des diagonales et on utilise l'écart relatif comme outil de comparaison :

$$\frac{a_{ii}^S - a_{ii}^j}{a_{ii}^S}.$$

Voici un exemple graphique pour les séquences jouets `CM_Uni` et `CM_Rand_50` et `CM_Fix_25_50_75` avec $P = 100$:

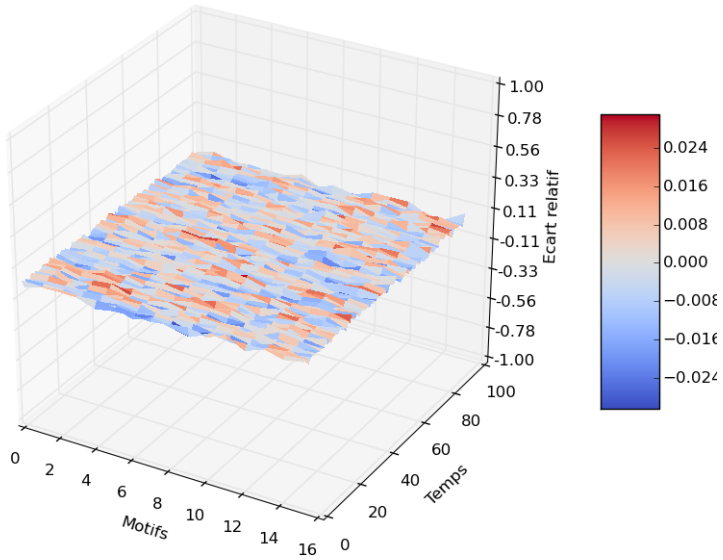


FIGURE 3.5: Test d'écart relatif dans le temps pour la séquence `CM_Uni`

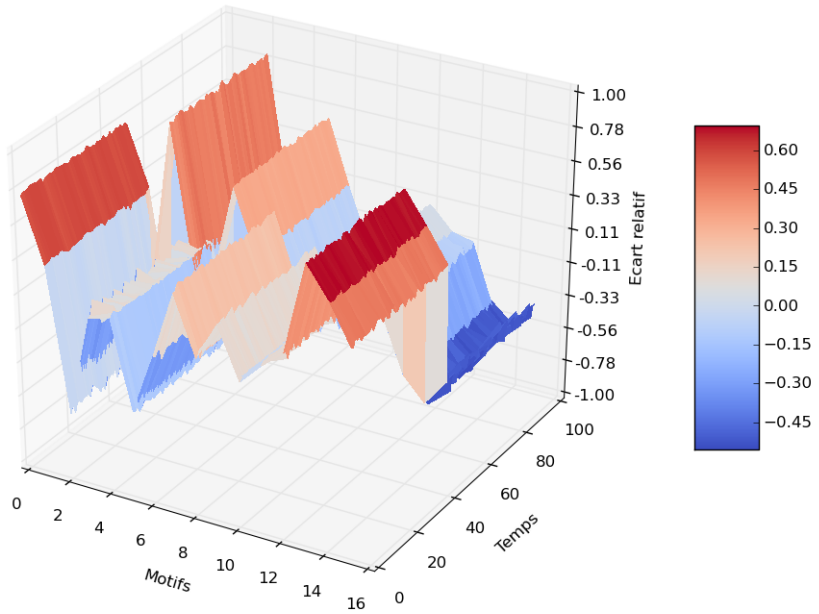


FIGURE 3.6: Test d'écart relatif dans le temps pour la séquence `CM_Rand_50`

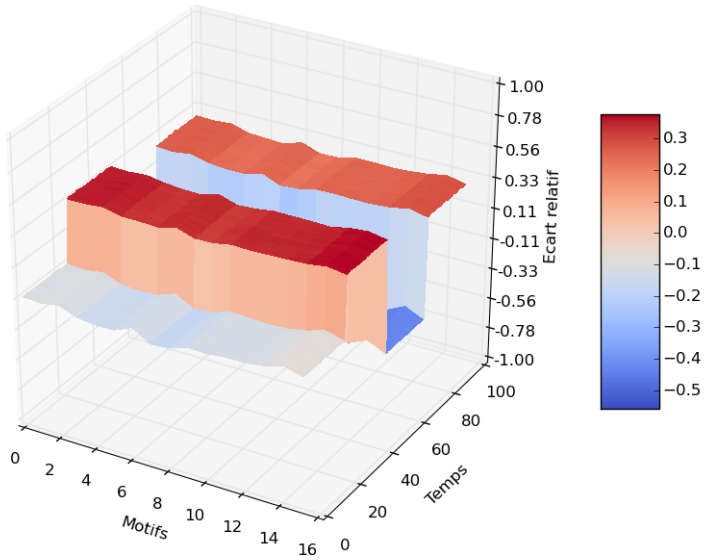


FIGURE 3.7: Test d'écart relatif dans le temps pour la séquence `CM_Fix_25_50_75`

Pour la séquence homogène, on ne discerne pas de rupture dans les valeurs de a_{ii} en revanche on distingue deux états dominants pour la séquence `CM_Rand_50` et quatre états dominants pour la séquence `CM_Fix_25_50_75`.

On peut alors aller plus loin et construire un algorithme qui va détecter les ruptures des valeurs de a_{ii} dans le temps et pouvoir découper la séquence en blocs homogènes par le calcul de p-valeurs sur les blocs.

3.3.2 Algorithme de découpage

L'objectif de l'algorithme est de proposer une analyse temporelle de l'homogénéité afin d'atténuer les effets de moyennes évoqués dans les tests présentés précédemment. Le principe général de l'algorithme est composé de trois étapes :

— **Phase de calcul** : On calcule une valeur v_j pour chaque sous-séquence S_j en fonction des probabilités de transitions de la sous-séquence.

— **Phase de décision** : On décide si la séquence est suffisamment homogène. Si elle l'est on arrête l'algorithme, dans le cas contraire on passe à l'étape de découpe.

— **Phase de découpe** : On détermine avec les valeurs de v_j l'indice de la sous-séquence qui partage la séquence en blocs les plus homogènes. Pour chacun des blocs on revient à l'étape de décision.

Il existe différentes possibilités pour le calcul de v_j ainsi que différentes manières de décider si les blocs sont homogènes. La méthode qui est présentée ci-dessous se rapproche

du cadre classique de l'analyse statistique des générateurs et utilise les régions de rejet et les p-valeurs comme outils décisionnels. Dans les sections suivantes nous discuterons des avantages et inconvénients de ces outils et aborderons d'autres méthodes d'analyse.

Phase de calcul :
Tout d'abord, on doit calculer la diagonale de A_S, c'est-à-dire les valeurs a_{ii}^S pour $i \in \{0, \dots, 2^M - 1\}$.

Ensuite, on détermine pour chaque sous-séquence S_j, $j \in \{0, \dots, P-1\}$, la valeur :

$$ER_j = \sum_{i=0}^{2^M-1} \frac{a_{ii}^j - a_{ii}^S}{a_{ii}^S}.$$

On dispose ainsi d'un tableau de taille P contenant les $(ER_j)_{0 \leqslant j < P}$. Voici le graphique correspondant pour la séquence `CM_Rand_50` :

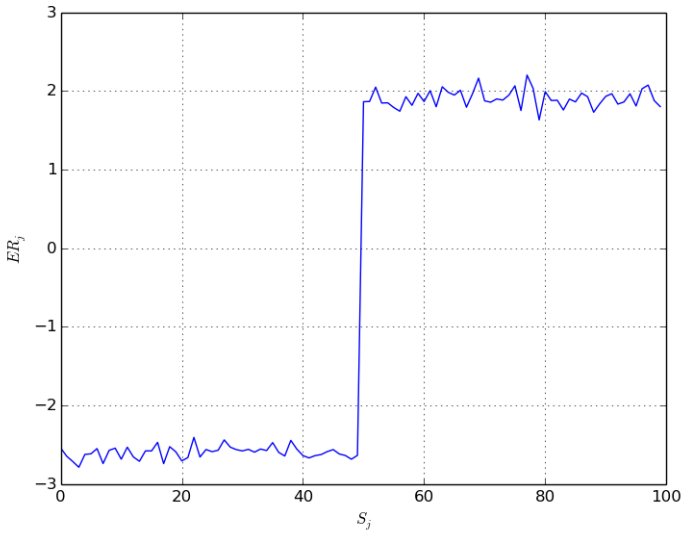


FIGURE 3.8: Écart relatif entre les sous-séquences pour la séquence `CM_Rand_50`

Phase de décision (p-valeur) :
On considère que la séquence est homogène en fonction de la p-valeur issue du test de χ^2 explicité au paragraphe 3.2.4.

Le principe est le suivant :

1. On détermine un intervalle $\mathcal{I}_p$ pour les p-valeurs (exemple : $[10^{-3}, 1 - 10^{-3}]$).
2. On effectue le test de χ^2 sur le ou les blocs de la séquence pour obtenir une p-valeur.
3. Tant que la p-valeur n'est pas dans $\mathcal{I}_p$ on continue l'algorithme par une nouvelle phase de découpe.

L'algorithme se termine lorsque chaque sous-séquence a une p-valeur dans l'intervalle $\mathcal{I}_p$ ou n'est plus coupable.

Phase de découpe :
Une fois les écarts relatifs calculés, on va déterminer le minimum et le maximum des écarts relatifs :

$$\min_{0 \leqslant j < P}(ER_j) \text{ et } \max_{0 \leqslant j < P}(ER_j).$$

On note $j_{\min}$ et $j_{\max}$ les indices des sous-séquences qui ont les écarts relatifs minimum et maximum et

$$\Delta_{\min,\max} = ER_{j_{\max}} - ER_{j_{\min}},$$

l'écart entre ces sous-séquences.

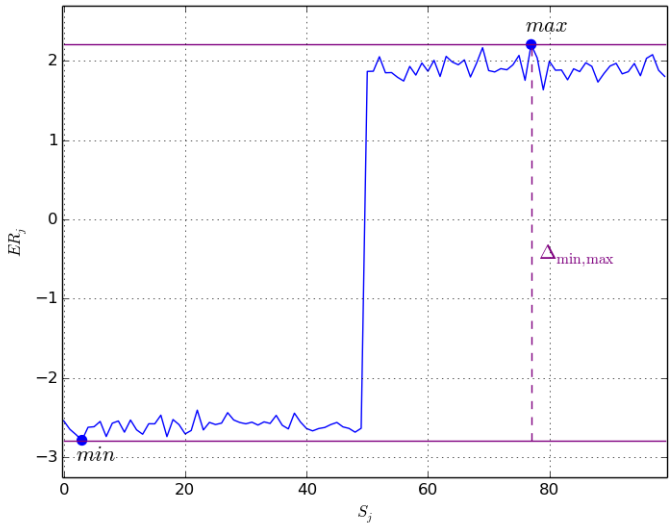


FIGURE 3.9: Algorithme de découpage : Calcul de $\Delta_{\min,\max}$

On va donc chercher à minimiser les valeurs de $\Delta_{\min,\max}$ dans les deux sous-séquences créées par la coupe.

Afin de déterminer la coupe optimale on va tester toutes les sous-séquences S_k comprises entre $S_{j_{\min}}$ et $S_{j_{\max}}$, pour cela on calcule le maximum et le minimum à gauche de la coupure :

$$max_g(k) = \max_{0 \leqslant j \leqslant k}(ER_j) \text{ et } min_g(k) = \min_{0 \leqslant j \leqslant k}(ER_j),$$

ainsi que le maximum et le minimum à droite de la coupure :

$$max_d(k) = \max_{k < j < n}(ER_j) \text{ et } min_d(k) = \min_{k < j < n}(ER_j).$$

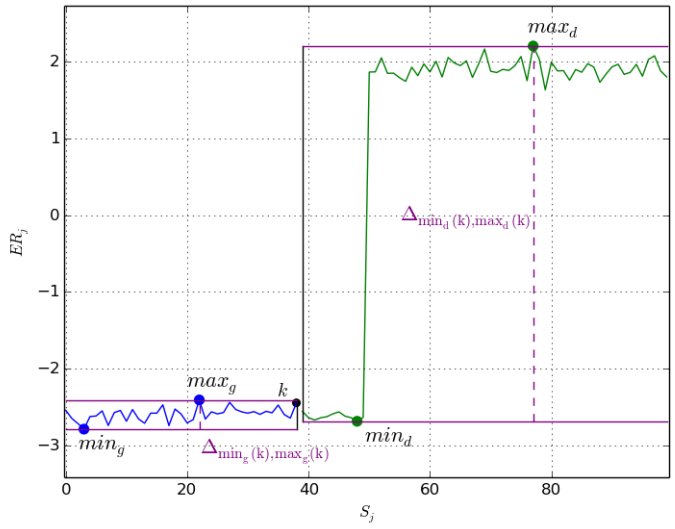


FIGURE 3.10: Algorithme de découpage : Recherche de la coupe optimale

On définit alors l'ensemble des coupes optimales $\mathcal{C}_{\text{opt}}$ comme l'ensemble des sous-séquences S_j, $j \in \{j_{\min}, \ldots, j_{\max}\}$ qui vérifient :

$$\Delta_{\min_g(j),\max_g(j)} + \Delta_{\min_d(j),\max_d(j)} = \min_{j_{\min} \leqslant k < j_{\max}} (\Delta_{\min_g(k),\max_g(k)} + \Delta_{\min_d(k),\max_d(k)}).$$

Dans le cas où l'ensemble $\mathcal{C}_{\text{opt}}$ contient plus d'un élément, on doit faire un calcul supplémentaire afin de déterminer l'unique coupure optimale.

Supposons que $\mathcal{C}_{\text{opt}} = \{S_{o_1}, \ldots, S_{o_q}\}$ avec $o_1, \ldots, o_q \in \{j_{\min}, \ldots, j_{\max}\}$ les indices des q coupures optimales. On définit alors l'indice de la coupure optimale comme :

$$\underset{j \in \{o_1,\ldots,o_q\}}{\arg\min} \left(\left| \frac{\max_g(j) - \min_g(j)}{2} - \sum_{k=o_1}^{j} \frac{ER_k}{j - o_1 + 1} \right| + \left| \frac{\max_d(j) - \min_d(j)}{2} - \sum_{k=j+1}^{o_q} \frac{ER_k}{o_q - j} \right| \right).$$

De manière intuitive, on cherche la coupure qui « centre » les moyennes des écarts relatifs à gauche et à droite.

Dans l'exemple étudié, la coupure optimale se situe au niveau de la sous-séquence 49 :

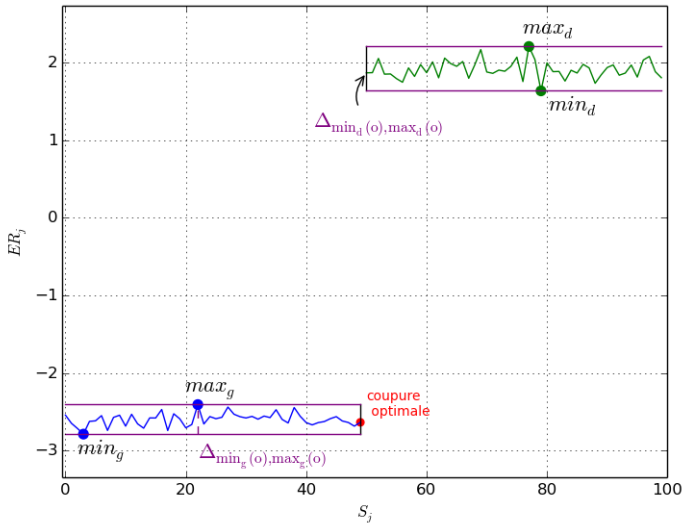


FIGURE 3.11: Algorithme de découpage : Coupe optimale

Complexité et résultats :
Pour déterminer la complexité de l'algorithme dans le pire cas on calcule le nombre d'opérations élémentaires à exécuter :

	Nombre d'opérations
$+$	$2P^2 + (2^{M+4} + 14)P + 11 \cdot 2^{M+1} \ln_2(P) + 2\frac{N}{P} + (6N - 15 \cdot 2^{M+1} - 15)$
$\times$	$P^2 + (2^M + 2)P + 3 \cdot 2^{M+3} \ln_2(P) + (N - 3 \cdot 2^{M+2})$
$\div$	$2^{M+1}P + 2^{M+3} \ln_2(P) - 2^M$

TABLE 3.6: Nombres d'opérations élémentaires pour l'algorithme de découpage.

A titre d'exemple pour les paramètres $N = 80\,000\,000$, $P = 100$ et $M = 4$, le nombre d'opérations est :

	Nombre d'opérations
$+$	240848844
$\times$	40014160
$\div$	4035

En prenant les paramètres $\mathcal{I}_p = [10^{-3}, 1 - 10^{-3}]$ et $P = 100$, les découpages et p-valeurs des séquences jouets sont les suivants :

Séquence	Indice	Début	Fin	p-valeur
`CM_Uni`	1	0	99	0.356
`CM_Rand_50`	1	0	49	0.832
	2	50	99	0.053
`CM_Rand_25_50_75`	1	0	24	0.400
	2	25	49	0.927
	3	50	74	0.676
	4	75	99	0.204

TABLE 3.7: Découpage des séquences jouets.

Voici l'illustration graphique des résultats pour la séquence `CM_Rand_25_50_75` :

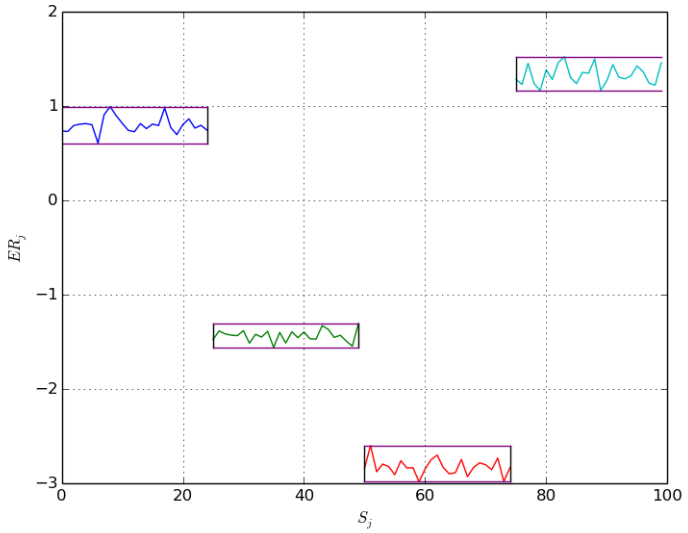


FIGURE 3.12: Algorithme de découpage : Résultat pour la séquence `CM_Rand_25_50_75`

Les séquences jouets nous permettent de voir que l'algorithme est capable de retrouver les sous-séquences homogènes et les indices des transitions entre ces sous-séquences.

En revanche on peut s'interroger sur les avantages et les inconvénients de la méthode de décision basée sur les p-valeurs. En effet, la p-valeur nous donne, par un calcul probabiliste sur un échantillon, une valeur d'adéquation à une loi (ici la loi du χ^2) mais ne donne en aucun cas un critère de décision pour l'échantillon. La décision se fait alors en fixant une région de rejet qui classe les p-valeurs de manière binaire : bonnes ou mauvaises. Dans notre cas la région de rejet est $\mathcal{I}_p$.

Bien que l'idée d'avoir un « score » représentant l'homogénéité d'un échantillon soit intéressante, la subjectivité liée au choix de la région de rejet est problématique. Il est par

exemple maladroit de dire à partir d'une p-valeur qu'une séquence est « plus homogène » qu'une autre et de décider que l'une des p-valeurs est « suffisamment bonne » pour que la séquence soit considérée comme homogène.

L'objectif du prochain paragraphe est de proposer une solution alternative au problème de décision liée aux p-valeurs dans le cadre de l'homogénéité des modèles markoviens.

3.3.3 Algorithme d'analyse en ligne de l'homogénéité

Dans les standards proposés par les organismes de certification tels que le NIST ou le BSI, il est souligné l'importance de détecter le plus rapidement possible les défaillances d'un générateur. Pour ce faire il existe des outils, appelés tests en ligne, permettant de contrôler à la volée le bon fonctionnement d'un RNG.

Ainsi, les test que nous présentons dans la suite est construit avec une structure de test en ligne, qui va au fur et à mesure que la suite de nombres aléatoires est construite, déceler les défauts d'homogénéité et identifier les blocs homogènes.

De plus, la règle de décision n'est pas basée sur les p-valeurs, contrairement à l'algorithme 3.3.2, mais permet à l'utilisateur de choisir une région de rejet en fonction de l'application d'utilisation de l'aléa.

La définition d'une chaîne de Markov homogène est basée sur le comportement de la matrice de transition dans le temps. Ainsi afin d'évaluer l'homogénéité d'une séquence il est donc naturel d'étudier l'évolution des matrices de transition de ses sous-séquences.

On définit pour cela :

$$ME_{j,S} = \left(\frac{a_{ik}^{j} - a_{ik}^{S}}{a_{ik}^{S}} \right)_{0 \leqslant i,k \leqslant 2^M - 1},$$

la matrice des erreurs relatives entre A_S et A_{S_j}.

On définit également cinq normes, deux normes matricielles :

$$||R||_{1,\text{Mat}} = \max_{0 \leqslant j \leqslant 2^M - 1} \sum_{i=0}^{2^M - 1} |r_{ij}|,$$

$$||R||_{\infty,\text{Mat}} = \max_{0 \leqslant i \leqslant 2^M - 1} \sum_{j=0}^{2^M - 1} |r_{ij}|,$$

et trois normes vectorielles :

$$||R||_{1,V} = \sum_{i=0}^{2^M - 1} \sum_{j=0}^{2^M - 1} |r_{ij}|,$$

$$||R||_{2,V} = \sqrt{\sum_{i=0}^{2^M-1}\sum_{j=0}^{2^M-1} r_{ij}^2},$$
$$||R||_{\infty,V} = \max_{0\leqslant i,j\leqslant 2^M-1} |r_{ij}|.$$

L'algorithme se déroule en P étapes dans lesquelles on cherche à définir si la sous-séquence S_j, $j \in \{0,\ldots,P-1\}$ est homogène avec les sous-séquences qui la précèdent. En supposant qu'à l'étape j la séquence soit composée de i blocs homogènes $\mathcal{B}_1,\ldots,\mathcal{B}_i$, l'étape $j+1$ se déroule de la manière suivante :

1. On calcule $A_{\mathcal{B}_i}$ ainsi que $A_{S_{j+1}}$.
2. On déduit $ME_{j+1,\mathcal{B}_i}$ des deux matrices de transitions $A_{\mathcal{B}_i}$ et $A_{S_{j+1}}$.
3. On calcule les normes de cette matrice d'erreurs relatives.
   - Si on détecte un saut significatif pour une des normes on considère que la sous-séquence S_{j+1} ne fait pas partie du bloc homogène $\mathcal{B}_i$. On crée alors un nouveau bloc homogène $\mathcal{B}_{i+1}$ composé uniquement de la sous-séquence S_{j+1}.
   - Si on ne détecte pas d'anomalie, on intègre la sous-séquence S_{j+1} au bloc $\mathcal{B}_i$.
4. On passe à l'étape suivante.

En pratique on définit un seuil au-dessus duquel on estime que le bloc n'est plus homogène, ce seuil pouvant être adapté en fonction des applications pour lesquelles on utilise l'aléa. Pour les exemples qui suivent nous avons défini un seuil basé sur des simulations de séquences avec pour modèles des chaînes de Markov homogènes.

Plus précisément en effectuant les simulations et en calculant les normes des matrices d'erreurs relatives, nous avons observé que les courbes pour chaque norme ne dépassaient pas certaines valeurs, ce sont alors ces valeurs qui nous ont servi de seuil.

Pour illustrer l'algorithme nous allons détailler quelques étapes pour la séquence `CM_Rand_25_50_75` en considérant la norme $||.||_{1,\text{Mat}}$, $P = 100$ et un seuil égal à 0.8.

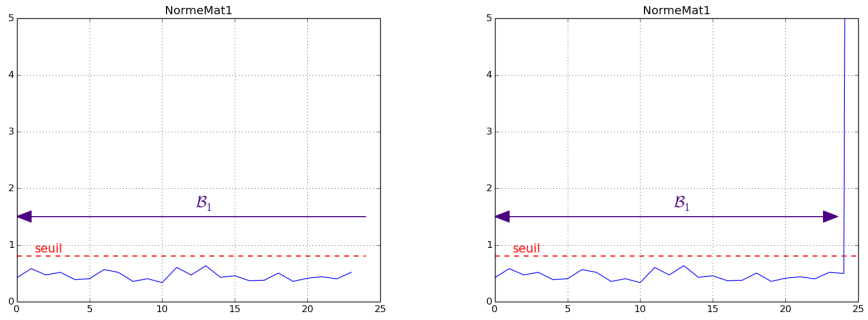


FIGURE 3.13: Étapes 23 et 25 de l'algorithme en ligne pour la séquence `CM_Rand_25_50_75`

Jusqu'à l'étape 23 on ne décèle aucune anomalie, le seuil fixé n'est jamais dépassé, on considère donc que toutes les sous-séquences font partie d'un même bloc $\mathcal{B}_1$. A l'inverse, à l'étape 25 la norme dépasse le seuil (la norme vaut 65.9). Par conséquent, on clôture le bloc $\mathcal{B}_1$ puis on crée un nouveau bloc à partir de la sous-séquence S_{25}.

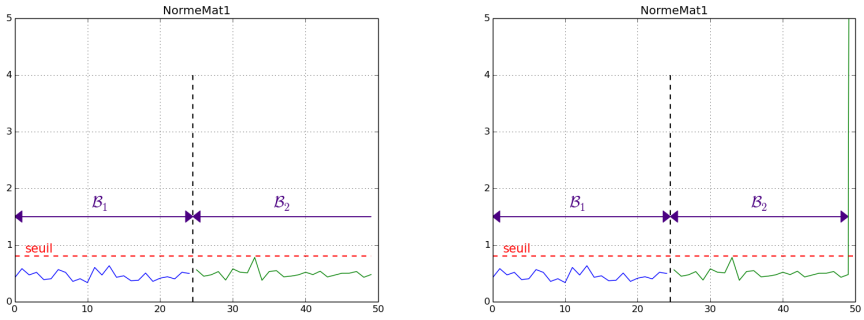


FIGURE 3.14: Étapes 49 et 50 de l'algorithme en ligne pour la séquence `CM_Rand_25_50_75`

Les sous-séquences S_{25} à S_{49} ont des normes en dessous du seuil et font donc partie du bloc $\mathcal{B}_2$. En revanche, une difformité est détectée à l'étape 50 (la norme vaut 49.8), ainsi l'algorithme construit le bloc $\mathcal{B}_3$ dont la sous-séquence S_{50} est la première composante. Après les 100 étapes on obtient un découpage comme suit :

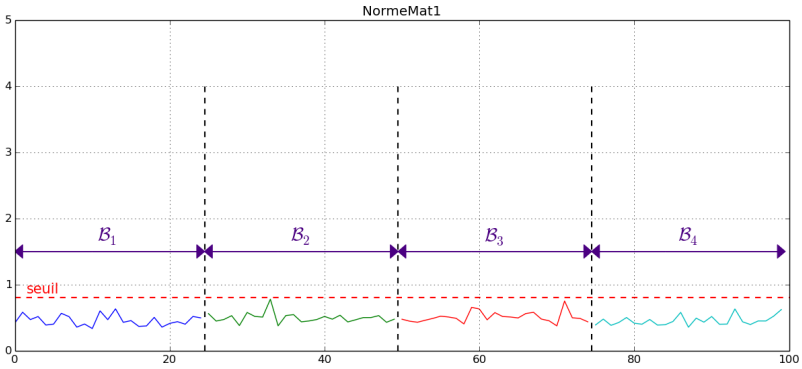


FIGURE 3.15: Résultat de l'algorithme en ligne pour la séquence `CM_Rand_25_50_75`

Le calcul de complexité montre que l'algorithme est moins coûteux que celui présenté dans le paragraphe 3.3.2.

	Nombre d'opérations
$+$	$P^2 + 3 \cdot 2^{2M} P + N$
$\times$	$2P$
$\div$	$P^2 + 2 \cdot 2^{2M} P + 1$

TABLE 3.8: Nombre d'opérations élémentaires pour l'algorithme en ligne.

Afin de comparer les complexités des deux algorithmes, on peut reprendre l'exemple présenté précédemment pour les paramètres $N = 80\,000\,000$, $P = 100$ et $M = 4$:

	Algorithme classique	Algorithme en ligne
$+$	240848844	42636800
$\times$	40014160	200
$\div$	4035	2611801

TABLE 3.9: Comparaison du nombre d'opérations élémentaires entre les deux algorithme de découpage.

3.3.4 Étude des différentes normes et des seuils associés

Le choix d'utiliser plusieurs normes repose sur le fait que chaque norme permet d'étudier des caractéristiques différentes des séquences de nombres et donc différentes propriétés du modèle basé sur les chaînes de Markov.

Afin d'illustrer les spécificités de chaque norme nous avons construit quatre séquences jouets dont la matrice de transition $A = (a_{ij})$ est modifiée à 50% de la séquence.

Nom	Modifications à 50%	Cibles
`CM_Mat1`	$a_{i\,0} = a_{i\,0} + \epsilon$ et $a_{i\,1} = a_{i\,1} - \epsilon$	Colonnes 0 et 1
`CM_MatInf`	$a_{0\,2j} = a_{0\,2j} + \epsilon$ et $a_{0\,2j+1} = a_{0\,2j+1} - \epsilon$	Ligne 0
`CM_Vec1`	$a_{i\,2j} = a_{i\,2j} + \epsilon$ et $a_{i\,2j+1} = a_{i\,2j+1} - \epsilon$	Tous les coefficients
`CM_VecInf`	$a_{0\,0} = a_{0\,0} + \epsilon$ et $a_{0\,1} = a_{0\,1} - \epsilon$	2 coefficients

TABLE 3.10: Construction de séquences jouets pour l'étude des différentes normes.

La norme matricielle $||R||_{1,\mathrm{Mat}}$ de part sa construction :

$$||R||_{1,\mathrm{Mat}} = \max_{0 \leqslant j \leqslant 2^M - 1} \sum_{i=0}^{2^M - 1} |r_{ij}|,$$

se focalise sur les coefficients d'une même colonne de la matrice R. Ainsi, dans le cadre d'une matrice de transition, on étudie les probabilités d'atteindre un état j.

Les résultats de l'algorithme pour la séquence `CM_Mat1` soulignent l'intérêt de cette norme.

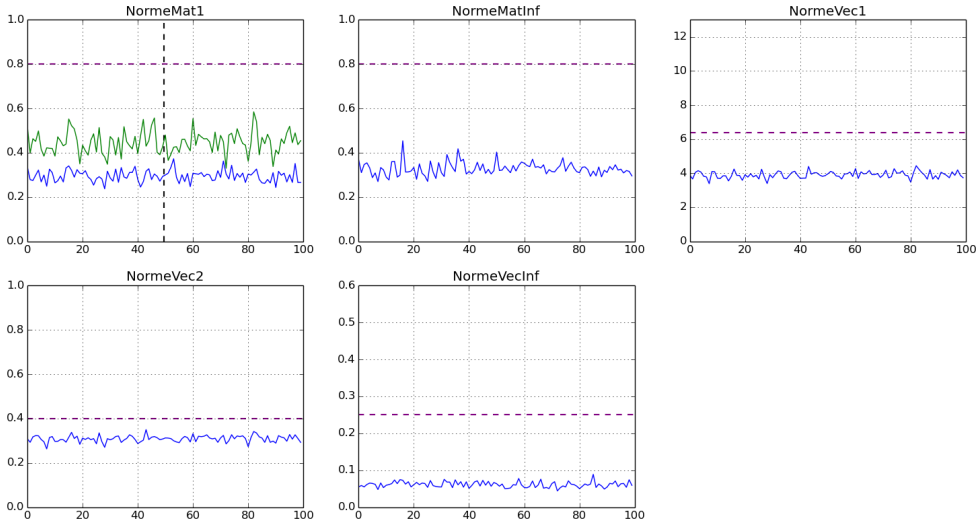


FIGURE 3.16: Analyse d'homogénéité de la séquence `CM_Mat1`.

En effet, la modification des probabilités d'atteindre les états $j = 0$ et $j = 1$ est déceler par la norme $||.||_{1,\text{Mat}}$, à l'inverse des autres normes qui considèrent la séquence comme un bloc homogène.

De façon analogue, la norme matricielle :

$$||R||_{\infty,\text{Mat}} = \max_{0 \leqslant i \leqslant 2^M - 1} \sum_{j=0}^{2^M-1} |r_{ij}|,$$

considère les coefficients d'une même ligne de la matrice R, ainsi elle caractérise, dans notre contexte d'utilisation, les probabilités de transition à partir d'un état i.

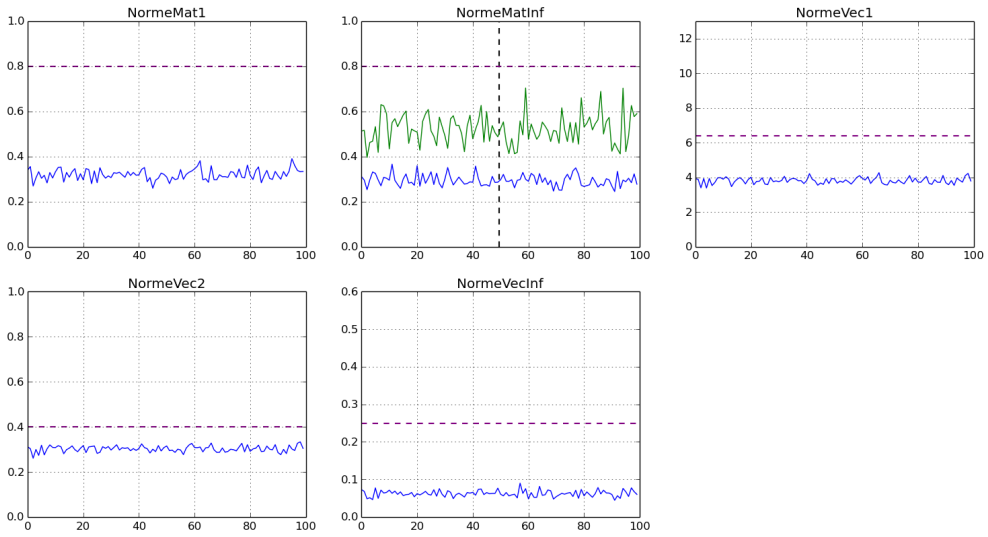


FIGURE 3.17: Analyse d'homogénéité de la séquence `CM_MatInf`.

La norme $||.||_{\infty,\text{Mat}}$ est ainsi la seule à détecter la modification de matrice de transition pour la séquence `CM_MatInf`.

Les normes vectorielles 1 et 2 sont, elles, construites de manière à examiner l'ensemble des coefficients d'une matrice :

$$||R||_{1,V} = \sum_{i=0}^{2^M-1} \sum_{j=0}^{2^M-1} |r_{ij}|, \qquad ||R||_{2,V} = \sqrt{\sum_{i=0}^{2^M-1} \sum_{j=0}^{2^M-1} r_{ij}^2}.$$

L'utilisation de ces normes pour l'algorithme de découpage permet l'analyse de déviations diffuses des matrices de transitions, par exemple, une légère modification de tous les coefficients comme pour la séquence `CM_Vec1`.

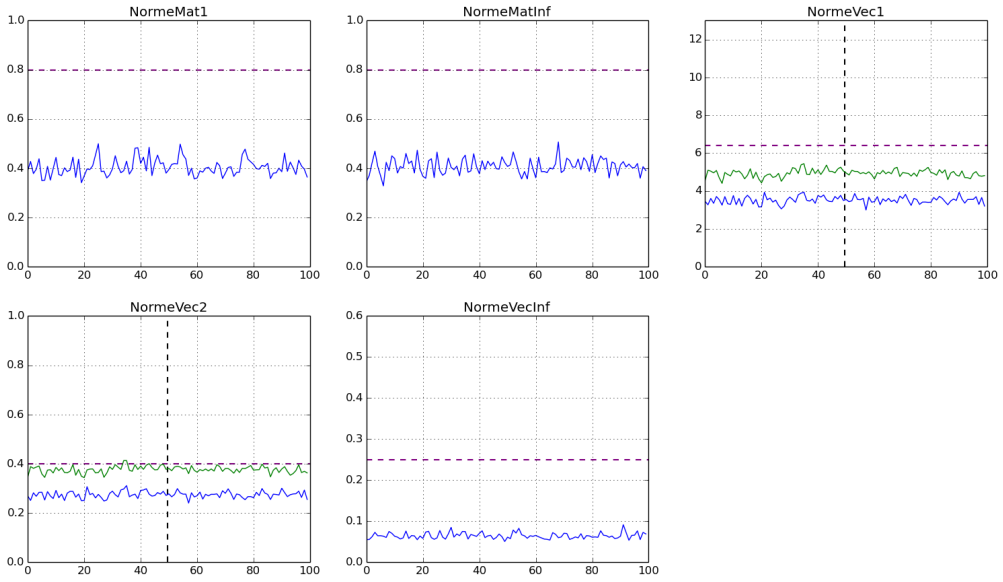


FIGURE 3.18: Analyse d'homogénéité de la séquence `CM_Vec1`.

Ces deux normes permettent de mettre en évidence la non-homogénéité de la séquence `CM_Vec1` et peuvent être considérées comme redondantes au vu de leur construction similaire. Cependant elles ont toutes les deux des avantages différents, la norme 2 ayant en pratique une détection des anomalies plus fine et la norme 1 ayant un coût de calcul plus faible ce qui peut être utile dans le cadre des tests en ligne.

Pour terminer, la norme vectorielle infinie permet de concentrer l'étude sur les coefficients de manière isolée :

$$||R||_{\infty,V} = \max_{0 \leqslant i,j \leqslant 2^M - 1} |r_{ij}|,$$

ce qui permet de déceler des défauts localisés sur des parties précises d'une matrice de transition, comme pour la séquence `CM_VecInf`.

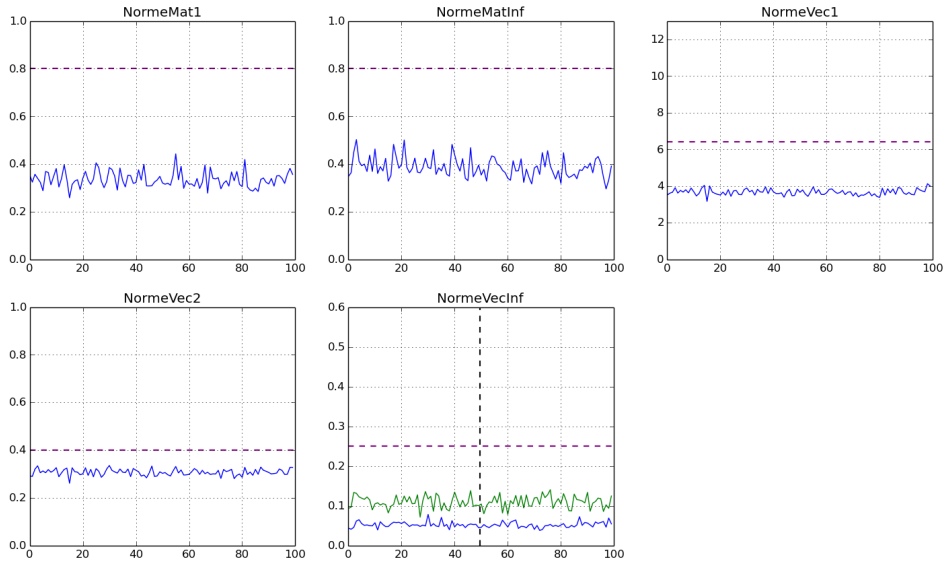


FIGURE 3.19: Analyse d'homogénéité de la séquence `CM_VecInf`.

Seuls deux coefficients ont été modifiés, ce qui n'est remarqué que par la norme $||.||_{\infty,V}$.

Ainsi, chaque norme permet d'analyser des aspects différents de la séquence, ce qui offre une étude plus précise de l'homogénéité. Cependant pour chaque norme il est nécessaire de définir un seuil de détection.

Le choix du seuil est variable en fonction de plusieurs critères. Le premier est le niveau de sécurité de l'utilisation de l'aléa, en effet plus l'application requiert un niveau de sécurité élevé plus le seuil sera bas, afin de garantir une stabilité importante des matrices de transition. Cependant, les erreurs de première et de deuxième espèce sont liées au choix du seuil, un seuil trop bas pourra ainsi créer un nombre trop important de fausses alarmes (erreur de première espèce).

Ensuite, le seuil s'adapte aux connaissances théorique du NDRBG, par exemple si les coefficients de la matrice de transition sont contraints, par le modèle, à des valeurs $\frac{1}{2^M} \pm \epsilon$, le seuil sera alors adapté en fonction.

Il est également possible de définir les seuils en fonction du pourcentage d'écart que l'on tolère entre les coefficients des matrices de transition successives de la séquence. Pour la norme $||.||_{1,V}$, si l'écart maximal accepté est de 2.5%, le seuil est alors de :

$$\sum_{i=0}^{2^M-1} \sum_{j=0}^{2^M-1} 0.025 = 2^{2M} \times 0.025,$$

ce qui correspond, pour des motifs de taille 4, à un seuil de 6.4.

Enfin, de manière empirique, il est possible de déterminer des bornes pour le choix du seuil pour une taille de séquence fixée. En effet les résultats de test sur un nombre important de séquences basées sur des chaînes de Markov homogènes permettent de donner une intuition sur les seuils réalistes.

3.3.5 Résultats sur les séquences jouets

Notons $\mathcal{B}(S_k)$ le bloc homogène dans lequel figure la sous-séquence S_k. On peut alors afficher la différence entre $||ME_{k,S}||$ (en vert sur le graphique) et $||ME_{k,\mathcal{B}(S_k)}||$ (en bleu sur le graphique), ainsi que les différents blocs que l'algorithme a détecté (en noir sur le graphique) :

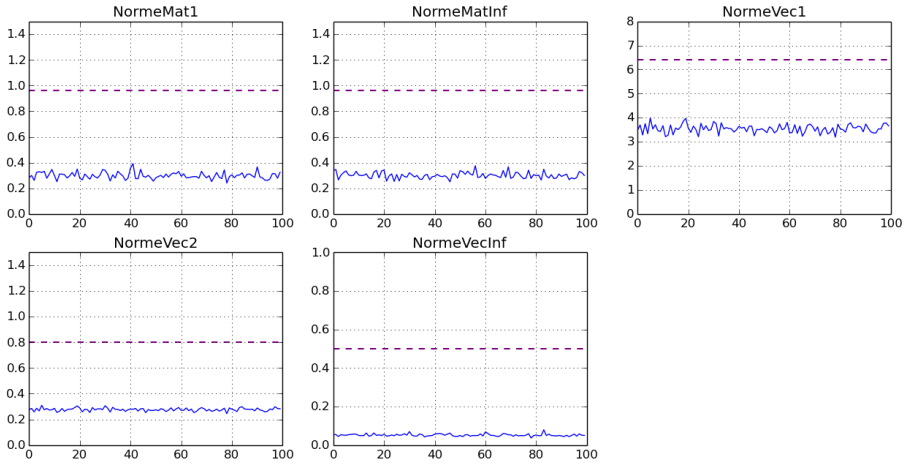


FIGURE 3.20: Évolution des matrices d'erreurs relatives pour la séquence `CM_Uni` avant et après découpage

Sur cette séquence qui sert de référence la courbe bleue et la courbe verte sont confondues car il n'y a qu'un seul bloc détecté par l'algorithme.

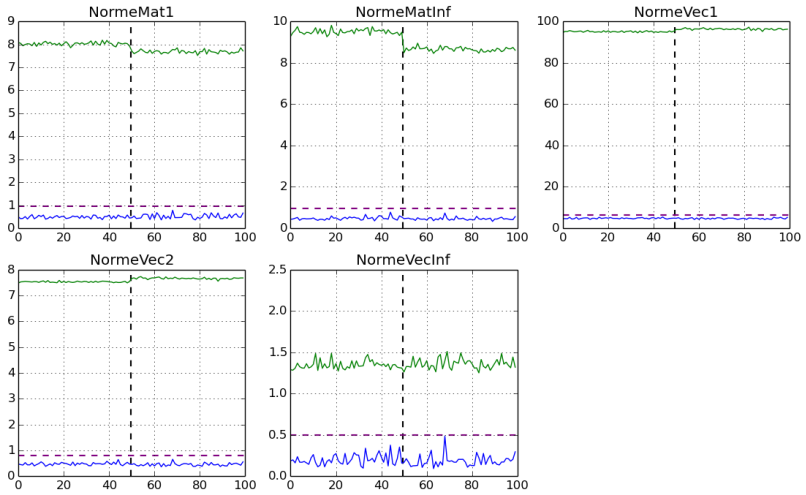


FIGURE 3.21: Évolution des matrices d'erreurs relatives pour la séquence `CM_Rand_50` avant et après découpage

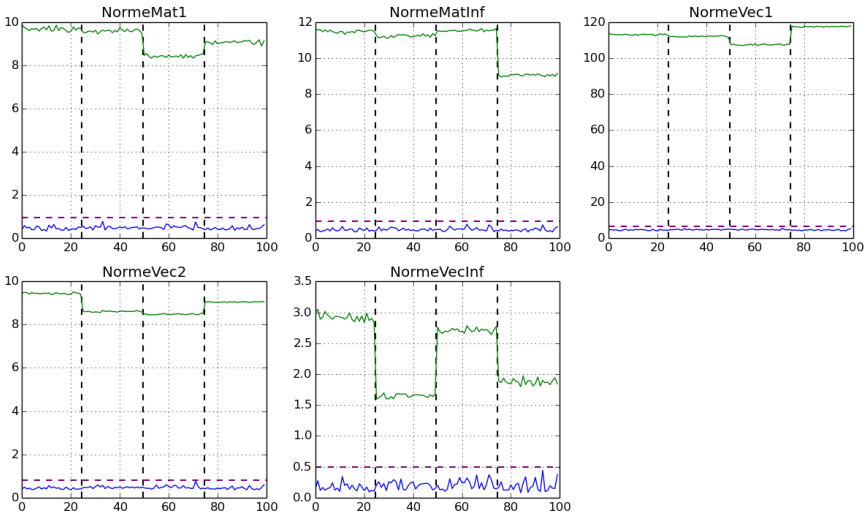


FIGURE 3.22: Évolution des matrices d'erreurs relatives pour la séquence `CM_Rand_25_50_75` avant et après découpage

Sur les deux séquences non homogènes, l'algorithme détecte les différents blocs homogènes ; deux pour la séquence `CM_Rand_50` et quatre pour la séquence `CM_Rand_25_50_75`.

Les résultats confirment également que toutes les normes ne détectent pas les anomalies de la même manière comme nous l'avons montré dans le paragraphe précédent. Ainsi chaque norme apporte des informations différentes sur la qualité et les défauts des séquences et peuvent être utiles pour différentes applications.

3.4 Analyse de générateurs

Dans cette partie nous présentons les résultats de tests présentés ci-dessus effectués sur des séquences issues de générateurs physiques ainsi que de générateurs pseudo aléatoires. Pour chaque générateur nous nous intéressons aux sous-séquences homogènes calculées par l'algorithme de découpage de la section 3.2 que nous appellerons « classique » ainsi que l'algorithme de découpage présenté dans la section 3.3.3 que nous appellerons « en ligne ». Pour les algorithmes de la section 3.2 nous étudierons également les p-valeurs associées aux blocs homogènes, la région des p-valeurs acceptées sera $[10^{-3}, 1-10^{-3}]$.

3.4.1 Générateurs pseudo-aléatoires

Nous présentons les résultats de séquences de 40Mo issues de trois générateurs pseudo-aléatoires et de deux LFSR ; la fonction `Rand` d'Openssl [Tea15a], la fonction `random` de Pari/GP [Gro15], un générateur basé sur la fonction de hachage SHA1 ainsi que deux LFSR de taille 8 et 32.

Résultats de l'algorithme classique :

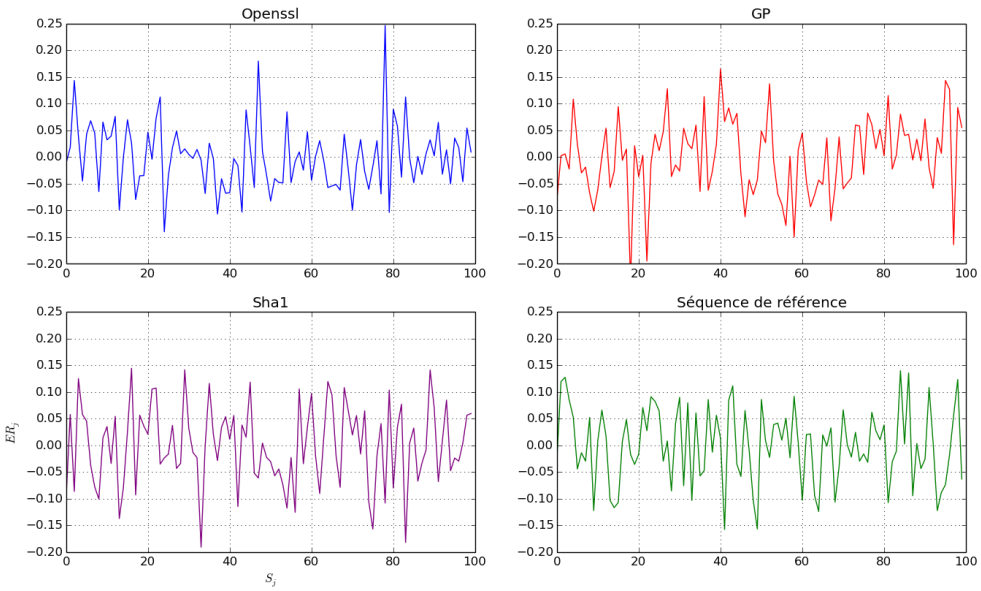


FIGURE 3.23: Écart relatif entre les sous-séquences pour les générateurs déterministes

Séquence	Nombre de sous-séquences	p-valeur
`Openssl`	1	0.571
`GP`	1	0.242
`Sha1`	1	0.119
`LFSR_8`	100	-
`LFSR_32`	100	-

TABLE 3.11: Découpage des séquences pseudo-aléatoire par l'algorithme classique.

Résultats de l'algorithme en ligne :

Pour les trois générateurs pseudo-aléatoires, aucune norme ne détecte d'anomalie et la séquence ne comporte qu'un seul bloc. En revanche pour les LFSR le seuil n'est pas dépassé mais les séquences montre une trop faible variation des matrices de transition, ce qui représente un défaut d'homogénéité.

Voici les graphiques correspondants pour les séquences `LFSR_8` et `Openssl` :

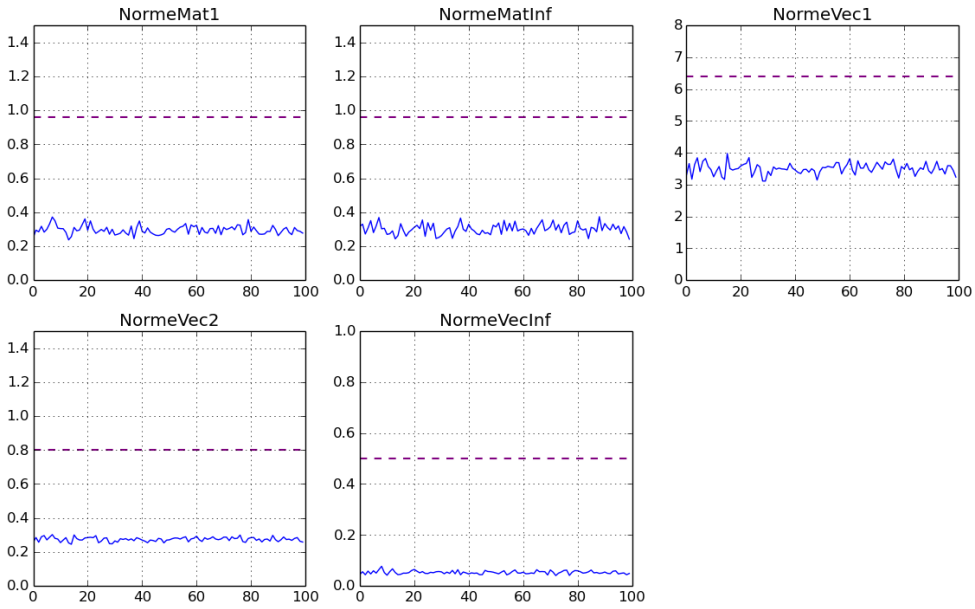


FIGURE 3.24: Évolution des matrices d'erreurs relatives pour la séquence Openssl avant et après découpage

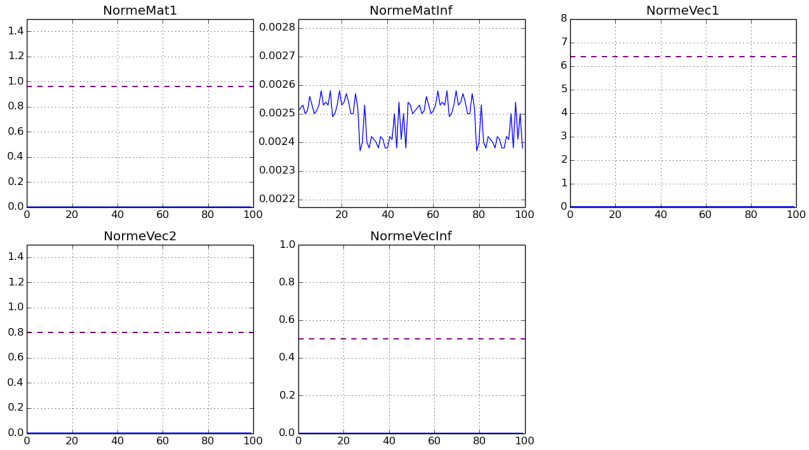


FIGURE 3.25: Évolution des matrices d'erreurs relatives pour la séquence LFSR_8 avant et après découpage

Un zoom a été effectué pour la norme $||.||_{\infty,\text{Mat}}$ afin de pouvoir observer les structures se répétant dans les graphiques. Pour les autres normes l'échelle a été conservée pour permettre de comparer ces résultats à ceux des autres séquences.

Bilan :

Les tests ne détectent pas d'anomalies pour les PRNG en revanche les LFSR montrent des défauts significatifs.

En effet, les normes des séquences `LFSR_8` et `LFSR_32` sont extrêmement faibles, de manière intuitive, ces séquences sont « trop » homogènes. Pour être plus précis, c'est le caractère prédictible, déterministe des LFSR qui est mis en évidence par le test. La taille de la période du générateur joue un rôle prépondérant dans le résultat mais on détecte une anomalie même pour des tailles supérieures à 2^{32}.

3.4.2 Générateurs sans défaut significatifs

Dans ce paragraphe, nous présentons l'étude sur plusieurs séquences issues de différents générateurs physiques, dont les résultats ne nous permettent pas de déceler d'anomalies d'homogénéité. L'analyse se concentre sur :

— Trois séquences de 40 Mo du générateur Intel [Cor14] présent sur les processeur Ivy Bridge, générées avec trois modes différents : le mode 16, le mode 32 et le mode 64 bits.

— Une séquence de 40 Mo du générateur Aléa II de l'entreprise Finlandaise Araneus [Ara].

— Une séquence de 40 Mo du générateur Quantis. Ce générateur utilise les propriétés quantiques des photons pour produire des séquences de nombres aléatoires, pour plus de détails sur le principe ou l'étude du générateur, on peut se référer aux travaux de M. Soucarros [Sou12].

— Six séquences de 40 Mo issues de trois générateurs présents dans les tablettes numériques Ipad : DevRandom, Ossl et SecRandom. Pour chaque générateur nous étudions une séquence générée en conditions normales et une séquence générée en conditions perturbées.

Résultats de l'algorithme classique :

Générateur	Séquence	Nombre de sous-séquences	p-valeur
Intel	Int_16	1	0.462
	Int_32	1	0.935
	Int_64	1	0.621
Aranéus	Ara_1	1	0.200
Quantis	Qua_1	1	0.452
Ipad	Ip_Dev	1	0.760
	Ip_Dev_Stress	1	0.964
	Ip_Ossl	1	0.472
	Ip_Ossl_Stress	1	0.057
	Ip_Sec	1	0.886
	Ip_Sec_Stress	1	0.960

TABLE 3.12: Découpage des séquences sans défaut significatifs par l'algorithme classique.

Pour l'analyse graphique nous ne présentons que le générateur Intel, cependant pour les autres séquences les résultats sont similaires.

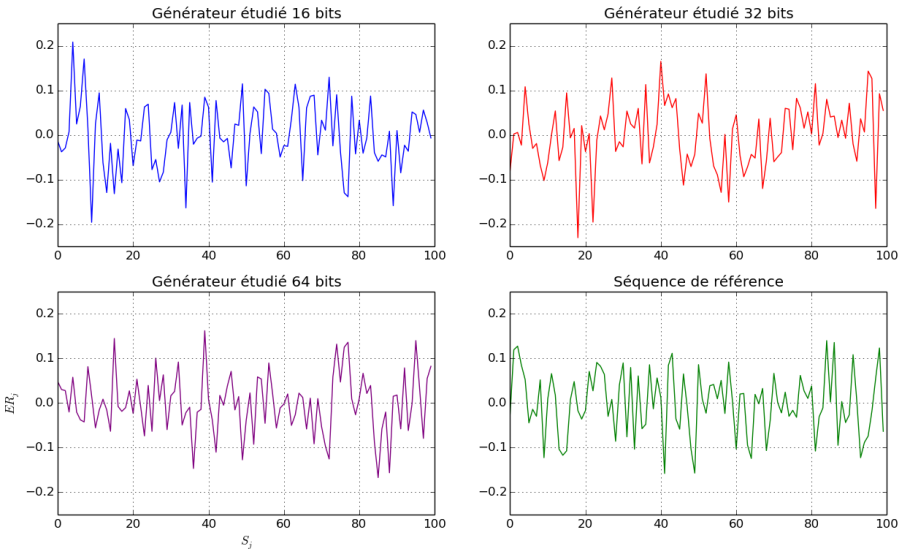


FIGURE 3.26: Écart relatif entre les sous-séquences pour le générateur Intel 16, 32 et 64 bits

Résultats de l'algorithme en ligne :

Pour toutes les séquences testées, les normes détectent un seul bloc et ne mettent pas en évidence un défaut d'homogénéité.

Voici le graphique correspondant pour la séquence Qua_1 du générateur Quantis :

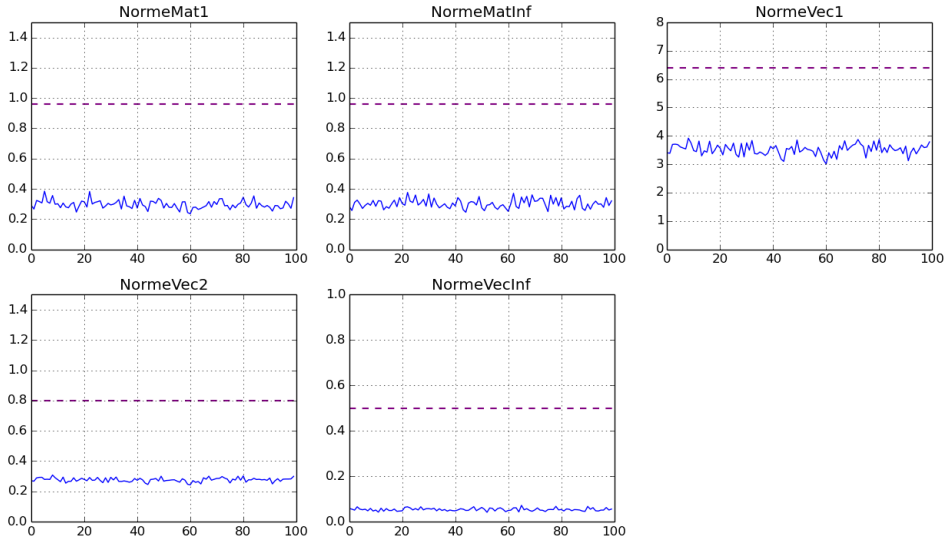


FIGURE 3.27: Évolution des matrices d'erreurs relatives pour la séquence Qua_1 avant et après découpage

Bilan :

Pour ces quatre générateurs, les deux algorithmes ne détectent pas de défaut dans l'homogénéité des séquences. En ce qui concerne l'Ipad, la perturbation appliquée aux trois générateurs ne modifie pas les résultats, tout comme le mode de génération des séquences Intel qui n'influe pas sur l'homogénéité.

3.4.3 ViaNano

Nous présentons deux séquences de 40 Mo issues du TRNG implanté dans un processeur Via Nano, soumis pendant les acquisitions à une température de 90 degrés Celsius.

Résultats de l'algorithme classique :

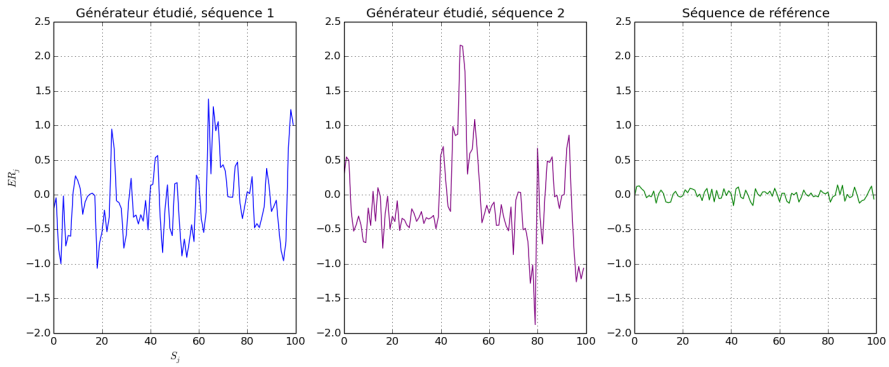


FIGURE 3.28: Écart relatif entre les sous-séquence pour le générateur ViaNano

Séquence	Nombre de sous-séquences	p-valeur
ViaNano1	84	-
ViaNano2	69	-

TABLE 3.13: Découpage des séquences ViaNano par l'algorithme classique.

Résultats de l'algorithme en ligne :

Séquence	Nombre de sous-séquences				
	$\|\|.\|\|_{1,M}$	$\|\|.\|\|_{\infty,M}$	$\|\|.\|\|_{1,V}$	$\|\|.\|\|_{2,V}$	$\|\|.\|\|_{\infty,V}$
ViaNano 1	52	45	67	46	6
ViaNano 2	56	53	74	56	8

TABLE 3.14: Découpage des séquences ViaNano par l'algorithme en ligne.

Voici le graphique correspondant à la séquence ViaNano 2 :

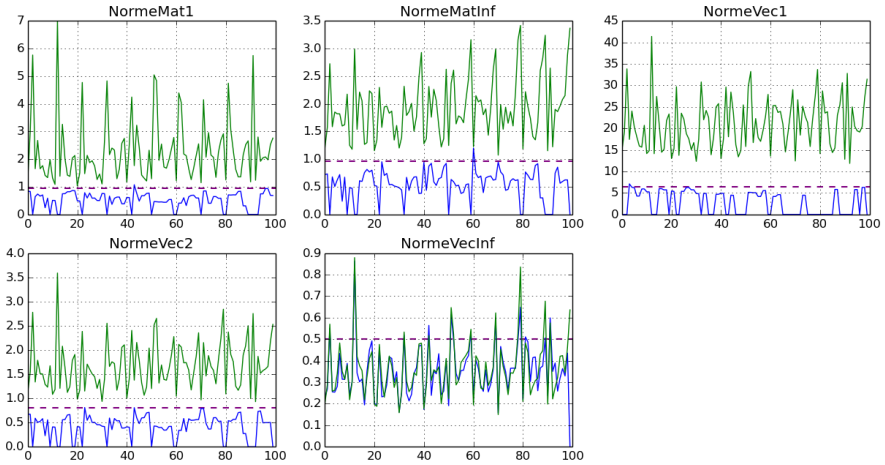


FIGURE 3.29: Évolution des matrices d'erreurs relatives pour la séquence ViaNano 2 avant et après découpage

Bilan :

Les deux algorithmes montrent pour les deux séquences des défauts importants concernant l'homogénéité en témoignent les graphiques et les p-valeurs. De plus, en observant la figure 3.29 on distingue une forme de périodicité dans les irrégularités des valeurs. On peut alors penser que des sous-séquences extraites de la séquence pourraient être de meilleure qualité.

En effet, si on prend l'exemple de la séquence extraite composée des sous-séquence S_{10i} avec $i \in \{0, \ldots, 9\}$ alors le test en ligne détecte un unique bloc homogène, en témoigne la norme vectorielle infinie présentée dans la Fig. 3.30.

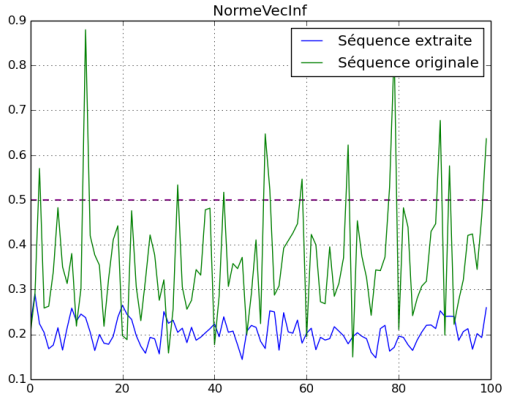


FIGURE 3.30: Résultats de la séquence ViaNano 2 et de la séquence extraite pour la norme vectorielle infinie.

3.4.4 Cyclone

Nous présentons trois séquences de 40 Mo issues du générateur asynchrone Cyclone (64, 128 et 256 étages) dont le schéma théorique est explicité dans la figure 2.8.

Résultats de l'algorithme en classique :

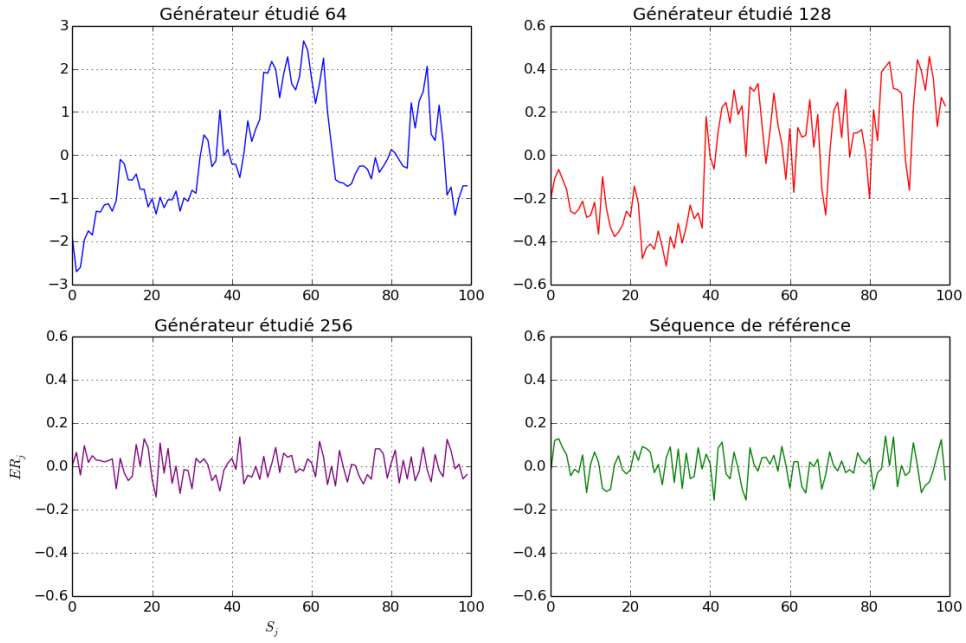


FIGURE 3.31: Écart relatif entre les sous-séquences pour le générateur Cyclone 64, 128 et 256 étages

Séquence	Nombre de sous-séquences	p-valeur
Cyclone64	96	-
Cyclone128	29	-
Cyclone256	1	0.112

TABLE 3.15: Découpage des séquences Cyclone par l'algorithme classique.

Résultats de l'algorithme en ligne :

Séquence	Nombre de sous-séquences				
	$\|\|.\|\|_{1,M}$	$\|\|.\|\|_{\infty,M}$	$\|\|.\|\|_{1,V}$	$\|\|.\|\|_{2,V}$	$\|\|.\|\|_{\infty,V}$
Cyclone 64	15	16	28	16	3
Cyclone 128	4	2	4	2	1
Cyclone 256	2	1	2	1	1

TABLE 3.16: Découpage des séquences Cyclone par l'algorithme en ligne.

Voici les graphiques correspondants aux séquences Cyclone 128 et Cyclone 256 :

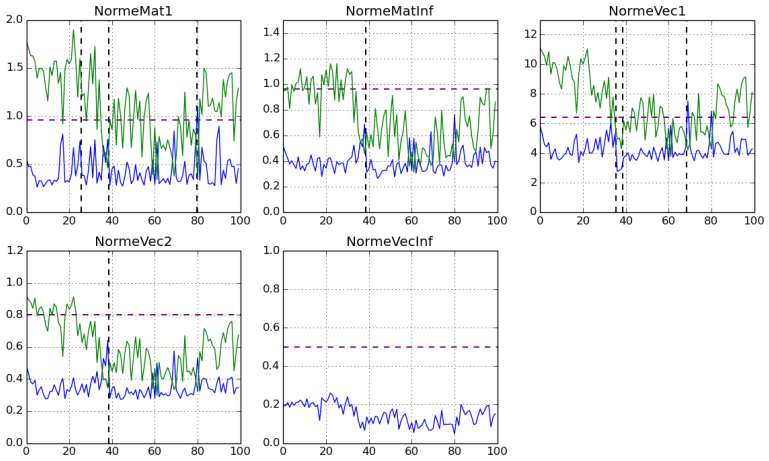


FIGURE 3.32: Évolution des matrices d'erreurs relatives pour la séquence Cyclone 128 avant et après découpage

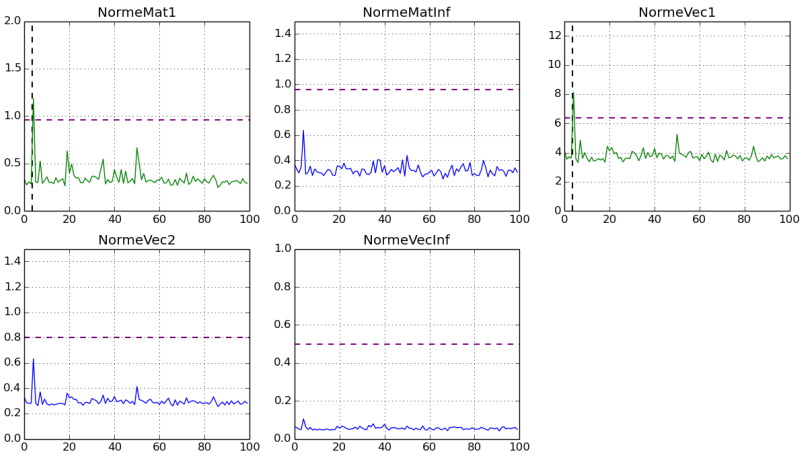


FIGURE 3.33: Évolution des matrices d'erreurs relatives pour la séquence Cyclone 256 avant et après découpage

Bilan :

Les résultats des deux algorithmes montrent que plus le nombre d'étages est important moins les séquences ont de défauts d'homogénéité. En revanche l'algorithme en ligne met en évidence une anomalie en début de séquence pour la version 256 étages du générateur. Cela peut être imputable à une phase d'initialisation du générateur.

3.4.5 Comparatifs de deux générateurs industriels

Nous présentons six séquences de 1 Mo issues de deux générateurs industriels (notés A et B) ; l'un basé sur les anneaux d'oscillateurs, l'autre sur des effets de transitions dans les anneaux d'oscillateurs (TERO). Pour notre étude nous n'avions pas connaissance de la technologie associée à chaque générateur, ainsi nous nous focalisons sur deux objectifs :

— Analyser le comportement des séquences dans le temps.

— Distinguer la technologie associée à chaque générateur.

Pour ce faire nous expliciterons également les résultats des batteries `Alphabit` et `Rabbit` de la suite de tests U01.

Résultats de l'algorithme en classique :

Pour les résultats des deux batteries issues des tests U01, nous présentons le pourcentage de tests réussis.

Séquence	Sous-séquences	p-valeur	Alphabit (% réussis)	Rabbit (% reussis)
`RNG_A_1`	1	0.112	0%	15%
`RNG_A_2`	1	0.414	100%	100%
`RNG_A_3`	1	0.819	100%	100%
`RNG_A_4`	13	-	0%	15%
`RNG_B_5`	19	-	6%	23%
`RNG_B_6`	1	0.152	71%	82%

Trois comportements différents apparaissent dans les résultats :

— Les séquences `RNG_A_2` et `RNG_A_3` réussissent les tests des batteries Alphabit et Rabbit ainsi que le test d'homogénéité. Aucune anomalie statistique n'est donc détectée.

— Les séquences `RNG_A_4` et `RNG_B_5` échouent aux tests des batteries et l'algorithme de découpage met en évidence plus de 10 blocs différents pour chacune des séquences. Ainsi on peut conclure que ces deux séquences ne sont pas homogènes.

— Les séquences `RNG_A_1` et `RNG_B_6` pour lesquelles l'algorithme de découpage ne montre pas de défaut d'homogénéité alors que les résultats des batteries révèlent un problème statistique. Il faut donc poursuivre l'étude pour comprendre l'origine de cette anomalie.

Résultats de l'algorithme en ligne :

Compte tenu de la faible quantité de données disponible pour chaque séquence, nous avons redéfinis les seuils de détection.

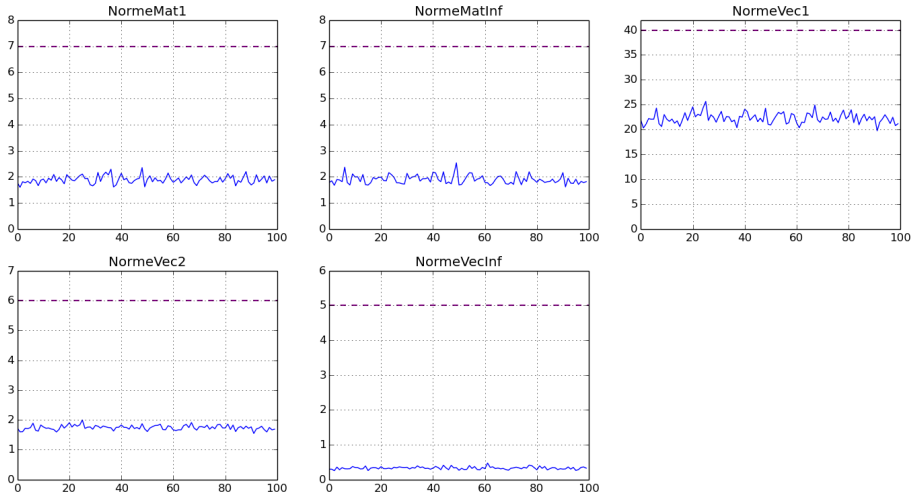


FIGURE 3.34: Évolution des matrices d'erreurs relatives pour la séquence `RNG_A_3` avant et après découpage

Comme pour les tests précédents, aucun défaut n'est détecté pour cette séquence.

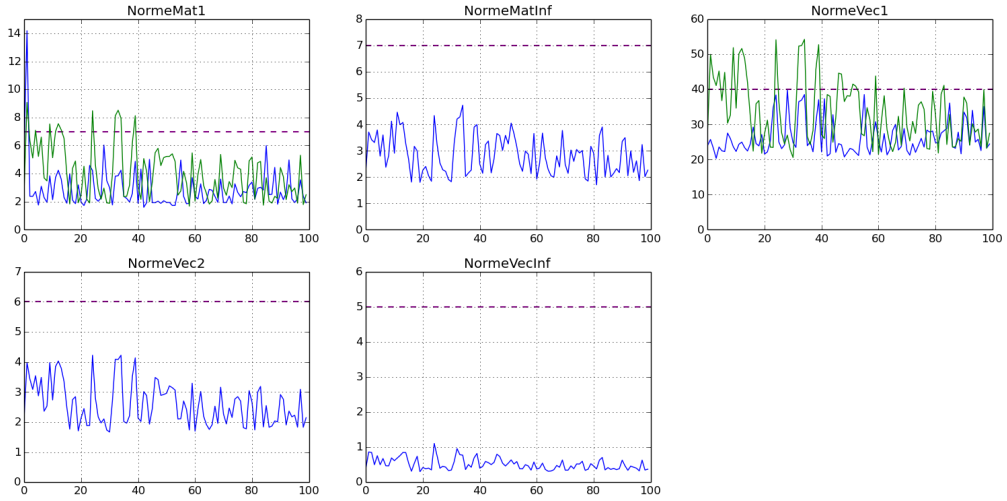


FIGURE 3.35: Évolution des matrices d'erreurs relatives pour la séquence `RNG_A_4` avant et après découpage

Les résultats de l'algorithme en ligne viennent confirmer le comportement non-homogène de cette séquence avec une perturbation plus accentuée au début de la séquence.

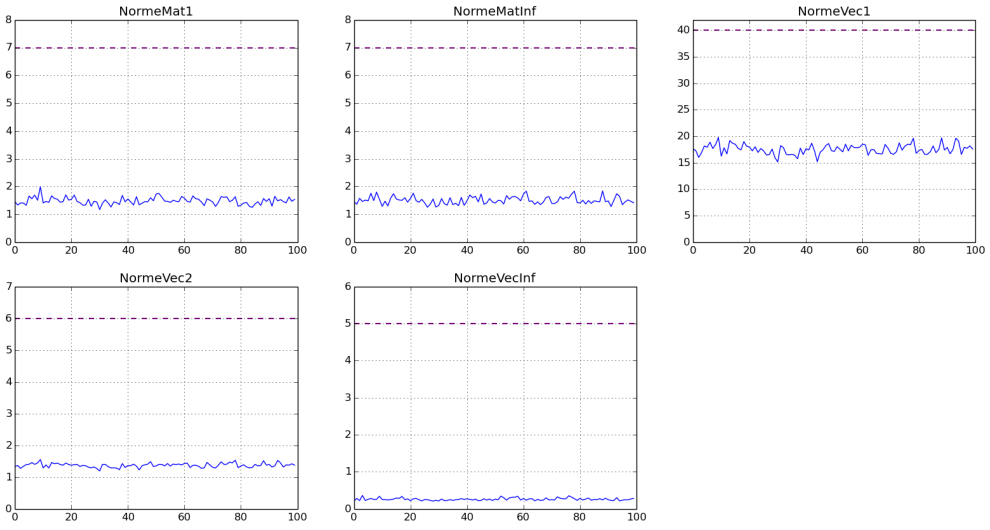


FIGURE 3.36: Évolution des matrices d'erreurs relatives pour la séquence `RNG_B_6` avant et après découpage

Les séquences `RNG_A_1` et `RNG_B_6` possèdent un défaut d'indépendance des motifs, mis en évidence par les batteries Alphabit et Rabbit, le modèle des chaînes de Markov intégrant

la dépendance, il est donc cohérent que l'algorithme en ligne ne montre pas de défaut d'homogénéité.

Étude des séquences 4 et 5 après découpage

Il est possible d'extraire par l'algorithme de découpage des blocs homogènes et de les étudier avec les outils classiques. Pour la séquence `RNG_A_4` le plus grand bloc homogène est constitué des sous-séquences S_{80} à S_{99} et pour la séquence `RNG_B_5` des sous-séquences S_{32} à S_{66}, voici les résultats des batteries Alphabit et Rabbit sur ces deux blocs.

Séquence	Alphabit (% réussis)	Rabbit (% reussis)
Bloc homogène `RNG_A_4`	0%	24%
Bloc homogène `RNG_B_5`	6%	34%

Les pourcentages de réussite pour la batterie Alphabit sont identiques sur la séquence entière et sur les blocs homogènes cependant pour la batterie Rabbit les résultats sont meilleurs pour la sous-séquence homogène extraite de chaque séquence. Ainsi on remarque que l'algorithme permet d'extraire une séquence avec de meilleures propriétés statistiques mais on voit également que des défauts persistent.

Bilan :

Nous avons pu analyser par les tests le comportement des séquences dans le temps et corroborer ces résultats avec les batteries des tests U01. Nous avons également pu distinguer des sous-séquences de meilleure qualité à l'intérieur des séquences étudiées. En revanche, les différents algorithmes ne nous permettent pas de distinguer les deux générateurs.

3.5 Conclusion

Au cours de ce chapitre nous avons proposé des outils d'analyse de générateurs basés sur les chaînes de Markov.

Dans un premier temps, nous avons développé des tests statistiques correspondant au schéma des standards actuels avec pour hypothèse nulle des modèles Markoviens. Ces tests peuvent ainsi être intégrés facilement à des batteries ou des normes comme celles du NIST ou du BSI.

Dans un deuxième temps, nous avons présenté un algorithme de découpage d'une séquence en blocs homogènes. La vérification de l'homogénéité d'une séquence de nombres aléatoires est importante, particulièrement avant l'exécution de nombreux tests (comme celui de Maurer) où l'hypothèse se base sur des chaînes de Markov homogènes. Il est donc intéressant d'utiliser cet outil en amont de l'exécution des batteries de tests statistiques.

Dans un troisième temps, nous avons exhibé les résultats des tests et algorithmes présentés sur des séquences issues de divers générateurs. Lorsque les résultats sont positifs, la méthodologie d'analyse classique peut être utilisée. En revanche lorsque les résultats sont négatifs les outils pour étudier les modèles Markoviens non homogènes sont peu nombreux. Nous allons donc, dans la suite, nous intéresser à ce type de modèles.

Chapitre 4

Modèles de Markov cachés

Sommaire

Dans la section 3.1 nous avons présenté la notion de chaîne de Markov homogène et celle de chaîne de Markov homogène par morceaux. Les modèles de Markov cachés que nous allons présenter dans cette section peuvent être vus comme des chaînes de Markov homogènes par morceaux, la répartition des morceaux étant aléatoire.

4.1 Introduction

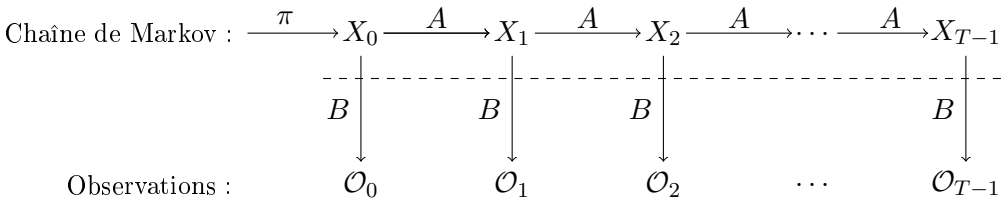
Les modèles de Markov cachés ont été développés en partie dans les années 1960 par les travaux de Leonard E. Baum [BP66], [BE67] et Ruslan L. Stratonovich [Str60].

Définition 4.1.1. *Un modèle de Markov caché est un modèle statistique dans lequel les états ne sont pas directement observables. On considère alors un système dont les états sont décrits par une chaîne de Markov homogène à espace d'états fini et à espace de temps discret X_k, $k \in \mathbb{N}$ ainsi qu'une suite d'observations $\mathcal{O}_k$, $k \in \mathbb{N}$ qui dépendent des états de la chaîne de Markov.*

Notations :

- $N =$ nombre d'états dans le modèle.
- $Q = \{q_0, q_1, \ldots, q_{N-1}\} =$ les états distincts de la chaîne de Markov.
- $A =$ matrice de transition de la chaîne de Markov.
- $\pi = (\pi_1, \ldots, \pi_N) =$ distribution initiale de la chaîne.
- $M =$ nombre de symboles observables.
- $V = \{0, 1, \ldots, M-1\} =$ l'ensemble des observations possibles.
- $T =$ taille de la séquence observée.
- $\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1}) =$ séquences des observations.
- $B =$ matrice des probabilités d'observations.

Principe :



Un modèle de Markov caché est donc caractérisé par A, B et π (M et N étant implicitement définis par A et B) et on le note $\lambda = (\pi, A, B)$.

Pour illustrer cette famille de modèles, on va considérer un problème de dendrochronologie qui consiste à étudier les changements de températures grâce aux cernes des arbres.

En effet, il est réaliste de supposer que la taille des cernes des arbres dépend de la température annuelle. Les tailles des cernes successives (petites, moyennes ou grandes) sont les observations et les températures annuelles (froides ou chaudes) sont les variables cachées. Ainsi on propose le modèle suivant :

$$\pi = \begin{matrix} C & F \end{matrix} \begin{pmatrix} 0.6 & 0.4 \end{pmatrix}, \qquad A = \begin{matrix} \\ C \\ F \end{matrix} \begin{matrix} C & F \end{matrix} \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}, \qquad B = \begin{matrix} \\ C \\ F \end{matrix} \begin{matrix} P & M & G \end{matrix} \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.7 & 0.2 & 0.1 \end{pmatrix},$$

dans lequel par exemple la probabilité de deux années chaudes consécutives vaut 0.7 et la probabilité d'un cerne moyen sachant que l'année était froide est de 0.2.

On peut définir trois problèmes concernant les modèles de Markov cachés.

Problème 1 : Connaissant le modèle $\lambda = (\pi, A, B)$ et les observations $\mathcal{O}$, on cherche à calculer la probabilité que ces observations correspondent au modèle, $P(\mathcal{O}|\lambda)$.

Dans le cadre de la dendrochronologie, le problème s'applique par exemple à la probabilité de l'observation $\mathcal{O} = (G, P)$ connaissant le modèle :

$$\begin{aligned} P(\mathcal{O}|\lambda) &= \sum_X P(\mathcal{O} = GP, X|\lambda), \\ &= \pi_F B_{FG} A_{FF} B_{FP} + \pi_F B_{FG} A_{FC} B_{CP} + \pi_C B_{CG} A_{CF} B_{FP} + \pi_C B_{CG} A_{CC} B_{CP}, \\ &= 0.1072. \end{aligned}$$

Problème 2 : Connaissant le modèle $\lambda = (\pi, A, B)$ et les observations $\mathcal{O}$, on cherche à calculer la séquence d'états optimale correspondant à la chaîne de Markov cachée.

Pour ce problème, il faut remarquer que la réponse n'est pas unique, elle dépend de la définition du terme « optimale ». En effet la solution sera différente si on cherche à optimiser la probabilité de chaque état indépendamment (point de vue des modèles de Markov cachés) ou si on optimise la probabilité de tous les états vus comme une chaîne (point de vue de la programmation dynamique).

Pour notre modèle exemple, il est intéressant de calculer la séquence d'états optimale pour les observations $\mathcal{O} = (P, M, P, G)$.

Pour le point de vue de la programmation dynamique on doit calculer pour toute séquence d'états X le probabilité $P(X, \mathcal{O}|\lambda)$. Par exemple pour la séquence $X = CFFC$ le calcul

est :

$$\begin{aligned} P(X = CFFC,\ \mathcal{O} = PMPG|\lambda) &= \pi_C B_{CP} A_{CF} B_{FM} A_{FF} B_{FP} A_{FC} B_{CG}, \\ &= 0.6 \cdot 0.1 \cdot 0.3 \cdot 0.2 \cdot 0.6 \cdot 0.7 \cdot 0.4 \cdot 0.5, \\ &= 3.024 \cdot 10^{-4}. \end{aligned}$$

On construit ainsi le tableau suivant :

États	$P(X, \mathcal{O}\|\lambda)$	Probabilité Normalisée
CCCC	$4.116 \cdot 10^{-4}$	0.0427
CCCF	$3.528 \cdot 10^{-5}$	0.0037
CCFC	$7.056 \cdot 10^{-4}$	0.0733
CCFF	$2.117 \cdot 10^{-4}$	0.0220
CFCC	$5.040 \cdot 10^{-5}$	0.0052
CFCF	$4.320 \cdot 10^{-6}$	0.0004
CFFC	$3.024 \cdot 10^{-4}$	0.0314
CFFF	$9.072 \cdot 10^{-5}$	0.0094
FCCC	$1.098 \cdot 10^{-3}$	0.1140
FCCF	$9.408 \cdot 10^{-5}$	0.0098
FCFC	$1.882 \cdot 10^{-3}$	0.1954
FCFF	$5.645 \cdot 10^{-4}$	0.0586
FFCC	$4.704 \cdot 10^{-4}$	0.0489
FFCF	$4.032 \cdot 10^{-5}$	0.0042
FFFC	$2.822 \cdot 10^{-3}$	0.2931
FFFF	$8.467 \cdot 10^{-4}$	0.0879

TABLE 4.1: Calcul des probabilités selon le point de vue programmation dynamique.

La chaîne dont la probabilité est maximale pour ce point de vue est FFFC.

En ce qui concerne le point de vue des chaînes de Markov cachés il est nécessaire de calculer le symbole le plus probable à chaque position. Grâce aux probabilités normalisées du tableau précédent il suffit, pour calculer la probabilité d'avoir le symbole q à la position i, de faire la somme des probabilités des chaînes contenant q en i-ème position.

Symbole	Position 1	Position 2	Position 3	Position 4
C	0.1881	0.5195	0.2289	0.8040
F	0.8119	0.4805	0.7711	0.1960

TABLE 4.2: Calcul des probabilités selon le modèle des chaînes de Markov.

La chaîne des symboles les plus probables à chaque position est alors $FCFC$ qui est différents de la chaîne la plus probable $FFFC$.

Problème 3 : Connaissant les observations $\mathcal{O}$ ainsi que les dimensions N et M, on cherche à calculer le modèle $\lambda = (\pi, A, B)$ qui maximise la probabilité de $P(\mathcal{O}|\lambda)$.

4.2 Résolutions des trois problèmes

Le problème 1 se résout en calculant, comme dans l'exemple développé précédemment, $\sum_X P(\mathcal{O}, X|\lambda)$ mais cette méthode est trop coûteuse lorsque les observations deviennent nombreuse et il est préférable d'appliquer la partie « Forward » de l'algorithme « Forward-Backward » [BE67], [BS68] (détaillée dans le paragraphe 4.3).

Pour le problème 2 il existe plusieurs résolutions en fonction de la définition du terme « optimale ». En effet, si l'on choisit le point de vue des modèles de Markov cachés la résolution se fait en utilisant partiellement l'algorithme « Forward-Backward » (explicité dans la suite). En revanche, pour la programmation dynamique on doit utiliser l'algorithme de Viterbi pour trouver la solution [Vit67], [For73].

Enfin, la résolution du problème 3 n'est que partielle, en effet, dans l'article [Rab89] on lit qu'il n'existe aucun moyen de trouver analytiquement le modèle qui maximise la probabilité d'apparition de la séquence d'observation. Il n'existe en fait aucune manière optimale, étant donnée une séquence d'observations, d'estimer les paramètres du modèle. Il est toutefois possible de choisir les paramètres λ de manière à maximiser localement la probabilité $P(\mathcal{O}|\lambda)$ en utilisant des procédures itératives [LRS83],[KGMS12] telles que la méthode de Baum-Welch ou des méthodes dites de « gradients ».

Dans la suite nous allons présenter deux méthodes de résolution du problème 3 :

- La méthode de Baum-Welch qui utilise l'algorithme Forward-Backward permettant également de résoudre les problèmes 1 et 2.
- Une méthode de gradient d'ordre 2.

4.3 Méthode de Baum-Welch

La méthode de Baum-Welch ([BP66],[BPW70]) est une spécialisation de la méthode « Expectation-Maximization » pour le cas des chaînes de Markov cachées. Elle s'appuie notamment sur l'algorithme « Forward-Backward » pour l'étape « Expectation ».

On peut décomposer la méthode de Baum-Welch en quatre étapes :

— Étape Forward : calcul des coefficients $\alpha_t(i)$.

— Étape Backward : calcul des coefficients $\beta_t(i)$.

— Étape γ : calcul des coefficients $\gamma_t(i)$.

— Estimation des paramètres de la chaîne de Markov cachée.

Étape Forward :

Cette étape consiste à calculer la probabilité de la séquence d'observations partielles jusqu'au temps t :

$$\alpha_t(i) = P(\mathcal{O}_1 \dots \mathcal{O}_t, X_t = q_i|\lambda) \text{ pour } t \in \{1, \dots, T\} \text{ et } i \in \{1, \dots, N\}.$$

Pour ce faire on utilise la formule de récursion :

$$\begin{cases} \alpha_1(i) = \pi_i B_{i\mathcal{O}_1}. \\ \alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) A_{ij}\right] B_{j\mathcal{O}_{t+1}}. \end{cases}$$

Par la définition des $\alpha_t(i)$ on a que :

$$P(\mathcal{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

Cette étape permet donc de résoudre le problème 1.

Étape Backward :

Cette étape consiste à calculer la probabilité de la séquence d'observations partielle du temps $t+1$ au temps T :

$$\beta_t(i) = P(\mathcal{O}_{t+1} \dots \mathcal{O}_T|X_t = q_i, \lambda) \text{ pour } t \in \{1, \dots, T\} \text{ et } i \in \{1, \dots, N\}.$$

Pour ce faire on utilise la formule de récursion :

$$\begin{cases} \beta_T(i) = 1. \\ \beta_t(i) = \sum_{j=1}^{N} A_{ij} B_{j\mathcal{O}_{t+1}} \beta_{t+1}(j). \end{cases}$$

Étape γ :

On définit les probabilités de transiter de l'état q_i au temps t à l'état q_j au temps $t+1$:

$$\gamma_t(i,j) = P(X_t = q_i, X_{t+1} = q_j|\mathcal{O}, \lambda).$$

ce que l'on peut réécrire avec les variables des étapes précédentes comme :

$$\begin{aligned}\gamma_t(i,j) &= \frac{P(X_t = q_i, X_{t+1} = q_j, \mathcal{O}|\lambda)}{P(\mathcal{O}|\lambda)}, \\ &= \frac{\alpha_t(i) A_{ij} B_{j\mathcal{O}_{t+1}} \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)}.\end{aligned}$$

Enfin on calcule :

$$\gamma_t(i) = \sum_{j=1}^{N} \gamma_t(i,j).$$

Étape d'estimation des paramètres :

Les variables suivantes peuvent être interprétées en fonction d'espérances :

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{ espérance du nombre de transitions de } q_i.$$

$$\sum_{t=1}^{T-1} \gamma_t(i,j) = \text{ espérance du nombre de transitions de } q_i \text{ à } q_j.$$

En utilisant ces formules on définit la méthode d'estimation des paramètres A, B et π :

$$\begin{aligned}\pi_i &= \text{ espérance de la fréquence de l'état } q_i \text{ au temps } t = 1 \ = \gamma_1(i). \\ A_{ij} &= \frac{\text{espérance du nombre de transition de } q_i \text{ à } q_j}{\text{espérance du nombre de transitions de } q_i}, \\ &= \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \\ B_{jk} &= \frac{\text{espérance du nombre de fois où dans l'état } q_j \text{ on observe } k}{\text{espérance du nombre de fois où l'on se trouve dans l'état } q_j}, \\ &= \frac{\sum_{t=1 \text{ t.q. } \mathcal{O}_t = k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.\end{aligned}$$

En notant $\lambda^{(l)}$, le modèle estimé à l'étape l, les travaux de Baum ont montré que :

— Soit le modèle $\lambda^{(l)}$ est un point critique de la fonction de vraisemblance (extremum local) et dans ce cas $\lambda^{(l)} = \lambda^{(l+1)}$.

— Soit le modèle $\lambda^{(l+1)}$ est plus probable et dans ce cas $P(\mathcal{O}|\lambda^{(l+1)}) > P(\mathcal{O}|\lambda^{(l)})$.

4.4 Méthodes de gradients

Lors de la résolution du problème 3 on cherche le modèle qui maximise la probabilité qu'une séquence d'observations soit issue de ce modèle. Il est alors naturel d'utiliser des outils standards d'optimisation numérique. C'est ce postulat qui est à l'origine des méthodes de gradients ([CBM98], [QAS00], [Tur08]).

Il existe plusieurs algorithmes basés sur les outils d'optimisation (en fonction de l'ordre des dérivées partielles, des paramètres considérés pour calculer le gradient, etc). Nous allons présenter une méthode dite de « Quasi-Newton » qui consiste à calculer une approximation de la matrice Hessienne afin de ré-estimer les paramètres du modèle.

On introduit les paramètres $(W_{ij})_{0\leqslant i,j\leqslant N-1}$ et $(V_{ij})_{0\leqslant i,j\leqslant N-1}$ tels que :

$$A_{ij} = \frac{e^{W_{ij}}}{\sum_{j=0}^{N-1} e^{W_{ij}}} \quad \text{et} \quad B_{ij} = \frac{e^{V_{ij}}}{\sum_{j=0}^{N-1} e^{V_{ij}}}.$$

On note $\lambda^{(k)}, k \in \mathbb{N}$ le k-ème modèle calculé par l'algorithme ainsi que $\mathcal{L}(\lambda^{(k)}) = \log(P(\mathcal{O}|\lambda^{(k)}))$ la log-vraisemblance de ce modèle.

Pour des raisons calculatoires on suppose ici que $\lambda^{(k)}$ est un vecteur de taille $N^2 + NM$ contenant les paramètres :

$$W_{ij},\ 0 \leqslant i,j \leqslant N-1 \text{ et } V_{jk},\ 0 \leqslant j \leqslant N-1,\ 0 \leqslant k \leqslant M-1.$$

On note également H_k et η_k l'approximation de la matrice Hessienne et le pas calculés lors de l'étape k.

A l'itération k on va calculer le $(k+1)$-ème modèle via l'équation :

$$\lambda^{(k+1)} = \lambda^{(k)} + \eta_k H_k \nabla\mathcal{L}(\lambda^{(k)}).$$

Pour cela on doit déterminer les valeurs de $\nabla\mathcal{L}(\lambda^{(k)})$, de H_k et de η_k.

Calcul du gradient ∇ :

$$\nabla\mathcal{L}(\lambda(k)) = \left(\frac{\partial\mathcal{L}(\lambda^{(k)})}{\partial W_{00}}, \ldots, \frac{\partial\mathcal{L}(\lambda^{(k)})}{\partial W_{N-1N-1}}, \frac{\partial\mathcal{L}(\lambda^{(k)})}{\partial V_{00}}, \ldots, \frac{\partial\mathcal{L}(\lambda^{(k)})}{\partial V_{N-1M-1}}\right)^{\intercal},$$

avec

$$\frac{\partial\mathcal{L}(\lambda^{(k)})}{\partial W_{ij}} = \frac{1}{\mathcal{L}(\lambda^{(k-1)})}\left(\sum_{t=1}^{T-1}\gamma_t(i,j) - A_{ij}\sum_{t=1}^{T-1}\gamma_t(i)\right),$$

et

$$\frac{\partial \mathcal{L}(\lambda^{(k)})}{\partial V_{jk}} = \frac{1}{\mathcal{L}(\lambda^{(k-1)})} \left(\sum_{t=1 \ t.q \ \mathcal{O}_t=k}^{T} \gamma_t(j) - B_{ij} \sum_{t=1}^{T} \gamma_t(i) \right),$$

où les valeurs $\gamma_t(j)$ et $\gamma_t(i,j)$ sont celles présentées dans la méthode de Baum-Welch.

Calcul de H_k :

Il existe plusieurs possibilités pour construire la suite $(H_k)_k$ afin d'approximer la matrice Hessienne, nous présentons ici la méthode de Davidon-Fletcher-Powell (DFP).

On note $v_k = \lambda^{(k)} - \lambda^{(k-1)}$ et $g_k = \nabla\mathcal{L}(\lambda^{(k)}) - \nabla\mathcal{L}(\lambda^{(k-1)})$. Alors :

$$H_k = H_{k-1} + \frac{v_k v_k^\intercal}{v_k^\intercal g_k} - \frac{H_{k-1} g_k g_k^\intercal H_{k-1}}{g_k^\intercal H_{k-1} g_k}, \text{ et } H_0 = I.$$

Calcul de η_k :

Le calcul se fait par la méthode présentée dans le paragraphe 9.7 du livre [PTVF92]. Le problème vient du fait que si on est trop éloigné de la solution, le pas complet ($\eta_k = 1$) peut conduire à une diminution de la log-vraisemblance. On va alors chercher le pas $0 < \eta_k \leqslant 1$ maximum qui nous permet d'avoir une suite $(\mathcal{L}(\lambda^{(k)}))_k$ croissante.

4.5 Comparaison des deux méthodes

Pour comparer les deux algorithmes nous allons étudier leurs complexités, leurs performances et les résultats sur des séquences jouets.

Dans les deux cas, l'algorithme Forward-Backward est utilisé pour préparer la phase d'estimation des paramètres. En revanche cette dernière phase est différente pour les deux méthodes.

4.5.1 Analyse de l'algorithme Forward-Backward

Dans la littérature est souvent exposé le problème de gestion des calculs intermédiaires. En effet plus la taille de l'échantillon est importante, plus les éléments des matrices intermédiaires sont petits et nécessitent une précision importante. Ainsi l'algorithme est présenté avec une variante composée d'étapes de mises à l'échelle (scaling). Dans la pratique, les calculs des différentes étapes (Forward, Backward, Gamma) sont ajustés (multiplication par des coefficients adaptés) afin de rester dans le domaine de précision. Cependant, de nos jours les logiciels de calcul exact nous permettent d'éviter les étapes de scaling.

Nous avons quatre implantations différentes de l'algorithme :

— L'algorithme « `Exact` » sans scaling et qui ne fait que du calcul exact.

— L'algorithme « `N_Exact` » sans scaling mais qui fait des calculs flottants. On doit donc augmenter la précision quand l'échantillon est plus important.

— L'algorithme « `Scal` » avec scaling et calculs flottants.

— L'algorithme « `Para` » avec scaling et calculs flottants et l'étape gamma en parallèle.

Voici la complexité de ces différents algorithmes en termes d'opérations élémentaires (résultats pour une itération de l'algorithme) :

Algorithme	**Additions**	**Multiplications**	**Divisions**
`Exact` / `N_Exact`	$3(T-1)N^2+N$	$6(T-1)N^2+T.N$	$(T-1)N^2$
`Scal` / `Para`	$3(T-1)N^2+(T+1)N$	$7(T-1)N^2+2T.N$	$(T-1)N^2+T$

TABLE 4.3: Comparaison du nombre d'opérations élémentaires pour les algorithme de Baum Welch.

Néanmoins, à part pour l'algorithme `Exact`, les calculs sont fait en précision p, arbitraire. Ainsi, la complexité de l'addition est en $\mathcal{O}(p)$ et la complexité de la multiplication et de la division est en $\mathcal{O}(M(p))$ où $M(p)$ représente le coût de la multiplication pour un entier de taille p.

Dans les exemples numériques qui suivent, les calculs sont faits à l'aide de la bibliothèque `GMP`, l'algorithme de multiplication utilisé dépend donc de la précision [Gt14]. Notons N_p le nombre de blocs de 64 bits de mantisse nécessaires pour avoir une précision p :

$$N_p = \left\lceil \frac{ln(10)}{ln(2^{64})}\, p \right\rceil,$$

on a alors les complexités de multiplication suivantes :

N_p	$[0,18]$	$[19,57]$	$[58,154]$	$[155,226]$	$[226,333]$	$[334,4736]$
Algorithme	Naïf	Karatsuba	Toom-3	Toom-4	Toom-6.5	Toom-8.5
$\mathcal{O}(M(p))$	$\mathcal{O}(N_p^2)$	$\mathcal{O}(N_p^{1.585})$	$\mathcal{O}(N_p^{1.465})$	$\mathcal{O}(N_p^{1.404})$	$\mathcal{O}(N_p^{1.318})$	$\mathcal{O}(N_p^{1.289})$

TABLE 4.4: Algorithme de multiplication en fonction de la précision p.

Dans le cadre de l'algorithme Forward-Backward, les différences de complexité viennent du fait que la version `Scal` nécessite une précision constante alors que la version `N_Exact` a une précision qui dépend de la taille de l'échantillon. Par exemple :

Algorithme	T	p	N_p	Multiplication
`N_Exact`	10	38	2	Naïve
`N_Exact`	1 000	1 300	68	Toom-3
`N_Exact`	10 000	13 000	675	Toom-8.5
`Scal`	10, 1 000, 10 000	38	2	Naïve

Dans le tableau qui suit on présente un comparatif des temps de calculs (en secondes) pour les différentes versions de l'algorithme exécutées sur un processeur Intel Core i7-3840QM possédant 4 cœurs à 2.8GHz :

N	T	Itérations	`Exact`	`N_Exact`	`Scal`	`Para`
16	10	2	545	0.036	0.045	0.157
16	1 000	10	/	16	17	15
16	10 000	10	/	561	177	155
16	50 000	10	/	/	909	769

TABLE 4.5: Temps de calcul de l'algorithme de Baum Welch pour différents paramètres.

L'algorithme en calcul exact est très rapide sur la première itération mais ensuite les fractions devenant trop grandes le temps de calcul est fortement dégradé (après la première itération les numérateurs et dénominateurs sont des nombres à environ 500 chiffres). L'algorithme flottant sans scaling est rapide mais plus l'échantillon est grand plus la précision doit être augmentée et les performances pour des tailles d'échantillons supérieures à 1 000 sont nettement moins bonnes que les algorithmes avec scaling.

Dans la suite nous utiliserons donc les méthodes avec mise à l'échelle, en tenant compte de la taille des observations pour choisir entre l'algorithme séquentiel et l'algorithme parallèle.

4.5.2 Estimation des paramètres

La phase d'estimation des paramètres est plus complexe pour la méthode de gradient (G) que pour celle de Baum-Welch (BW), en effet elle nécessite le calcul du gradient, de la matrice hessienne ainsi que du pas. Ce dernier calcul est le plus coûteux car pour chaque pas testé il faut connaître la valeur de log-vraisemblance du nouveau modèle ce qui implique d'exécuter une étape Forward.

Ce phénomène est illustré par le nombre d'opérations élémentaires nécessaires pour chaque étape :

Étape	Additions	Multiplications	Divisions
Estimation (BW)	$4(T-1)N^2+T$	0	$2N^2$
Gradient (G)	$2(2T-1)N^2$	$2N^2$	$2N^2$
Hessienne (G)	$4N^2+4N-2$	$4N^2+4N$	$2N^2+2N$
Pas (G)	$\rho(T-1)N^2+$ $(\rho(T+2)+2)N$	$\rho(T-1)N^2+$ $(2\rho(T+1)+2)N$	$N+\rho(T+7)$

TABLE 4.6: Nombre d'opérations élémentaires pour les différentes étapes d'estimation des parmaètres.

où ρ désigne le nombre maximal de pas que l'on peut tester.

Les performances (en temps) sur un processeur Intel Core i7-3840QM 2.8GHz sont également révélatrices du coût plus important de la méthode de gradient :

N	ρ	T	Itérations	Baum-Welch (s)	Gradient (s)
16	10	100	10	1.780	4.550
16	10	1 000	10	17	33
16	10	10 000	10	177	347

TABLE 4.7: Comparaison des temps de calcul entre l'algorithme de Baum Welch et du gradient.

4.6 Résultats sur des séquences jouets

4.6.1 Construction des séquences

Pour illustrer les résultats des deux méthodes présentées nous avons construit des séquences jouets suivants des modèles de Markov cachés.

La construction de séquences ayant pour modèle $\lambda = (\pi, A, B)$ se fait en deux étapes :

- Construire une chaîne de Markov $X_0, \ldots, X_{T-1}$ de matrice de transition A et de distribution initiale π (comme expliqué dans le paragraphe 3.2.1).
- Construire une séquence d'observations $\mathcal{O}_0, \ldots, \mathcal{O}_{T-1}$ à partir de la chaîne de Markov.

Lors de la deuxième étape, pour construire l'élément $\mathcal{O}_t$ le principe est semblable à la construction de la chaîne de Markov, on tire aléatoirement $r_t \in [0,1]$, puis en considérant $i = X_t$ on construit la suite d'intervalles :

$$\mathcal{I}_0 = [0, B_{i0}],\ \mathcal{I}_1 = [B_{i0}, B_{i0}+B_{i1}],\ \ldots, \mathcal{I}_{15} = \left[\sum_{j=0}^{14} B_{ij}, 1\right].$$

Si on note k l'entier tel que $r_t \in \mathcal{I}_k$ alors $\mathcal{O}_t = k$.

En suivant ce principe nous avons construit trois séquences, dont les paramètres sont présentés ci-dessous :

Nom	Taille	Motifs	π	A	B	E1	E2
`HMM_R`	40Mo	4 bits	Uniforme	R	R	3.990	3.774
`HMM_RU`	40Mo	4 bits	Uniforme	RU	RU	4.000	3.978
`HMM_RH`	40Mo	4 bits	Uniforme	RU	RH	4.000	3.955

TABLE 4.8: Constructions des séquences jouets pour les modèles de Markov cachés.

avec

— E1 l'entropie de Shannon et E2 l'entropie minimale.

— R une matrice stochastique aléatoire.

— RU une matrice stochastique dont les coefficients sont de la forme $a_{ij} = \frac{1}{16} + \epsilon_{ij}$ avec $\sum_j \epsilon_{ij} = 0$.

— RH une matrice stochastique dont les coefficients sont de la forme :

$$\begin{cases} a_{ij} = \frac{1}{2} + \epsilon_{ij} \text{ si } j \in \{i, (i+1) \mod 16\}, \\ a_{ij} = \epsilon_{ij} \text{ sinon,} \end{cases}$$

avec $\sum_j \epsilon_{ij} = 0$.

4.6.2 Analyse statistique classique des séquences jouets

Outre l'entropie des séquences jouets précisée dans le tableau précédent, il est intéressant de voir le comportement des séquences générées par les modèles de Markov cachés par rapport aux tests statistiques classiques. Ainsi, nous présentons les résultats de deux tests de la batterie FIPS : le test de biais et le test de fréquence sur les motifs de 4 bits.

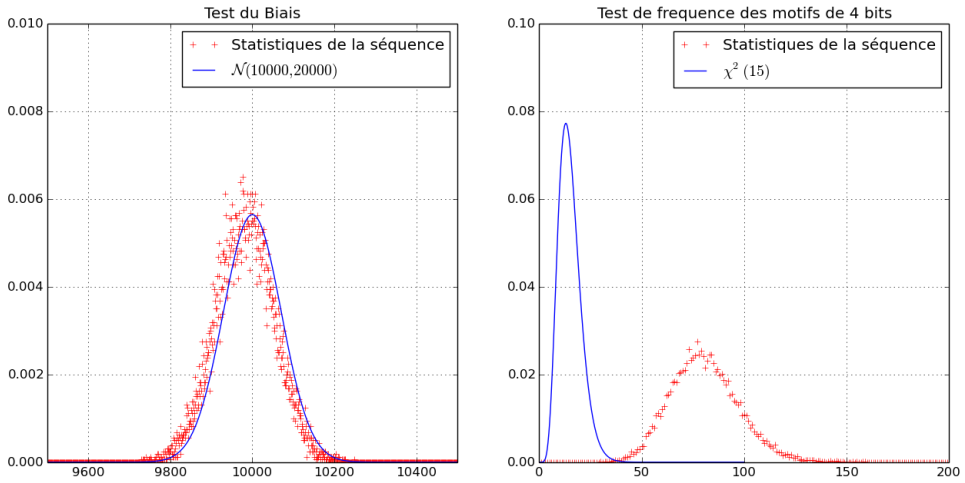


FIGURE 4.1: Tests de biais et de fréquence pour la séquence `HMM_R`

Pour le modèle jouet construit à partir de matrices aléatoires, les tests statistiques révèlent une légère déviation de biais (moyenne autour de 9990) ainsi qu'une déviation plus importante concernant la fréquence des motifs de 4 bits (moyenne autour de 80).

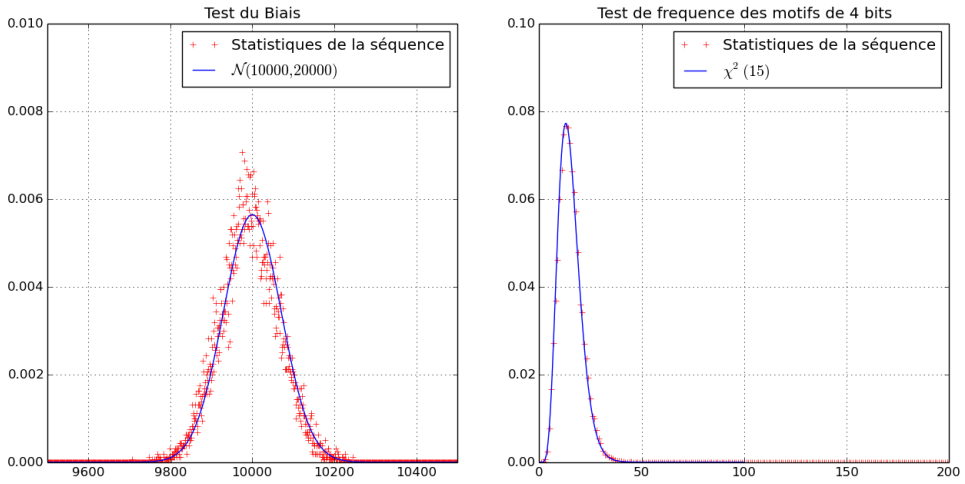


FIGURE 4.2: Tests de biais et de fréquence pour la séquence `HMM_RU`

Dans le cas du modèle basé sur des matrices uniformes, les deux test présentés ne montrent aucun défaut significatif, tant pour le biais que pour la fréquence des motifs. En revanche pour le dernier modèle jouet, dont la construction repose sur une matrice A uniforme et une matrice B dégénérée (voir construction de la matrice `RH` 4.6.1), les deux tests révèlent une légère déviation statistique à la fois sur le biais (moyenne autour de 9950) et sur la fréquence des motifs (moyenne autour de 16).

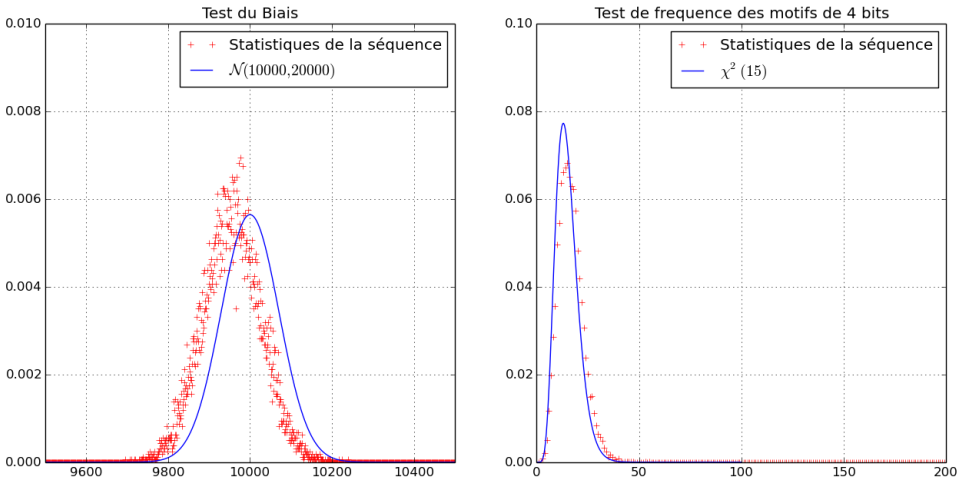


FIGURE 4.3: Tests de biais et de fréquence pour la séquence `HMM_RH`

L'intérêt de présenter les résultats de ces séquences aux tests statistiques est que ces données peuvent être considérées comme des caractéristiques du modèle. Ainsi, lorsque l'on cherche à inférer un modèle de Markov caché à partir d'une séquence aléatoire, on peut utiliser les résultats aux tests statistiques comme un critère de qualité du modèle inféré.

4.6.3 Comparaison entre deux modèles de Markov cachés

Afin de pouvoir contrôler le bon fonctionnement des algorithmes présentés, il faut pouvoir comparer deux modèles de Markov cachés en définissant une notion de distance.

Une première idée pour estimer la « proximité » de deux modèles $\lambda = (A, B, \pi)$ et $\lambda' = (A', B', \pi')$ et de calculer la distance euclidienne entre les matrices A, A' et B, B' :

$$d(A, A') = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A_{ij} - A'_{ij})^2},$$

$$d(B, B') = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A_{ij} - A'_{ij})^2}.$$

Cependant, il existe des modèles dont les matrices A, A' et B, B' sont différentes et produisant des séquences d'observations similaires. Prenons les deux modèles triviaux suivants :

$$A = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix},$$

et

$$A' = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \quad B' = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

Dans cet exemple les distances euclidiennes entre les matrices sont non nulles et pourtant les séquences d'observations générées sont identiques : $\mathcal{O} = (1, 1, 1, 1, \dots)$.

Ainsi, dans le but d'évaluer la distance entre deux modèles de Markov cachés, il est primordial de tenir compte à la fois des matrices qui définissent chacun des modèles et des séquences d'observations générées par ces modèles.

Les distances les plus utilisées dans la littérature pour répondre à ces exigences sont des variantes de la divergence de Kullback-Leibler [RJLS85, JR85, XUP05]. Cette divergence appelée également entropie relative est une mesure non-symétrique de la différence entre deux distributions de probabilité [Kul68] :

$$\mathcal{D}_{KL}(P, Q) = \sum_{i \in \Omega} P(i) \log \left(\frac{P(i)}{Q(i)} \right),$$

pour deux distributions discrètes de même univers Ω.

La distance que nous allons utiliser pour présenter les résultats des deux algorithmes est dérivée de la divergence de Kullback-Leibler afin d'être adaptée au cas des chaînes de Markov cachés à espace d'états discrets et espace d'observations discrets [XUP05].

Conformément avec les notations précédemment utilisées la distance entre deux modèles $\lambda = (A, B, \pi)$ et $\lambda' = (A', B', \pi')$ se calcule comme suit :

$$\mathcal{D}(\lambda, \lambda') = \frac{1}{T} \log \left(\frac{\sum_{i=0}^{N-1} \xi(\lambda, i, T)}{\sum_{i=0}^{N-1} \xi(\lambda', i, T)} \right),$$

avec pour $t \in \{1, \dots, T\}$:

$$\xi(\lambda, i, t) = \sum_{j=0}^{N-1} N A_{ji} \, B_{j\mathcal{O}_t} \, \xi(\lambda, j, t-1),$$

et $\xi(\lambda, i, 0) = \pi_0$ pour tout $i \in \{0, \dots, N-1\}$.

Pour présenter les résultats de comparaison entre les deux algorithmes sur des modèles de Markov cachés jouets, nous avons fait varier plusieurs paramètres :

— T le nombre d'observations.

— I le nombre maximum d'itérations des algorithmes.

— S la séquence utilisée.

— A_0 et B_0 les matrices d'initialisation du modèle.

Pour la clarté des figures, on introduit trois matrices d'initialisation M_U la matrice uniforme, M_R une matrice aléatoire et M_H la matrice dont les coefficients diagonaux et sur-diagonaux valent $\frac{1}{2}$ et les autres 0.

Nous avons effectué les tests sur des séquences d'observations de tailles 10 000 et 50 000, les interprétations étant similaires, nous ne présenterons que les résultats sur les séquences de taille 50 000, ceux pour les séquences de 10 000 observations sont en annexes.

4.6.4 Résultats pour 50 000 observations avec initialisation aléatoire

Lors de l'étude d'une séquence aléatoire venant d'un NDRBG il est rare d'avoir des informations précises concernant la structure du générateur, ainsi il est important d'étudier le comportement des méthodes présentées ci-dessus dans le cas d'une initialisation du modèle aléatoire.

Les graphiques exposés ci-dessous présentent à la fois la log-vraisemblance des modèles λ_i itérés, c'est-à-dire $\log(P(\lambda_i \mid \mathcal{O}))$ pour $i \in \{0, \dots, I\}$ ainsi que la valeur de la variante de la divergence de Kulback-Liebler (voir paragraphe 4.6.3) $\mathcal{D}(\lambda, \lambda_i)$ en considérant λ comme le modèle théorique que l'on cherche à inférer.

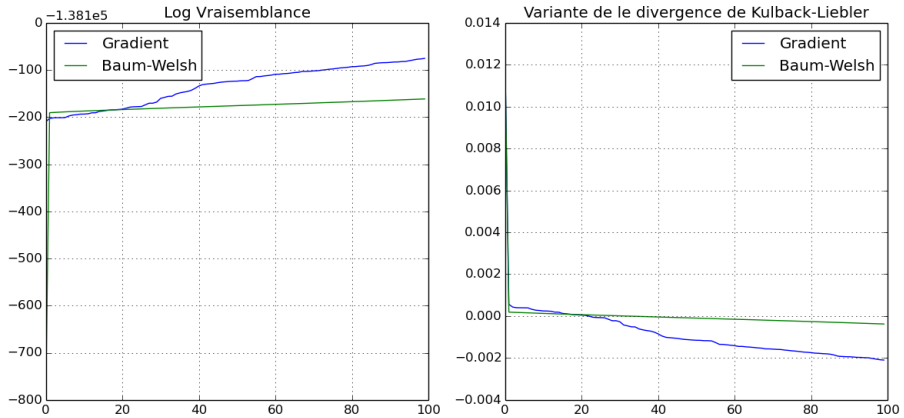


FIGURE 4.4: Méthodes de Baum-Welch et de gradient pour $S =$ `HMM_R`, $T = 50\,000$, $I = 100$, $A_0 = M_R$ et $B_0 = M_R$.

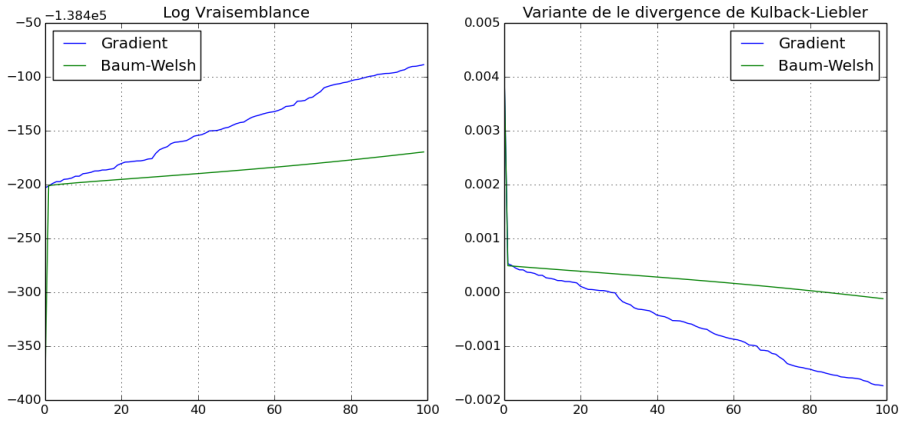


FIGURE 4.5: Méthodes de Baum-Welch et de gradient pour $S =$ `HMM_RH`, $T = 50\,000$, $I = 100$, $A_0 = B_0 = M_R$.

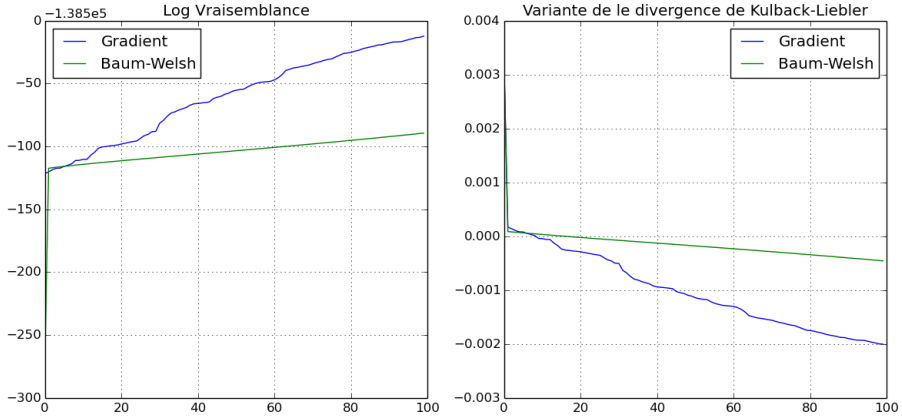


FIGURE 4.6: Méthodes de Baum-Welch et de gradient pour $S =$ `HMM_RU`, $T = 50\,000$, $I = 100$, $A_0 = B_0 = M_R$.

De manière générale, la méthode du gradient converge plus rapidement que la méthode de Baum-Welch vers des modèles λ' dont la probabilité, relativement à la séquence d'observation $\mathcal{O}$, est plus grande. On remarque notamment qu'à partir d'une certaine itération (environ 10 itérations pour la figure 4.4) les modèles ont la propriété :

$$P(\mathcal{O}|\lambda') \geqslant P(\mathcal{O}|\lambda),$$

où λ est le modèle de la séquence jouet.

4.6.5 Résultats pour 50 000 observations avec initialisation ciblée

Les travaux de Baum, Welsh et Rabiner font état de la difficulté d'inférer le modèle dans le cas que nous venons de présenter, c'est-à-dire sans avoir d'information sur le modèle de Markov caché théorique. Nous allons donc voir dans le paragraphe suivant les résultats des deux algorithme lorsque l'initialisation est proche (au sens de la norme euclidienne sur les matrices A et B) du modèle théorique.

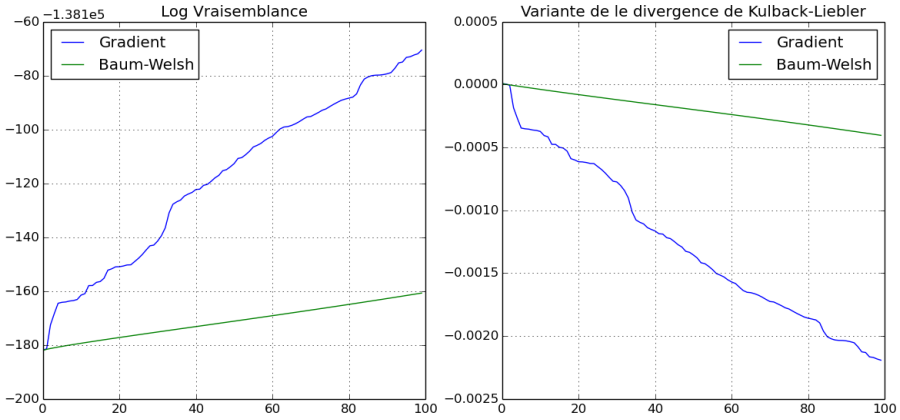


FIGURE 4.7: Méthodes de Baum-Welch et de gradient pour $S =$ `HMM_R`, $T = 50\,000$, $I = 100$, $A_0 = B_0 = M_R$.

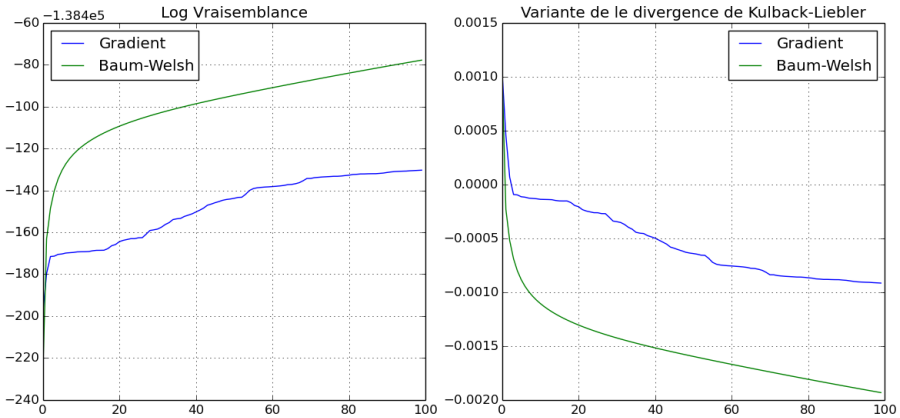


FIGURE 4.8: Méthodes de Baum-Welch et de gradient pour $S =$ `HMM_RH`, $T = 50\,000$, $I = 100$, $A_0 = M_U$ et $B_0 = M_H$.

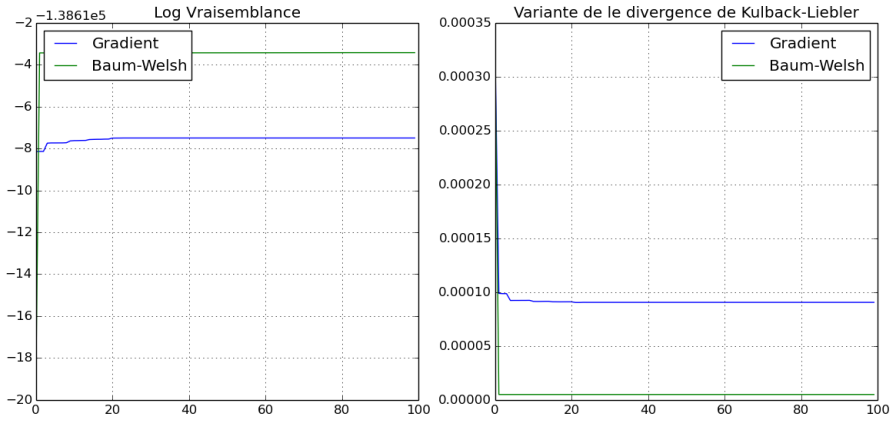


FIGURE 4.9: Méthodes de Baum-Welch et de gradient pour $S =$ `HMM_RU`, $T = 50\,000$, $I = 100$, $A_0 = B_0 = M_U$.

Les deux méthodes n'ont pas le même comportement en fonction du type d'initialisation. En effet la méthode du gradient est très peu influencée par l'initialisation et les modèles itérés ont des probabilités similaires relativement à la séquence d'observation $\mathcal{O}$.

Cependant, la méthode de Baum-Welch dépend fortement de l'initialisation, ainsi pour l'initialisation $A_0 = B_0 = M_U$ de la figure 4.9 l'algorithme reste « bloqué » autour des matrices uniformes (qui représentent une forme d'état d'équilibre). On voit également que pour la figure 4.8 la méthode de Baum-Welch converge plus rapidement que la méthode du gradient, tirant dans ce cas avantage de l'initialisation.

En conclusion, ces résultats tendent à montrer que la méthode du gradient bien que plus coûteuse, itère des modèles plus probables dans le cas général. En revanche lorsque l'on dispose d'informations sur le modèle théorique, la méthode de Baum-Welch peut donner de meilleurs résultats.

4.7 Simulation de propriétés statistiques utilisant les HMM

Dans le paragraphe 4.6.2, nous avons explicité les résultats statistiques des différentes séquences jouets. Nous allons pouvoir ainsi comparer les résultats des séquences jouets avec les résultats de séquences construites à partir des modèles inférés. L'objectif étant de voir s'il est possible, à partir d'une séquence d'observations, de construire un modèle de Markov caché possédant les mêmes analyses statistiques que la séquence d'origine.

Afin de rendre les graphiques plus lisibles, les courbes présentées pour les résultats du test du biais sont les lois gaussiennes, approximées par la méthode des moindres carrées non

linéaire, résultant des échantillons étudiés. A titre d'exemple, nous présentons (Fig. 4.10) l'approximation gaussienne pour les statistiques de la séquence `HMM_RU`.

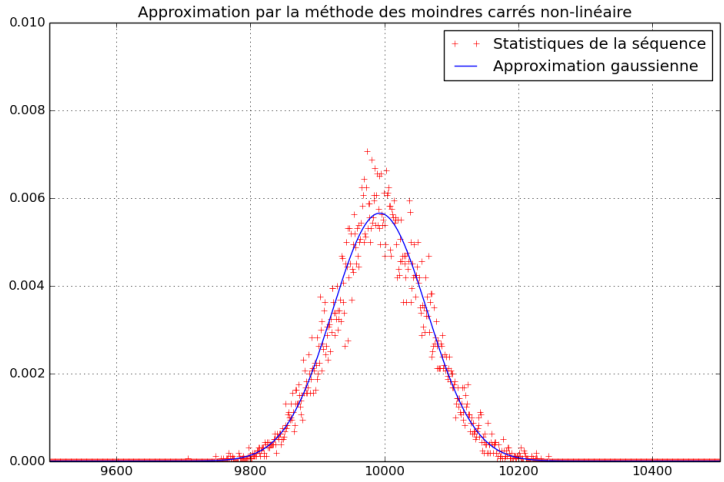


FIGURE 4.10: Approximation par la méthode des moindres carrés non linéaire des statistiques du test du biais pour la séquence `HMM_RU`.

Pour chaque séquence jouet nous avons utilisé les méthodes de Baum Welch et du gradient avec différents paramètres pour calculer des modèles de Markov cachés inférés que nous notons $\lambda_{\text{Seq,Meth,T,Init}}$ où Seq représente la séquence (R,RU ou RH), Meth la méthode d'inférence (BW ou G), T la taille de la séquence d'échantillon et Init l'initialisation aléatoire ou ciblée (A ou C). Une fois le modèle inféré, nous construisons alors une séquence de taille identique à la séquence d'origine, basée sur le modèle de Markov $\lambda_{\text{Seq,Meth,T}}$.

Par exemple pour le graphique 4.11, nous comparons les résultats statistiques entre la séquence `HMM_R` et la séquence construite à partir du modèle inféré $\lambda_{\text{R,G,50\,000,A}}$.

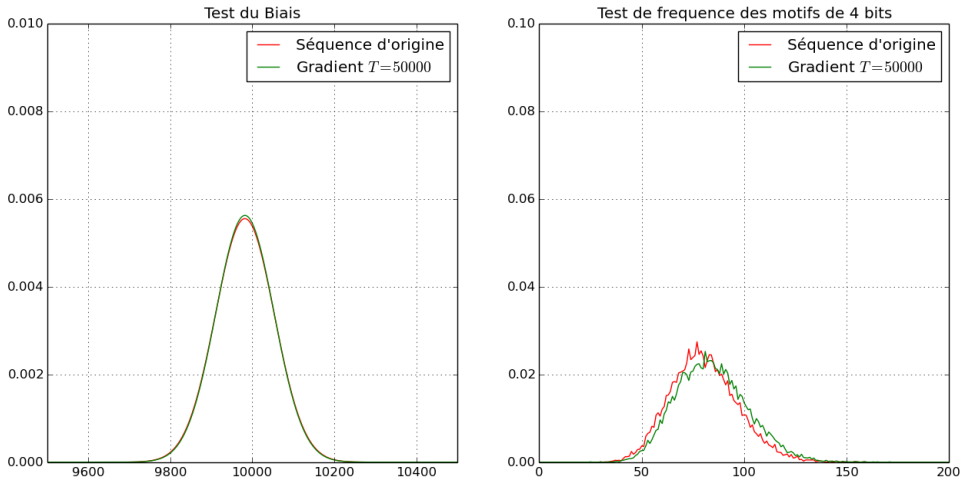


FIGURE 4.11: Comparaison du comportement statistique entre la séquence `HMM_R` et la séquence issue du modèle $\lambda_{\mathrm{R,G,50\,000,A}}$.

Tout d'abord nous pouvons remarquer que la séquence inféré sur `HMM_R` a des résultats statistiques similaires à ceux de la séquence d'origine. Cependant, certains paramètres influent sur la qualité du modèle itéré. On déduit ainsi de la figure 4.12 que la taille de la séquence d'observation est un facteur important pour avoir un comportement statistique similaires entre les modèles.

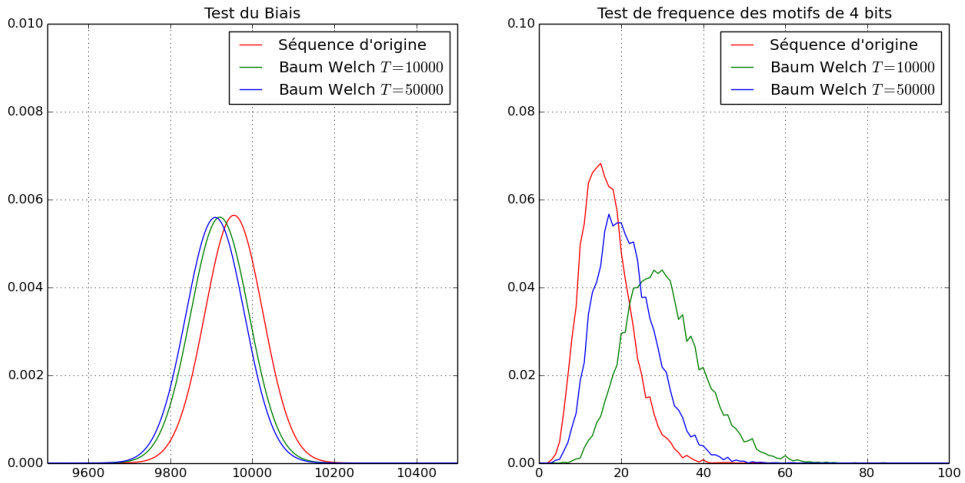


FIGURE 4.12: Comparaison du comportement statistique entre la séquence `HMM_RH` et les séquences issue des modèles $\lambda_{\mathrm{RH,BW,10\,000,A}}$ et $\lambda_{\mathrm{RH,BW,50\,000,A}}$.

En effet, plus la taille de l'échantillon est grande plus le modèle itéré est proche (en terme de résultats statistiques) du modèle de départ.

En revanche, quelle que soit la méthode utilisée, l'initialisation (aléatoire ou ciblée) n'améliore pas l'inférence du modèle.

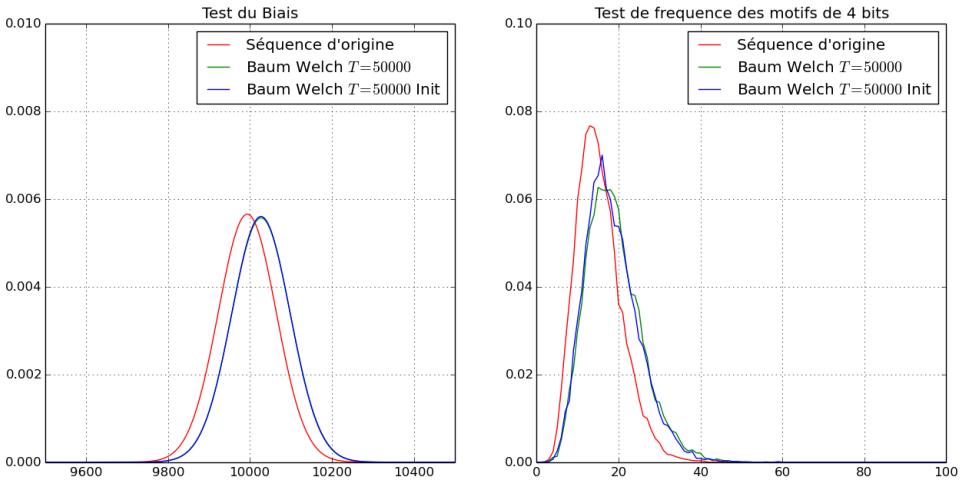


FIGURE 4.13: Comparaison du comportement statistique entre la séquence `HMM_RU` et les séquences issue des modèles $\lambda_{\mathrm{RU,BW,50\,000,A}}$ et $\lambda_{\mathrm{RU,BW,50\,000,C}}$.

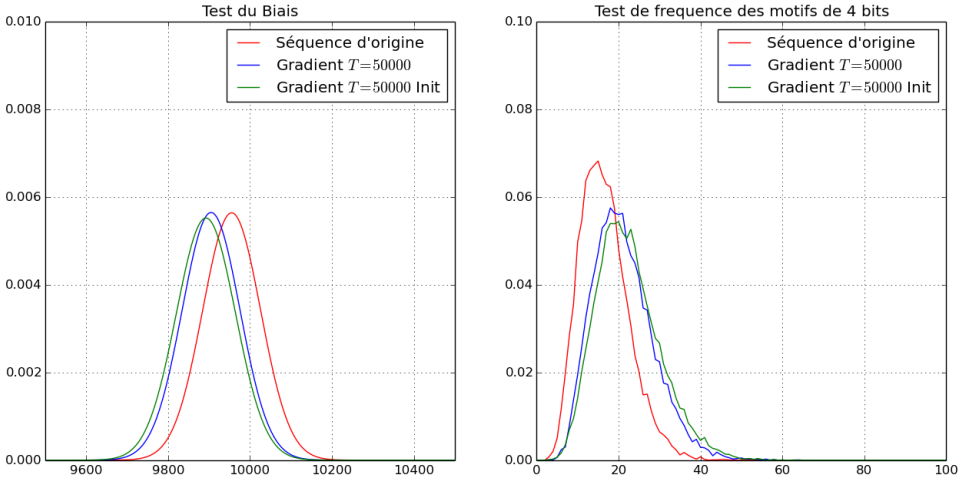


FIGURE 4.14: Comparaison du comportement statistique entre la séquence `HMM_RH` et les séquences issue des modèles $\lambda_{\mathrm{RH,G,50\,000,A}}$ et $\lambda_{\mathrm{RH,G,50\,000,C}}$.

Concernant la différence entre les deux méthodes, la méthode du gradient confère aux modèles itérés des propriétés statistiques plus proches de celles des modèles originaux mais la différence n'est pas significative par rapport à la méthode de Baum Welch et ne justifie pas, pour les tests effectués, la différence de temps de calcul.

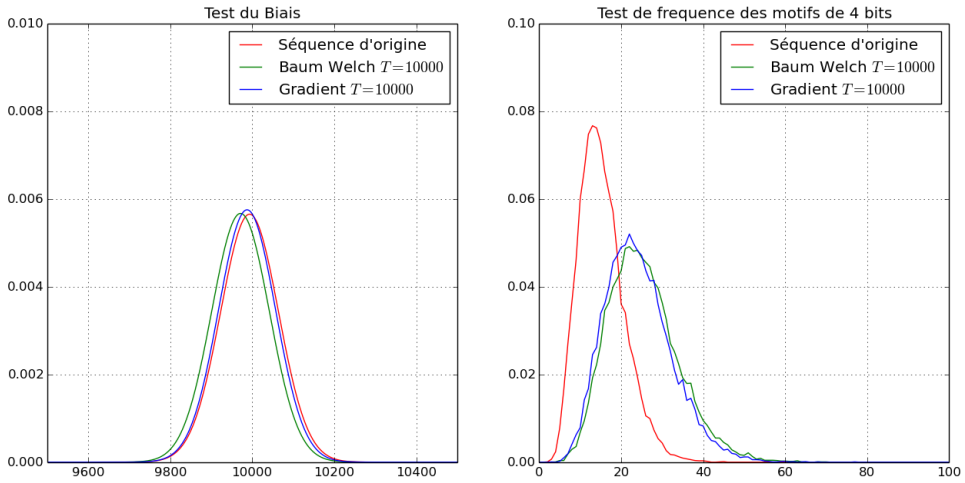


FIGURE 4.15: Comparaison du comportement statistique entre la séquence `HMM_RU` et les séquences issue des modèles $\lambda_{\mathrm{RU,BW,10\,000,A}}$ et $\lambda_{\mathrm{RU,G,10\,000,A}}$.

En conclusion, pour que les modèles de Markov cachés inférés soient indistinguables, par l'analyse statistique classique des NDRBG, des modèles théoriques le paramètre le plus important est la taille de la séquence d'observations. Les tests que nous avons menés montrent qu'à partir de 50000 échantillons, il était possible de reproduire les résultats d'une séquence aléatoire (Fig. 4.11) et d'en déduire un modèle de Markov.

4.8 Conclusion

Tout au long de ce mémoire nous avons insisté sur la nécessité de choisir, pour les analyses statistiques, des modèles réalistes par rapport aux contraintes physiques liées aux sources d'entropie. Les modèles de Markov cachés que nous avons présentés dans ce chapitre sont un bon exemple de modèles réalistes.

En effet, le principe d'une séquence observable générée à partir d'une séquence d'états cachés peut modéliser la construction d'un NDRBG, dans lequel nous n'avons accès qu'à la séquence de nombres finale et non aux différents états internes du générateur.

Les méthodes d'inférences expliquées dans ce chapitre permettent à la fois d'analyser un générateur dont le modèle serait les chaînes de Markov cachées mais également de construire une séquence basée sur ces modèles simulant le comportement statistique d'une séquence de modèle quelconque.

Les outils proposés ci-dessus forment une base sur laquelle beaucoup de développements sont possibles autour de ces modèles, notamment la construction de tests statistiques ayant pour hypothèse les chaînes de Markov cachées.

Chapitre 5

Conclusion et Perspectives

Tout au long de ce mémoire nous avons souligné l'importance de la modélisation et de la validation des générateurs aléatoires. Nous avons également prêté une attention particulière à leurs utilisations dans les systèmes embarqués, ce qui introduit des contraintes de consommation, de puissance de calcul,....

C'est dans ce contexte que nous avons, dans un premier temps, contribué à l'état de l'art des sources d'entropie et des modèles stochastiques qui leur sont associés. Nous avons ainsi proposé des sources d'entropie issue de la physique quantique. Les phénomènes que nous avons exposés étant fortement liés à l'informatique quantique, le développement de ce domaine permettra à l'avenir leurs utilisations cryptographiques.

Dans un deuxième temps, nous avons développé des outils d'analyse temporelle de l'homogénéité des séquences de nombres aléatoires. En effet, la plupart des batteries et standards actuels supposent, pour leurs analyses (par exemple le test de Maurer), que les générateurs fournissent des séquences homogènes dans le temps. Nous avons donc proposé, basé sur le modèle des chaînes de Markov, un algorithme de recherche des blocs homogènes d'une séquence de NDRBG. Une fois cette hypothèse vérifiée les conditions imposées par les tests tels que celui de Maurer sont vérifiées.

Nous avons également mis au point des tests statistiques classiques (voir schéma 1.2.1) prenant comme hypothèse nulle le modèle des chaînes de Markov afin d'intégrer dans H_0 la dépendance des variables.

Dans un troisième et dernier temps, nous avons étudié des modèles plus larges, les chaînes de Markov cachées, afin d'adapter les outils existant aux problématiques des générateurs d'aléa cryptographique. Nous avons proposé un comparatif entre différentes méthodes d'inférence pour ce type de modèles et utilisé celles-ci pour simuler le comportement statistique de séquences jouets.

Ce travail nous a permis de mettre en évidence la fracture existante entre les modèles stochastiques issus des sources d'entropie et les modèles statistiques utilisés dans les tests d'hypothèses des normes et des batteries actuelles.

D'un côté, il est important de poursuivre l'amélioration des modèles stochastiques afin de dériver, par exemple, des tests adaptés aux spécificités de la source d'entropie associée.

D'un autre côté, les tests génériques proposés par les institutions de standards et de certification doivent intégrer des modèles moins stricts pour respecter les contraintes physiques liées aux générateurs non-déterministes. Les chaînes de Markov cachées proposent un compromis idéal entre le reflet de la réalité physique et le besoin d'outils mathématiques pour la construction de tests statistiques. Cependant, il reste à développer les méthodes de résolutions des problèmes évoqués dans le chapitre 4, notamment les méthodes d'inférences ainsi que le calcul de distance entre deux modèles de Markov cachés.

Enfin, il est primordial, pour contrôler la qualité d'un générateur, de pouvoir détecter pendant son fonctionnement l'apparition de défauts statistiques. Pour se faire, il est naturel de considérer une analyse temporelle de la séquence émise. L'algorithme de découpage présenté au chapitre 3 est construit de manière à être réalisé en ligne. Néanmoins, il ne respecte pas les contraintes des tests embarqués, notamment en terme de capacité de calcul. Ainsi, il serait intéressant de l'adapter afin de respecter ces conditions et de pouvoir dériver un test embarqué contrôlant l'homogénéité des séquences dans le temps.

Bibliographie

[AFG12] A. Aspect, C. Fabre, and G. Grynberg. Optique quantique i : Lasers, 2012. https://catalogue.polytechnique.fr/site.php?id=63&fileid=347.

[AHMO13] T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye. A worst-case-aware design methodology for noise-tolerant oscillator-based true random number generator with stochastic behavior modeling. *IEEE Transactions on Information Forensics and Security*, 2013. http://dblp.uni-trier.de/db/journals/tifs/tifs8.html#AmakiHMO13.

[Ald94] P. Aldebert. *Simulation en régime transitoire des bruits dans les circuits électroniques*. 1994.

[Ama13] T. Amaki. A study on oscillator-based true random number generator robust to environmental fluctuation, 2013. http://ir.library.osaka-u.ac.jp/dspace/bitstream/11094/27479/1/25846_%E8%AB%96%E6%96%87.pdf.

[ANS14] ANSSI. Annexe b1 au référentiel général de sécurité (version 2.0) : Choix et dimensionnement des mécanismes cryptographiques, 2014. http://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf.

[Ara] Araneus Alea II TRNG. http://www.araneus.fi/products/alea2/en/.

[Ash90] R.B. Ash. *Information theory*. Dover Publication, 1990.

[BE67] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73 :360–363, 1967. http://projecteuclid.org/download/pdf_1/euclid.bams/1183528841.

[BFV10] F. Bernard, V. Fischer, and B. Valtchanov. Mathematical Model of Physical RNGs Based On Coherent Sampling. *Tatra Mountains - Mathematical Publications*, 2010. https://hal-ujm.archives-ouvertes.fr/ujm-00531665/file/2010_tatra.pdf.

[BKS12a] Elaine Barker, John Kelsey, and John Bryson Secretary. Nist draft special publication 800-90a recommendation for random number generation using deterministic random bit generators, 2012. http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf.

[BKS12b] Elaine Barker, John Kelsey, and John Bryson Secretary. Nist draft special publication 800-90b recommendation for the entropy sources used for random bit generation, 2012. http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf.

[BLMT10] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux. On the security of oscillator-based random number, 2010. https://perso.univ-rennes1.fr/david.lubicz/articles/gda.pdf.

[Bou08] R. Bourlès. Mouvements browniens, 2008. http://renaud.bourles.perso.centrale-marseille.fr/MathsFi/Chap%2015%20-%20Mouvements%20Browniens.pdf.

[BP66] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37 :1554–1563, 1966. http://projecteuclid.org/download/pdf_1/euclid.aoms/1177699147.

[BPW70] L.E. Baum, G.S. Petrie, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics 41*, 1970. http://projecteuclid.org/download/pdf_1/euclid.aoms/1177697196.

[Bre04] R. P. Brent. Note on marsaglia's xorshift random number generators. *Journal of Statistical Software*, 2004. http://maths-people.anu.edu.au/~brent/pd/rpb218.pdf.

[BRGD13] M. Ben-Romdhane, T. Graba, and J. L. Danger. Stochastic model of a metastability-based true random number generator. *Lectures Note in Computer Science*, 2013. http://link.springer.com/chapter/10.1007/978-3-642-38908-5_7.

[BS68] L. E. Baum and G. R. Sell. Growth transformations for functions on manifolds. *Pacific J. Math.*, 27(2) :211–227, 1968. https://projecteuclid.org/download/pdf_1/euclid.pjm/1102983899.

[Bé13] J. Béard. *Notes de Cours : Chaînes de Markov*, 2013. http://math.univ-lyon1.fr/~jberard/notes-CM-www.pdf.

[CBM98] O. Cappe, V. Buchoux, and E. Moulines. Quasi-newton method for maximum likelihood estimation of hidden markov models. *Speech and Signal Processing, ICASSP98*, 1998. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.1821&rep=rep1&type=pdf.

[CFAF13a] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A self-timed ring based true random number generator. *International symposium on advanced research in asynchronous circuits and systems*, 2013. https://hal.archives-ouvertes.fr/ujm-00840593/document.

[CFAF13b] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A very high speed true random number generator with entropy assessment. *CHES*, 2013. https://hal.archives-ouvertes.fr/ujm-00859906/document.

[CK85] R. J. Cook and H. J. Kimble. Possibility of direct observation of quantum jumps. *Physical Review Letters*, 1985. http://authors.library.caltech.edu/10990/1/COOprl85.pdf.

[Cor14] Intel Corporation. Intel Digital Random Number Generator (DRNG), 2014. https://software.intel.com/sites/default/files/managed/4d/91/DRNG_Software_Implementation_Guide_2.0.pdf.

[CP05] W. Coppock and C. Philbrook. A mathematical and physical analysis of circuit jitter with application to cryptographic random bit generator. 2005. http://users.wpi.edu/~martin/MQP/cp05MQP.pdf.

[CTDRG12a] C. Cohen-Tannoudji, J. Dupont-Roc, and G. Grynberg. *Photons et atomes. Introduction à l'électrodynamique quantique : Introduction à l'électrodynamique quantique.* EDP Sciences, 2012.

[CTDRG12b] C. Cohen-Tannoudji, J. Dupont-Roc, and G. Grynberg. *Processus d'interaction entre photons et atomes.* EDP Sciences, 2012.

[DM13] M. Dichtl and B. Meyer. Apparatus and method for generating a random bit sequence, 2013. https://www.google.com/patents/US8410857.

[DMR00] A. Demir, A. Mehrotra, and J. Roychowdhury. Phase noise in oscillators : a unifying theory and numerical methods for characterization. *IEEE Trans. Circuits Syst. I*, 47 :655–674, 2000. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.1882&rep=rep1&type=pdf.

[Fey70] Richard P. Feynman. *Feynman lectures on physics.* Addison-Wesley, second edition, 1970.

[For73] G. D. Forney. The Viterbi algorithm. *Proceedings of IEEE*, 61(3) :268–278, 1973. http://arxiv.org/pdf/cs/0504020.pdf.

[FWN+10] M. Fürst, H. Weier, S. Nauerth, D. G. Marangon, C. Kurtsiefer, and H. Weinfurter. High speed optical quantum random number generation. *Opt. Express*, 2010. http://xqp.physik.uni-muenchen.de/publications/files/articles_2010/optexpress_18_13029.pdf.

[Gro15] The Pari Group. *PARI/GP, version* 2.8.0, 2015. http://pari.math.u-bordeaux.fr/.

[Gt14] T. Granlund and the GMP development team. *GNU MP : The GNU Multiple Precision Arithmetic Library*, 6.0.0 edition, 2014. https://gmplib.org/manual/Multiplication-Algorithms.html.

[Had15] P. Haddad. *Caractérisation et modélisation de générateurs de nombres aléatoires dans les circuits intégrés logiques*. PhD thesis, Université Jean Monnet, 2015. ...

[Ham09] J. Hamon. *Asynchronous oscillators and architectures for UWB impulse radio signal processing*. PhD thesis, Institut National Polytechnique de Grenoble - INPG, 2009. https://tel.archives-ouvertes.fr/tel-00481841/file/oaa_0310.pdf.

[Har11] L. Hars. Random number generation based on oscillatory metastability in ring circuits, 2011. https://eprint.iacr.org/2011/637.pdf.

[Hen12] N. Heninger. There's no need to panic over factorable keys–just mind your ps and qs, 2012. https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs/

[Iss11] O. El Issati. *Oscillateurs Asynchrones en Anneau : de la Théorie à la Pratique*. PhD thesis, Université de Grenoble, 2011. http://tima.imag.fr/publications/files/th/2011/sro_0356.pdf.

[JAW+00] T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 2000. http://dx.doi.org/10.1063/1.1150518.

[JR85] B.H. Juang and L.R. Rabiner. A probabilistic distance measure for hidden markov models. *AT T Technical Journal*, 1985.

[KGMS12] W. Khreich, E. Granger, A. Miri, and R. Sabourin. A survey of techniques for incremental learning of hmm parameters, 2012. http://www.sciencedirect.com/science/article/pii/S002002551200120X.

[Knu97] D. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.) : Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1997. http://e-x-a.org/stuff/books/knuth/Addison.Wesley.Donald.E.Knuth.The.Art.of.Computer.Programming.Volume.2.pdf.

[KS11] W. Killmann and W. Schindler. A proposal for : Functionality classes for random number generators, 2011. http://www.ibbergmann.org/AIS31_Functionality_classes_for_random_number_generators.pdf.

[Kul68] S. Kullback. *Information theory and statistics*. Courier Corporation, 1968.

[LHA+12] A. K. Lenstra, J. P. Hughes, M. Augier, J. W. Bos, T. Kleinjung, and C. Wachter. Ron was wrong, whit is right, 2012. https://eprint.iacr.org/2012/064.pdf.

[LRS83] S.E. Levinson, L.R. Rabiner, and M.M. Sondhi. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *Bell System Technical Journal, The*, 1983. http://www.isib.cnr.it/control/finesso/bstj62-4-1035.pdf.

[LS07] P. L'Ecuyer and R. Simard. Testu01 : A c library for empirical testing of random number generators. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, 2007. http://www.iro.umontreal.ca/~lecuyer/myftp/papers/testu01.pdf.

[Mar95] G. Marsaglia. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, 1995. http://www.stat.fsu.edu/pub/diehard/.

[Mau92] U. Maurer. A universal statistical test for random bit generators. *Journal of cryptology*, 1992. ftp://ftp.inf.ethz.ch/pub/crypto/publications/Maurer92a.pdf.

[MLC+14] Y. Ma, J. Lin, T. Chen, C. Xu, Z. Liu, and J. Jing. Entropy evaluation for oscillator-based true random number generators. In *Cryptographic Hardware and Embedded Systems – CHES 2014*. 2014. http://dx.doi.org/10.1007/978-3-662-44709-3_30.

[MT02] G. Marsaglia and W.W. Tsang. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 2002. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.156.7149&rep=rep1&type=pdf.

[NIS02] Fips pub 140-2, security requirements for cryptographic modules, 2002. U.S.Department of Commerce/National Institute of Standards and Technology.

[NS02] P.Q. Nguyen and I.E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 2002. http://link.springer.com/article/10.1007/s00145-002-0021-3.

[NS03] P.Q. Nguyen and I.E. Shparlinski. The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, codes and cryptography*, 2003. http://link.springer.com/article/10.1023%2FA%3A1025436905711.

[Oll] Y. Ollivier. Aspects de l'entropie en mathématiques. http://www.yann-ollivier.org/entropie/entropie.pdf.

[Ove10] Fail Overflow. Consol hacking 2010 : Ps2 epic fail, 2010. https://events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C (2Nd Ed.) : The Art of Scientific Computing*. Cambridge University Press, 1992. http://www2.units.it/ipl/students_area/imm2/files/Numerical_Recipes.pdf.

[QAS00] F. Qin, A. Auerbach, and F. Sachs. A direct optimization approach to hidden markov modeling for single channel kinetics. *Biophysical Journal 79*, 2000. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1301083/pdf/11023897.pdf.

[Qua10a] ID Quantique. Quantis-oem application note. 2010. http://www.idquantique.com/images/stories/PDF/quantis-random-generator/quantis-appnote.pdf.

[Qua10b] ID Quantique. Random number generation using quantum physics. 2010. http://www.idquantique.com/images/stories/PDF/quantis-random-generator/quantis-whitepaper.pdf.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of IEEE*, volume 77, pages 257–286. IEEE, 1989. http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf.

[RG09] G. Ribordy and O. Guinnard. Method and apparatus for generating true random numbers by way of a quantum optics process, 2009. http://www.google.com.ar/patents/US7519641.

[RJLS85] L.R. Rabiner, B.H. Juang, S.E. Levinson, and M.M. Sondhi. Some properties of continuous hidden markov model representations. *AT T Technical Journal*, 1985.

[RSN+01] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, J. Dray, S. Vo, M. Smid, M. Vangel, A. Heckert, and L.E. Bassham Iii. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001. `http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22b.pdf`.

[Ré61] A. Rényi. On measures of entropy and information. University of California Press, 1961. `http://l.academicdirect.org/Horticulture/GAs/Refs/Renyi_1961.pdf`.

[Sch03] Werner Schindler. A stochastical model and its analysis for a physical random number generator presented at ches 2002. pages 276–289, 2003. `http://dx.doi.org/10.1007/978-3-540-40974-8_22`.

[SDF11] Martin Simka, Milos Drutarovský, and Viktor Fischer. Testing of PLL-based True Random Number Generator in ChangingWorking Conditions. *RADIOENGINEERING,*, 2011. `https://hal-ujm.archives-ouvertes.fr/ujm-00667723/file/2011_Radioengineering_viktor.pdf`.

[Sel66] E.S. Selmer. *Linear recurrence relations over finite fields.* Department of Mathematics, University of Bergen, 1966.

[SGG+00] A. Stefanov, N. Gisin, O. Guinnard, L. Guinnard, and H. Zbinden. Optical quantum random number generator. *J.Mod.Opt.*, 2000. `http://arxiv.org/pdf/quant-ph/9907006v1.pdf`.

[Sha01] C.E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 2001. `http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf`.

[Sha04] R. Shaltiel. Recent developments in explicit constructions of extractors. 2004. `http://cshome.haifa.ac.il/~ronen/online_papers/survey.ps`.

[Sha11] R. Shaltiel. An introduction to randomness extractors, 2011. `http://www.cs.haifa.ac.il/~ronen/online_papers/ICALPinvited.pdf`.

[Sou12] M. Soucarros. *Analyse des générateurs de nombres aléatoires dans des conditions anormales d'utilisation.* PhD thesis, Université de Grenoble, 2012. `https://tel.archives-ouvertes.fr/tel-00759976/file/These.pdf`.

[Ste09] W. J. Stewart. *Probability, Markov chains, queues, and simulation.* 2009.

[Str60] R. Stratonovich. Conditional markov processes. *Theory of Probability & Its Applications*, 5(2) :156–178, 1960.

[TBM08] C. Tokunaga, D. Blaauw, and T. Mudge. True random number generator with a metastability-based quality control. *Solid-State Circuits, IEEE Journal of*, 43 :78–85, Jan. 2008. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4443211.

[Tea15a] The OpenSSL Development Team. *OpenSSL Project*, 1.0.2a edition, 2015. https://www.openssl.org/docs/apps/openssl.html.

[Tea15b] The OpenSSL Development Team. *OpenSSL Project, Pseudo random generator*, 2015. https://www.openssl.org/docs/manmaster/crypto/rand.html.

[Tre01] L. Trevisan. Extractors and pseudorandom generators. *J. ACM*, 2001. http://doi.acm.org/10.1145/502090.502099.

[Tur08] R. Turner. Direct maximization of the likelihood of a hidden markov model. *Computational Statistics and Data Analysis 52*, 2008. http://www.sciencedirect.com/science/article/pii/S0167947308000200.

[VD10] M. Varchola and M. Drutarovsky. New high entropy element for fpga based true random number generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 351–365. 2010. http://www.iacr.org/archive/ches2010/62250341/62250341.pdf.

[VDM] M. Varchola, M. Drutarovsky, and Repka M. Robust fpga based true random number generator utilizing oscillatory metastability in transition effect ring oscillators. http://www.wseas.us/e-library/conferences/2015/Dubai/CSST/CSST-12.pdf.

[VHYSK08] I. Vasyltsov, E. Hambardzumyan, K. Young-Sik, and B. Karpinskyy. Fast digital trng based on metastable ring oscillator. 5154 :164–180, 2008. https://www.iacr.org/archive/ches2008/51540162/51540162.pdf.

[Vit67] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2) :260–269, April 1967. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1054010&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1054010.

[vN51] J. von Neumann. Various techniques used in connection with random digits. *J. Res. Nat. Bur. Stand.*, 1951. `https://dornsifecms.usc.edu/assets/sites/520/docs/VonNeumann-ams12p36-38.pdf`.

[WG09] W. Wei and H. Guo. Bias-free true random-number generator. *Opt. Lett.*, 2009. `http://arxiv.org/pdf/0905.0779v2.pdf`.

[XUP05] L. Xie, V.A. Ugrinovskii, and I.R. Petersen. Probabilistic distances between finite-state finite-alphabet hidden markov models. *Automatic Control, IEEE Transactions on*, 2005. `http://www.ent.mrt.ac.lk/iml/paperbase/TAC%20Collection/TAC/2005/april/8.pdf`.