Générateurs de nombres pseudo-aléatoires utilisant des récurrences linéaires modulo 2

Présentation du sujet de thèse

Revue de la littérature et
identification des objectifs de recherche

14 août 2002

François Panneton
Département d'informatique et
de recherche opérationnelle
Université de Montréal

Table des matières

1 Introduction **4**

1.1 Structure générale des algorithmes de génération de nombres aléatoires . . . 4

1.2 Qualités d'un générateur de nombres pseudo-aléatoires 5

1.3 Générateurs utilisant une récurrence linéaire modulo 2 6

1.4 Équidistribution 7

1.5 Simulation Monte Carlo et intégration quasi-Monte Carlo 7

1.6 Plan du document 8

2 Générateurs à récurrences linéaires modulo 2 **10**

2.1 Propriétés générales des générateurs à récurrences linéaires modulo 2 10

2.2 Générateurs à récurrence linéaires existants 12

2.2.1 Générateur GFSR 12

2.2.2 Générateur de Tausworthe 12

2.2.3 Générateur à congruence linéaire polynômial 14

2.2.4 Générateur TGFSR 14

2.2.5 Générateur Mersenne Twister 15

2.2.6 "Multiple Recursive Matrix Method" 16

2.2.7 Générateur basé sur un automate cellulaire 17

3 Équidistribution **19**

3.1 Calcul de l'équidistribution par la méthode matricielle 21

3.2 Calcul de l'équidistribution par des réseaux 21

3.2.1 Réseau en dimension 22

3.2.2 Réseau en résolution 23

3.2.3 Réseau dual 24

3.3 "Tempering" permettant d'améliorer l'équidistribution 25

3.4 Réseaux digitaux 25

4 Description du projet de recherche **27**

4.1 Étude de générateurs MRMM basés sur des polynômes dans $\mathbb{F}_{2^w}$ 27

4.2 Étude de générateurs à congruence linéaire dans $\mathbb{F}_{2^w}[z]/P(z)$ 29

4.3 Variations 30

4.4 Étude de l'équidistribution des automates cellulaires 31

4.5 Recherche de règles de réseaux digitaux extensibles 31

4.6 Trouver de bons (q, k, t)-réseaux digitaux 35

4.7 Accélération du calcul de la base du réseau dual 35

4.8 Implantation des méthodes dans le progiciel REGPOLY 36

4.9 Étude de l'utilisation de pré-calculs afin d'accélérer les implantations des générateurs 37

4.10 Développement de librairies contenant de bons générateurs 39

5 Conclusion **41**

A Arithmétique dans $\mathbb{F}_{2^w}$ **42**

B Définition de réseau polynômial **43**

1 Introduction

Ce que je propose d'étudier dans le cadre de mon doctorat est la génération de nombres pseudo-aléatoires et d'ensembles de points uniformes grâce à des récurrences linéaires modulo 2. Un générateur de nombres pseudo-aléatoires est un algorithme déterministe qui tente d'imiter le hasard. Ces algorithmes sont utilisés surtout pour la simulation numérique par ordinateur, aussi appelée simulation Monte Carlo.

Dans cette introduction, je propose de vous familiariser aux générateurs de nombres pseudo-aléatoires. Tout d'abord, je vous donnerai une définition de générateur de nombres pseudo-aléatoires et des qualités que devrait posséder un bon. Ensuite, je discuterai du type de générateur que je vais étudier et du critère d'équidistribution. Ce dernier permet d'évaluer la qualité d'un générateur. Par après, je parlerai brièvement de la manière dont les générateurs de nombres pseudo-aléatoires sont utilisés pour la simulation Monte Carlo. Finalement, je décrirai les différentes parties de ce document. À la fin de cette introduction, j'espère que vous aurez une bonne idée du thème général de mon projet de recherche.

1.1 Structure générale des algorithmes de génération de nombres aléatoires

Il existe plusieurs types d'algorithmes déterministes pour la génération de nombres pseudo-aléatoires (afin d'alléger le texte on désignera un générateur de nombres pseudo-aléatoire par *générateur de nombres aléatoires* ou simplement *générateur*). Dans [10], on trouve un excellent compte-rendu des principaux algorithmes de génération de nombres aléatoires. Également, on explique une structure commune à tous les générateurs qui se résume à la définition suivante.

Définition 1.1 *(L'Ecuyer [10])*
Soit un espace fini d'états S, une loi de probabilité P qui permet de choisir un élément de S, une fonction $f : S \to S$, appelée fonction de transition, et une fonction $g : S \to U \subset \mathbb{R}$, appelée fonction de sortie. On appelle générateur de nombres pseudo-aléatoires un algorithme qui choisit un élément $s_0 \in S$, appelé germe, grâce à la loi de probabilité P, et qui produit une séquence $u_n = g(s_n) \in U, n \geq 0$, où $s_n = f(s_{n-1})$, qui tente d'imiter une séquence de variables aléatoires indépendantes et uniformes sur U. On note ce générateur $G = (S, P, f, U, g)$.

Le générateur produit, à chacune des itérations, un nouvel état $s_n \in S$ avec f et une sortie $u_n \in U$, obtenue avec g. Pour la très grande majorité des générateurs de nombres aléatoires, $U = [0, 1)$. Remarquons que, puisque le nombre d'états est fini, la séquence des nombres en sortie (les u_n) est périodique. La période maximale (la période étant le nombre de valeurs produites avant que la séquence ne se répète) d'un générateur est $|S|$.

Malheureusement, le simple fait de choisir une structure compatible avec cette définition ne garantit pas que la séquence des valeurs produites soit de bonne qualité, c'est-à-dire qu'elle

imite bien une séquence de nombres vraiment aléatoires. Pour que la séquence $u_0, u_1, \ldots$ soit de qualité, il faut choisir S, f et g judicieusement. Ce choix doit s'appuyer sur des arguments théoriques et empiriques solides. Il existe des générateurs contenus dans des programmes et des librairies informatiques couramment utilisés qui sont de piètre qualité [13]. La difficulté de trouver de bons S, f et g justifie ce projet de recherche. Je tenterai de choisir de bons S, f et g pour une classe spécifique de générateurs que je décris dans la section 1.3.

1.2 Qualités d'un générateur de nombres pseudo-aléatoires

Cette section discute des qualités que devraient avoir un générateur. Celles-ci permettent de nous guider dans le choix de S, f et g. Au risque de me répéter, les générateurs de nombres pseudo-aléatoires doivent être construits de façon soigneuse afin de bien imiter une source de hasard parfaite. La très grande majorité des générateurs de nombres pseudo-aléatoires tentent de reproduire une suite de variables aléatoires uniformes et indépendantes dans l'intervalle $[0, 1)$. Ce type de variable aléatoire est très utile puisqu'elle permet de reproduire des variables aléatoires qui suivent n'importe quelle loi de probabilité [9].

Dans [12], les auteurs décrivent les principaux critères permettant de définir ce qu'est un bon générateur de nombres aléatoires. Ces critères sont :

1. bonnes propriétés statistiques et d'uniformité : on désire obtenir une séquence de nombres $u_0, u_1, \ldots$ qui passe avec succès la plupart des tests statistiques raisonnables ;
2. longue période : supposons que l'on ait besoin de N valeurs aléatoires pour une simulation, alors il faut que la période des valeurs produites soit beaucoup plus grande que N ;
3. efficacité : le temps de calcul nécessaire afin de produire les valeurs doit être le plus petit possible par rapport au temps de la simulation. Ce facteur devient important dans des simulations qui prennent plusieurs jours d'exécution ;
4. répétabilité : l'utilisateur doit être capable de reproduire la même séquence de nombres facilement. Ceci est important pour la vérification des programmes et pour certaines techniques de réduction de variance.
5. facilité d'implantation et séparabilité : le générateur doit pouvoir être exécuté sur tous les types d'ordinateurs standards. Également, il doit être facile de "sauter" d'une valeur u_n à une autre u_{n+s} facilement, même quand s est grand. De cette manière, on peut utiliser plusieurs sous-séquences de la séquence $u_0, u_1, ..$ et considérer chaque sous-séquence comme un générateur indépendant (ceci exige que la période de la séquence $u_0, u_1, ..$ soit assez grande pour le permettre).

Dans la recherche que je propose, je vais construire de nouveaux algorithmes de génération de nombres aléatoires qui sont bons par rapport à chacun de ces critères.

1.3 Générateurs utilisant une récurrence linéaire modulo 2

Dans cette section, en partant de la définition des générateurs de nombres aléatoires introduite à la section 1.1, on définit l'espace d'états S, la fonction de transition f et la fonction de sortie g utilisés pour les générateurs à récurrence linéaire modulo 2. Cette classe de générateur est celle à laquelle je m'intéresse dans le cadre de mon doctorat.

Pour cette classe de générateurs, l'espace d'états S est $\mathbb{F}_2^k$, $\mathbb{F}_2$ étant le corps fini à deux éléments [21]. Cet espace d'états est représenté par un vecteur de k bits. Le n-ième état, s_n, est noté $\mathbf{x}_n = (x_n^{(0)}, x_n^{(1)}, \ldots, x_n^{(k-1)})^T$. Il est à remarquer que pour la suite de ce document, un symbole mathématique mis en caractère gras représente un vecteur de bits, tandis que le même symbole (en caractère normal) avec comme exposant un autre symbole mis entre parenthèses représente un bit particulier du vecteur de bits. La numérotation des bits commence par 0 et va de gauche à droite.

La fonction de transition $f : S \to S$, est une multiplication par une matrice X, qu'on appelle *matrice de transition*, de dimension $k \times k$ avec coefficients dans $\mathbb{F}_2$. La transition se fait par

$$\mathbf{x}_n = X\mathbf{x}_{n-1}. \tag{1}$$

Il s'agit d'une multiplication ordinaire d'un vecteur par une matrice, sauf que les multiplications et les additions se font dans $\mathbb{F}_2$.

Afin d'obtenir la sortie, on a besoin d'un vecteur de L bits $\mathbf{y}_n$ que l'on définit par

$$\mathbf{z}_n = B\mathbf{x}_n \tag{2}$$

$$\mathbf{y}_n = Y\mathbf{z}_n \tag{3}$$

où $\mathbf{z}_n = (z_n^{(0)}, z_n^{(1)}, \ldots, z_n^{(k-1)})^T$, $\mathbf{y}_n = (y_n^{(0)}, y_n^{(1)}, \ldots, y_n^{(L-1)})^T$, B est une matrice $k \times k$, Y est une matrice $L \times k$ et L est appelé *résolution de sortie du générateur*. La matrice B est la matrice de *tempering* [26] et est de dimension $k \times k$. Cette matrice permet d'améliorer l'équidistribution d'un générateur. Ceci est expliqué plus en détails dans la section 3.3. La matrice Y est la matrice qui décide du nombre de bits de résolution que le générateur aura à sa sortie. Si $L \leq k$, alors celle-ci est la matrice identité $k \times k$ (I_k) à laquelle on enlève les $k - L$ dernières lignes. Ce type de matrice identité tronquée est noté $I_{L\times k}$. Si $L > k$, alors la matrice Y est une matrice identité à laquelle on rajoute des lignes. La sortie $u_n \in [0, 1]$ associée à $\mathbf{x}_n$ est

$$u_n = \sum_{i=1}^{L} y_n^{(i-1)} 2^{-i}. \tag{4}$$

Les différents générateurs qui seront étudiés ont tous cette structure, seules leurs matrices X, B et Y respectives les différencient. Il est à remarquer que, dans la définition donnée à la section 1.1, la fonction de sortie $g : S \to U$ est la combinaison des équations (2), (3) et (4).

1.4 Équidistribution

Dans la section 1.2, on énumère différents critères permettant de juger de la qualité des générateurs de nombres aléatoires. Le premier critère est que le générateur doit avoir de bonnes propriétés statistiques et d'uniformité. L'équidistribution est une mesure de l'uniformité des points générés dans l'hypercube $[0,1)^t$.

Considérons l'ensemble Ω_t de tous les points à t dimensions produits par les valeurs successives d'un générateur, à partir de tous les états initiaux possibles $\mathbf{x}_0$, défini par

$$\Omega_t = \{\vec{u}_{0,t} = (u_0, \ldots, u_{t-1}) : \mathbf{x}_0 \in \mathbb{F}_2^k\}.$$

Maintenant, supposons que l'on partitionne l'hypercube $[0,1)^t$ en $2^{t\ell}$ petits hypercubes de même grandeur. La plus grande valeur de ℓ pour laquelle chacun des petits hypercubes contient le même nombre de points est appelée la *résolution* en dimension t et est notée ℓ_t. On dit que Ω_t (et par le fait même, le générateur) est *(t,ℓ)-équidistribué* si tous les petits hypercubes contiennent le même nombre de points, soit $2^{k-t\ell}$ points. Ceci est possible seulement si $k \geq t\ell$, parce que la cardinalité de Ω_t est au plus 2^k. La séquence $u_0, u_1, \ldots$ (ainsi que le générateur qui la produit) est dite *équidistribuée au maximum (ME)*, si ℓ_t atteint sa borne supérieure pour toutes les valeurs de t possibles, c'est-à-dire $\ell_t = \ell_t^*$ où $\ell_t^* \stackrel{\text{def}}{=} \min(L, \lfloor k/t \rfloor)$ pour $t = 1, \ldots, k$. On définit aussi *l'écart en dimension t* par $\Lambda_t \stackrel{\text{def}}{=} \ell_t^* - \ell_t$. On dit que le générateur a une *résolution maximale en dimension t* si $\Lambda_t = 0$. Selon ces définitions, on peut dire que le générateur est équidistribué au maximum (ME), si $\Lambda_t = 0$ pour $t = 1, \ldots, k$.

Les générateurs qui seront trouvés dans le cadre de ma recherche auront des valeurs de Λ_t petites. Ce critère sera celui principalement utilisé pour la recherche de bons générateurs. Un autre, qui prend compte de l'équidistribution de vecteurs de valeurs non successives et dont on fait mention dans la section 3, sera aussi utilisé pour la sélection de bons générateurs. L'équidistribution est utilisée pour sélectionner les générateurs à récurrence linéaire modulo 2 parce qu'il est facile à calculer. Les algorithmes de calculs de l'équidistribution prennent avantage de la linéarité des générateurs. Une description de ces algorithmes est donnée dans la section 3. Un autre critère, qui s'apparente à l'équidistribution, est la q-valeur du (q,k,t)-réseau digital que l'ensemble Ω_t décrit. Cette notion sera expliquée plus en détails dans la section 3.4.

1.5 Simulation Monte Carlo et intégration quasi-Monte Carlo

Beaucoup de simulations numériques par ordinateur qui utilisent des variables aléatoires uniformes peuvent se voir comme une intégration. Soit μ, la valeur que l'on désire estimer par la simulation et $f(\vec{u})$, le résultat d'une simulation étant donné le vecteur de t variables aléatoires uniformes $\vec{u}$ que l'on donne en entrée à la simulation. On peut voir la simulation

comme une méthode d'approximation de l'intégrale, sur l'hypercube unitaire $(0,1]^t$, d'une fonction réelle f, donnée par

$$\mu = \int_{(0,1]^t} f(\vec{u})\, d\vec{u}.$$

De façon générale, la simulation Monte Carlo fonctionne de la manière suivante. On prend un ensemble de points $P_n = \{\vec{u}_1, \ldots, \vec{u}_n\}$ qui est un ensemble de n vecteurs $\vec{u}_i$ indépendants et uniformément distribués sur $(0,1]^t$ et on prend la moyenne de f sur tous les points de P_n,

$$R_n = \frac{1}{n} \sum_{i=1}^{n} f(\vec{u}_i),$$

comme une approximation de μ. Il est possible de calculer un intervalle de confiance pour cette estimateur à l'aide de la variance empirique de R_n.

Il existe un autre type d'algorithme qui tente d'évaluer la même intégrale, mais avec un ensemble de points P_n plus uniforme que l'ensemble de points uniformément distribués de la simulation Monte Carlo. Cette technique est l'intégration quasi-Monte Carlo.

Une manière d'obtenir cet ensemble très uniforme est d'utiliser un générateur de nombres pseudo-aléatoires qui a un petit ensemble d'état $\|S\| = n$ (par exemple, $n = 2^{12}$). On nomme un générateur qui possède cette propriété *petit générateur*. L'ensemble de points est

$$P_n = \{\vec{u}_i = (u_0, u_1, \ldots, u_{t-1}) : s_0 \in S\}$$

qui est l'ensemble des vecteurs de t valeurs successives obtenus à partir de tous les états initiaux.

Les générateurs que je vais trouver pourront être utilisés pour les deux applications. Des générateurs à longues périodes (ou grands générateurs) pourront être utilisés pour la simulation Monte Carlo et des petits générateurs trouveront leur utilité pour l'intégration quasi-Monte Carlo. On peut choisir les générateurs selon différents critères d'uniformité de P_n. Parmi ceux-ci, on retrouve l'équidistribution. D'autres critères sont mentionnés plus loin dans ce document.

1.6 Plan du document

Dans les prochaines sections, on parlera des propriétés générales que possèdent tous les générateurs qui utilisent une récurrence linéaire modulo 2. On passera en revue les principaux types de générateurs à récurrence linéaire modulo 2 que l'on retrouve dans la littérature scientifique. On discutera du critère d'équidistribution et de la manière de le calculer. On conclura en donnant les objectifs précis de ce projet de recherche.

Voici une liste de mes principaux objectifs.

- Je veux étudier des générateurs qui utilisent $\mathbb{F}_{2^w}$, le corps fini à 2^w éléments. Aussi, je vais étudier d'autres générateurs qui ont une structure similaire, mais qui ne sont pas nécessairement basés sur des récurrences dans $\mathbb{F}_{2^w}$.
- L'équidistribution de différents types d'automates cellulaires (la définition de ce type de générateur est donnée dans la section 2.2) sera étudiée, ce qui n'a jamais été fait auparavant.
- Je veux aussi étudier les règles de réseaux extensibles basés sur des récurrences. Il s'agit d'une nouveauté et je compte développer la théorie sur ce type d'ensembles de points.
- Je ferai aussi des contributions pour la méthode de détermination de l'équidistribution par un réseau polynômial en résolution.
- Je développerai des librairies informatiques contenant de bons générateurs et de bons ensembles de points pour l'intégration quasi-Monte Carlo.
- Je continuerai le développement du progiciel REGPOLY[1] qui permet l'étude des propriétés d'uniformité pour les générateurs de nombres pseudo-aléatoires et pour des réseaux digitaux basés sur des récurrences. Je vais aussi modifier REGPOLY afin de tenir compte des réseaux digitaux généraux qui ne sont pas nécessairement basés sur des récurrences. De plus, je vais inclure des critères de recherche autre que l'équidistribution. Le développement de ce logiciel constituera une grande partie de mon travail.

[1] J'ai commencé le développement de ce progiciel au cours de ma maîtrise.

2 Générateurs à récurrences linéaires modulo 2

Dans cette section, on discute des propriétés des générateurs qui utilisent une récurrence linéaire modulo 2. On révise également les principaux générateurs de ce type que l'on retrouve dans la littérature scientifique.

2.1 Propriétés générales des générateurs à récurrences linéaires modulo 2

Il est maintenant question des propriétés générales des générateurs qui suivent le cadre défini par les équations (1)-(4). Mais avant, étudions plus en détails la récurrence dans $\mathbb{F}_2$

$$x_j = (a_1 x_{j-1} + ... + a_k x_{j-k}) \bmod 2, \tag{5}$$

où k est appelé l'ordre de la récurrence si $a_k \neq 0$ et $a_i \in \mathbb{F}_2$ pour $i = 1, \ldots, k$. Cette récurrence est très utile puisqu'elle nous permettra d'analyser tous les générateurs à récurrence linéaire modulo 2.

À cette récurrence est associé un polynôme dans $\mathbb{F}_2[z]$ que on appelle *polynôme caractéristique de la récurrence*. Ce polynôme est

$$P(z) = z^k - a_1 z^{k-1} - \cdots - a_k. \tag{6}$$

Grâce à ce polynôme, on peut déduire des informations importantes sur la récurrence. Une de ces informations est la période de la récurrence. En effet, il est bien connu que cette récurrence est de période maximale $2^k - 1$ si et seulement si le polynôme caractéristique est primitif [21].

Pour une initialisation donnée $x_0, \ldots, x_{k-1}$ de la récurrence (5), on associe la fonction génératrice

$$G(z) = x_0 z^{-1} + x_1 z^{-2} + \cdots + x_n z^{-n} + \cdots = \sum_{n=1}^{\infty} x_{n-1} z^{-n}.$$

Cette fonction n'en est pas une au sens propre, mais simplement une expression formelle. Elle fait partie de l'anneau des séries formelles de Laurent sur $\mathbb{F}_2$. Cet anneau est noté $\mathbb{F}_2[[z]]$ et comprend toutes les séries de la forme

$$\pi(z) = \alpha_n z^n + \alpha_{n-1} z^{n-1} + \cdots,$$

où $n \in \mathbb{Z}$ et $\alpha_i \in \mathbb{F}_2, i \leq n$.

On peut démontrer, en adaptant la preuve du théorème 6.40 de [21], le théorème suivant.

Théorème 2.1 *(adaptation de Lidl et Niederreiter [21])*
Soit $x_0, x_1, \ldots$ une séquence qui suit la récurrence (5), $P(z)$ le polynôme caractéristique de cette récurrence et $G(z) \in \mathbb{F}_2[[z]]$ sa fonction génératrice. L'identité

$$G(z) = \frac{g(z)}{P(z)} \tag{7}$$

est valide avec

$$g(z) = -\sum_{j=0}^{k-1} \sum_{i=0}^{k-1-j} a_{k-i-j-1} x_i z^j \in \mathbb{F}_2[z]$$

où l'on met $a_0 = -1$. De manière opposée, si $g(z)$ est un polynôme quelconque sur $\mathbb{F}_2$ avec $deg(g(z)) < k$ et si $P(z)$ est donné par (6), alors la série de Laurent $G(z)$ définie par (7) est la fonction génératrice de la récurrence linéaire modulo 2 donnée par (5).

Ce théorème nous permet d'associer à la récurrence (5), pour une initialisation donnée, un polynôme $g(z) \in \mathbb{F}_2[z]/P(z)$. L'anneau $\mathbb{F}_2[z]/P(z)$ est l'anneau[2] des polynômes modulo $P(z)$.

Ce qui suit tente de démontrer l'importance de la récurrence donnée par l'équation (5) pour l'analyse des générateurs de nombres aléatoires qui suivent une récurrence linéaire modulo 2. Pour ceci, on suppose que la matrice X est de plein rang ainsi que la matrice de tempering B. Dans ce cas, il est trivial de démontrer que la séquence des $\mathbf{z}_n, n \geq 0$ suit la récurrence

$$\mathbf{z}_n = BXB^{-1}\mathbf{z}_{n-1}. \tag{8}$$

Soit $\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \ldots$, une séquence de vecteurs qui suit la récurrence donnée par l'équation (8). Également, soit $\gamma_i = (z_0^{(i)}, z_1^{(i)}, z_2^{(i)}, \ldots)$, la séquence de bits observée au bit i des vecteurs $\mathbf{z}_n, n \geq 0$, pour un $\mathbf{z}_0$ donné. On peut démontrer [31] que, pour toute récurrence qui suit l'équation (8), chaque séquence $\gamma_i, 1 \leq i \leq k$ suit une récurrence linéaire donnée par l'équation (5), mais avec une initialisation différente. On détermine les coefficients a_i, $i = 1, \ldots, k$, en calculant le polynôme caractéristique $P(z)$ de la matrice X. Puisque chaque séquence $\gamma_i, 1 \leq i \leq k$ suit une récurrence linéaire du type (5), alors il est possible d'associer à chaque séquence une fonction génératrice $G_i(z) = g_i(z)/P(z)$.

Ainsi, pour une initialisation donnée $\mathbf{x}_0$ d'un générateur qui suit le cadre donné par les équations (1)-(4), on peut identifier, à l'aide du théorème 2.1, un unique vecteur de fonctions génératrices

$$(g_0(z), \ldots, g_{k-1}(z))/P(z)$$

qui caractérise entièrement la séquence des vecteurs $\mathbf{z}_n, n \geq 0$ et par le fait même, puisque B est de plein rang, la séquence des vecteurs $\mathbf{x}_n, n \geq 0$. Ce vecteur de fonctions génératrices sera important pour le calcul de l'équidistribution par la méthode du réseau en résolution. Cette technique sera vue en détails dans la section 3.

[2]Tout comme $\mathbb{Z}_p = \mathbb{Z}/p$ est l'anneau des entiers modulo p.

2.2 Générateurs à récurrence linéaires existants

A ce jour, il existe plusieurs types de générateurs à récurrence linéaire modulo 2 dans la littérature scientifique. Parmi les principaux, on retrouve le générateur de Tausworthe [43], le generalized feedback shift register (GFSR) [20], le twisted GFSR (TGFSR) [25, 26, 36], le Mersenne twister [28, 34], la Multiple Recursive Matrix Method [31, 32] et l'automate cellulaire [48]. Ce qui différencie ces générateurs est leur matrice X. Voici une brève description de chacun de ces types de générateurs.

2.2.1 Générateur GFSR

Un GFSR suit une récurrence de la forme

$$\mathbf{v}_n = a_1\mathbf{v}_{n-1} + ... + a_r\mathbf{v}_{n-r} \tag{9}$$

où les $\mathbf{v}_n, n \geq 0$ sont des vecteurs de w bits et les $a_i \in \mathbb{F}_2$, $i = 1, \ldots, k$.

L'état du générateur est le vecteur $\mathbf{x}_n = (\mathbf{v}_n^T, \cdots, \mathbf{v}_{n-r+1}^T)^T$. Il est facile de voir que chaque bit de $\mathbf{v}_n$ (et par le fait même, chaque bit de $\mathbf{x}_n$) suit la récurrence de base (5) avec $k = r$ et les mêmes valeurs de a_i.

Ainsi, on peut voir, à la lumière de la section 2.1, que tout générateur qui suit le cadre général décrit dans la section 1.3 peut être exprimé par une récurrence de type GFSR. Soit $P(z) = z^k - a_1z^{k-1} - \cdots - a_k$, le polynôme caractéristique de X et $G_i(z) = g_i(z)/P(z) = \sum_{j=0}^{\infty} g_{i,j}z^{-j}$, la fonction génératrice du i-ième bit des $\mathbf{x}_n, n \geq 0$. En mettant $r = k$, en initialisant les $\mathbf{v}_n = (g_{0,n}, \ldots, g_{k-1,n})$ pour $n = 0, \ldots, k-1$ et en prenant les coefficients de $P(z)$ comme valeurs de a_i dans (9), on obtient $\mathbf{v}_n = \mathbf{x}_n, n \geq 0$.

Pour un générateur à récurrence linéaire d'ordre k, et dont la sortie a L bits de résolution, l'état du générateur est de kw bits pour sa représentation par un GFSR. Optimalement, pour un générateur d'ordre k, l'état est de k bits. Ce désavantage fait en sorte que l'implantation d'un générateur par un GFSR n'est pas efficace sur le plan de la mémoire. Également, pour des raisons d'efficacité, on doit se restreindre à des GFSR dont le polynôme caractéristique $P(z)$ est un trinôme ou un pentanôme. Les générateurs basés sur un trinôme sont reconnus pour avoir de mauvaises propriétés statistiques [27, 26, 2]. En général, pour avoir de bonnes propriétés statistiques, le polynôme caractéristique doit avoir beaucoup de coefficients non nuls [1, 2]. Pour ces raisons, l'implantation par des GFSR n'est pas celle que l'on adoptera pour ce projet de recherche.

2.2.2 Générateur de Tausworthe

Le générateur de Tausworthe utilise explicitement la récurrence de base (5). Soit $x_0, x_1, \ldots$, une séquence dans $\mathbb{F}_2$ qui suit l'équation (5). L'état du générateur de Tausworthe à la n-ième itération est

$$\mathbf{x}_n = (x_{ns}, x_{ns+1}, \ldots, x_{ns+k-1})^T$$

où s est un paramètre entier. Habituellement, le vecteur de bits de la sortie associé à cet état du générateur est

$$\mathbf{y}_n = (x_{ns}, x_{ns+1}, \ldots, x_{ns+L-1}) \tag{10}$$

et on utilise la sortie définie par (4).

On peut facilement voir que ce générateur entre bien dans le cadre général défini antérieurement. La récurrence peut être écrite sous la forme

$$\mathbf{x}_n = \bar{X}^s_{\text{taus}} \mathbf{x}_{n-1}$$

où

$$\bar{X}_{\text{taus}} = \begin{pmatrix} & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ a_k & a_{k-1} & a_{k-2} & \ldots & a_1 \end{pmatrix}.$$

Afin d'obtenir la sortie (sans autre transformation linéaire), si $L > k$, on définit la matrice $Y = (I_k \quad C)^T$ où C permet d'obtenir les $L - w$ bits les moins significatifs de $\mathbf{y}_n$, sinon $Y = I_{L\times k}$. La matrice B, pour l'instant, est la matrice I_k.

Pour ce générateur, il n'est pas suffisant que le polynôme caractéristique de la matrice $\bar{X}_{\text{taus}}$ soit primitif pour obtenir une période maximale. Pour une valeur de s donnée, il est également nécessaire de satisfaire la condition $\text{pgcd}(s, 2^k - 1) = 1$.

Ce générateur peut aussi être vu comme une récurrence dans le corps $\mathbb{F}_2[[z]]$ des séries formelles de Laurent [5]. Soit

$$\pi(z) = \alpha_n z^n + \alpha_{n-1} z^{n-1} + \cdots, \tag{11}$$

un élément de $\mathbb{F}_2[[z]]$ où n est un entier et les opérateurs

$$\begin{aligned} \text{frac}(\pi(z)) &= \alpha_{-1} z^{-1} + \alpha_{-2} z^{-2} + \cdots, & (12) \\ \text{trunc}_l(\pi(z)) &= \alpha_n z^n + \alpha_{n-1} z^{n-1} + \cdots + \alpha_{-l} z^{-l} & (13) \end{aligned}$$

où $l > 0$ est un entier.

Si on initialise correctement le générateur de Tausworthe et la sortie est donnée par (10) et (4), on peut obtenir la même séquence $u_0, u_1, \ldots$ par la séquence $\tilde{u}_0 \tilde{u}_1, \ldots$ définie par

$$\begin{aligned} q_i &= \text{trunc}_L(\text{frac}(a(z)^i / P(z))) = \alpha_{i,1} z^{-1} + \ldots + \alpha_{i,L} z^{-L}, \quad i = 0, 1, 2, \ldots & (14) \\ \tilde{u}_i &= \sum_{j=1}^{L} \alpha_{i,j} 2^{-j}, \quad i = 0, 1, 2, \ldots & (15) \end{aligned}$$

et où $a(z) = z^s \bmod P(z) \in \mathbb{F}_2[z]/P(z)$, $P(z)$ est le polynôme caractéristique de la récurrence de base (5) et L est le nombre de bits voulus à la sortie. Cette manière de voir le générateur de Tausworthe permet de définir un réseau sur un sous-ensemble de $\mathbb{F}_2[[z]]^t$. Ce réseau est utile pour calculer l'équidistribution de ce type de générateur.

2.2.3 Générateur à congruence linéaire polynômial

Le générateur à congruence linéaire polynômial (ou GCL polynômial) peut être représenté par une récurrence dans $\mathbb{F}_2[z]/P(z)$ où $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$ est un polynôme primitif de degré k. L'état du générateur, $p_n(z)$, est un élément de $\mathbb{F}_2[z]/P(z)$ et suit la récurrence

$$p_n(z) = a(z)p_n(z) \bmod P(z) \tag{16}$$

où $a(z) = z^s$ est un élément de $\mathbb{F}_2[z]/P(z)$. Tout comme le générateur de Tausworthe, celui-ci est de période maximale si et seulement si $\text{pgcd}(2^k - 1, s) = 1$.

Si à chaque $p_n(z) = c_1 z^{k-1} + \cdots + c_k$ on associe un vecteur de bits $\mathbf{x}_n = (c_k, \ldots, c_1)$, la récurrence (16) peut être exprimée sous forme matricielle par

$$\mathbf{x}_n = \bar{X}^s \mathbf{x}_{n-1} \tag{17}$$

où

$$\bar{X}_{\text{gcl}} = \begin{pmatrix} & & & a_k \\ 1 & & & a_{k-1} \\ & 1 & & a_{k-2} \\ & & \ddots & \vdots \\ & & 1 & a_1 \end{pmatrix}.$$

Fait intéressant à remarquer, $\bar{X}_{\text{gcl}} = \bar{X}_{\text{taus}}^T$ pour un polynôme $P(z)$ donné. La différence entre les deux générateurs se voit dans la manière de générer le prochain $\mathbf{x}_n$ quand $s = 1$. Dans le cas du générateur de Tausworthe, on décale les bits $\mathbf{x}_{n-1}$ de 1 position vers la gauche et met à jour le dernier bit par une combinaison linéaire des bits de $\mathbf{x}_{n-1}$ pour obtenir $x_n^{(k-1)}$. Dans le cas du GCL polynômial, on décale les bits de $\mathbf{x}_{n-1}$ de 1 position vers la droite et on modifie plusieurs bits dépendamment si le dernier bit de $\mathbf{x}_{n-1}$ est nul ou pas. Le point important est que dans le cas du générateur de Tausworthe, on modifie 1 bit en observant plusieurs bits et dans le cas du GCL polynômial, on modifie plusieurs bits en observant 1 seul bit. On fera une remarque semblable lorsque l'on discutera des générateurs dans $\mathbb{F}_{2^w}$ dans la section 4.

2.2.4 Générateur TGFSR

Le TGFSR est basé sur la récurrence

$$\mathbf{v}_n = \mathbf{v}_{n+m-r} + A\mathbf{v}_{n-r}, \tag{18}$$

où A est une matrice binaire de dimension $w \times w$ et les $\mathbf{v}_i$ sont des vecteurs-colonnes. En choisissant de façon adéquate les valeurs de w, r, m et A, il est possible d'atteindre une période de $2^{rw} - 1$, qui est la période maximale pour ce type de générateur.

On observe que ce générateur entre dans le cadre général décrit à la section 1.3, puisque

$$\begin{pmatrix} \mathbf{v}_n \\ \mathbf{v}_{n-1} \\ \vdots \\ \mathbf{v}_{n-r+2} \\ \mathbf{v}_{n-r+1} \end{pmatrix} = X \begin{pmatrix} \mathbf{v}_{n-1} \\ \mathbf{v}_{n-2} \\ \vdots \\ \mathbf{v}_{n-r+1} \\ \mathbf{v}_{n-r} \end{pmatrix} \tag{19}$$

où

$$X = \begin{pmatrix} & & & I_w & & A \\ I_w & & & & & \\ & I_w & & & & \\ & & I_w & & & \\ & & & \ddots & & \\ & & & & I_w & \end{pmatrix} \tag{20}$$

et, l'élément $(0,0)$ de la matrice I_w de la première ligne de X est l'élément $(0,(r-m-1)w)$ de X. L'état du générateur est $\mathbf{x}_n = (\mathbf{v}_n^T, \mathbf{v}_{n-1}^T, \ldots, \mathbf{v}_{n-r+1}^T)^T$.

Les auteurs de [25] décrivent les avantages des TGFSR sur les GFSR. On y retrouve aussi le théorème suivant sur les conditions nécessaires et suffisantes pour d'atteindre la période maximale.

Théorème 2.2 *(Matsumoto et Kurita [25])*
Soit $\phi_A(t)$, le polynôme caractéristique de la matrice A. Le générateur a une période maximale de $2^{rw}-1$ si et seulement si $\phi_A(t^r+t^m)$ est un polynôme primitif de degré rw. Dans ce cas, chaque bit du vecteur $\mathbf{v}_i$ suit une récurrence de la forme (5), avec polynôme caractéristique $\phi_A(t^r+t^m)$.

2.2.5 Générateur Mersenne Twister

Le Mersenne twister a été introduit pour la première fois dans [28] et est une généralisation du TGFSR. La récurrence est de la forme

$$\mathbf{v}_n = \mathbf{v}_{n+m-r} + A(\mathbf{v}_{n-r}^h | \mathbf{v}_{n-r+1}^b). \tag{21}$$

Il y a 5 paramètres à cette récurrence : les entiers r et w, un entier p, $0 \le p \le w-1$, un entier m, $0 \le m \le r$, et une matrice A dont les entrées sont dans $\mathbb{F}_2$. Les vecteurs $\mathbf{v}_n$ sont des vecteurs-colonnes. Le symbole $\mathbf{v}_{n-r}^h$ représente les $w-p$ premiers bits de $\mathbf{v}_{n-r}$ tandis que $\mathbf{v}_{n-r+1}^b$ représente les p derniers bits de $\mathbf{v}_{n-r+1}$. L'opération | représente la concaténation des opérandes. Également, on observe que si $p=0$, la récurrence (21) devient (18), d'où la généralisation du TGFSR.

On démontre maintenant que ce générateur entre dans le cadre général défini à la section 1.3. L'état du Mersenne twister est représenté sur $rw-p$ bits. La période maximale de ce type de générateur est alors de 2^{nw-p}. L'état est

$$\mathbf{x}_n = (\mathbf{v}_n^T, \mathbf{v}_{n-1}^T, \ldots, \mathbf{v}_{n-r+2}^T, \mathrm{trunc}_{w-p}(\mathbf{v}_{n-r+1})^T)^T$$

où $\text{trunc}_{w-p}((\mathbf{v}_{n-r+1})^T)$ est le vecteur de $w-p$ bits composé des $w-p$ premiers bits de $(\mathbf{v}_{n-r+1})^T$. La matrice X de dimension $(rw-p)\times(rw-p)$ est

$$X = \begin{pmatrix} & & I_w & & S \\ I_w & & & & \\ & I_w & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & I_{w-p} \end{pmatrix} \tag{22}$$

où

$$S = A \begin{pmatrix} & I_{w-p} \\ I_p & \end{pmatrix}$$

est une matrice $w \times w$.

Le polynôme caractéristique de la récurrence dépend de la matrice A et une formule permettant de déterminer celui-ci est donnée dans [28].

2.2.6 "Multiple Recursive Matrix Method"

La "Multiple Recursive Matrix Method" (MRMM) est une autre généralisation du TGFSR. Par abus de langage, on parlera de *générateur MRMM* pour tout générateur qui utilise cette méthode. La récurrence est

$$\mathbf{v}_n = A_1\mathbf{v}_{n-1} + A_2\mathbf{v}_{n-2} + \cdots + A_r\mathbf{v}_{n-r} \tag{23}$$

où les $\mathbf{v}_n$ sont des vecteurs de w bits et les matrices A_i, $i = 1, \ldots, r$, sont de dimensions $w \times w$. L'état du générateur est $\mathbf{x}_n = (\mathbf{v}_n^T, \cdots, \mathbf{v}_{n-r+1}^T)^T$ et la matrice de transition de ce type de générateur est donnée par

$$X = \begin{pmatrix} A_1 & A_2 & \ldots & A_{r-1} & A_r \\ I_w & & & & \\ & I_w & & & \\ & & \ddots & & \\ & & & I_w & \end{pmatrix}. \tag{24}$$

Théorème 2.3 *(Niederreiter [30])*
N'importe quelle séquence générée par (23) a une période inférieure ou égale à $2^{rw}-1$. *De plus, la période est* $2^{rw}-1$ *si et seulement si*

$$det\left(z^r I_w + \sum_{i=1}^{r} z^{r-i} A_i\right) \tag{25}$$

est un polynôme primitif sur $\mathbb{F}_2$.

Ce type de générateur est une généralisation d'un générateur qui sera décrit dans la section 4. Ce dernier est une récurrence linéaire dans $\mathbb{F}_{2^w}$ similaire à la récurrence (5) qui est dans $\mathbb{F}_2$.

Il est à remarquer que dans [32], on utilise ce type de générateurs pour obtenir des vecteurs pseudo-aléatoires dans $[0,1)^w$. Soit la fonction $h : \mathbb{F}_2^w \to [0,1)^w$ définie par

$$h(\mathbf{v}) = (v^{(0)}, \ldots, v^{(w-1)})/2 \tag{26}$$

où l'on prend la valeur de chaque bit comme un nombre réel. Le n-ième vecteur en w dimension est obtenu par

$$\vec{u}_n = h(\mathbf{v}_n) \in [0,1)^w.$$

Il s'agit d'une approche bien différente de celle qu'on utilise. Avec cette méthode, on ne peut générer que des vecteurs qui ont pour coordonnées que les valeurs 0 et 1/2. Cette approche est utile si on remplace $\mathbb{F}_2$ par un corps fini qui a une cardinalité p plus grande. Dans ce cas, on remplace le 2 par p dans l'équation (26). Ceci permet de mieux couvrir chacune des coordonnées.

2.2.7 Générateur basé sur un automate cellulaire

Un automate cellulaire correspond à un tableau $M(t)$ de dimension $n \times m$ d'éléments dans $\mathbb{F}_2$ qui évolue dans le temps. On appelle un élément $m_{i,j}(t), 0 \leq i < n, 0 \leq j < m$ du tableau $M(t)$ *cellule*. Pour ce système, le temps évolue de façon discrète et à chaque incrément du temps, la valeur de chacune des cellules est modifiée selon la valeur des cellules voisines et selon des règles préétablies. Ce type de générateur est surtout utilisé dans la vérification de composantes électroniques. Ils sont implantés de façon matérielle plutôt que logicielle.

Malgré le fait que ce type de générateur soit répandu et bien connu par la communauté scientifique, rien n'a été fait au niveau de l'évaluation de l'équidistribution des points produits par ces générateurs. Ainsi, je tenterai de trouver de bons automates cellulaires pour la génération de nombres aléatoires.

Dans le cas où $n = 1$ et $m = k$, on obtient un automate cellulaire en 1 dimension. Pour les besoins de ce projet de recherche, nous ne considérerons que les règles qui correspondent à une transformation linéaire et dont le prochain état d'une cellule $(m_{1,j}(t+1))$ est déterminé à partir de l'état de ses voisins les plus proches. On spécifie par v la taille du voisinage. Si on note $x_t^{(j)} = m_{1,j}(t)$, ce type d'automate cellulaire suit la récurrence

$$x_t^{(j)} = \sum_{s=-v}^{v} a_{j,s} x_{t-1}^{(j+s)}, \quad j = 0, \ldots, k-1, \tag{27}$$

où les $a_{j,s} \in \mathbb{F}_2$ sont fixes, $x_t^{(\ell)} = 0$ pour $\ell \notin \{0, 1, \ldots, k-1\}$ et $a_{j,s} = 0$ si $(j,s) \notin \{(j,s) : 0 \leq j < k, -v \leq s \leq v\}$.

Par exemple, supposons que $v = 1$. Dans ce cas, l'état suivant d'une cellule $(x_{t+1}^{(j)})$ est déterminé à partir de l'état de ses voisins immédiats ($x_t^{(j-1)}$ et $x_t^{(j+1)}$) et de son état actuel

$(x_t^{(j)})$. Il est facile de voir que ce type d'automate cellulaire entre bien dans le cadre d'un générateur à récurrence linéaire modulo 2 défini dans la section 1.3. L'état de ce générateur est le vecteur de bits $\mathbf{x}_t = (x_t^{(0)}, \ldots, x_t^{(k-1)})$. La matrice X de transition est

$$X = \begin{pmatrix} a_{0,0} & a_{0,1} & & & & & \\ a_{1,-1} & a_{1,0} & a_{1,1} & & & & \\ & a_{1,-1} & a_{1,0} & a_{1,1} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & a_{k-2,-1} & a_{k-2,0} & a_{k-2,1} \\ & & & & & a_{k-1,-1} & a_{k-1,0} \end{pmatrix}. \tag{28}$$

Cette matrice est de dimension $k \times k$. Le polynôme caractéristique de ce type de générateur est le polynôme caractéristique de la matrice X.

Des exemples de règles linéaires courantes (quand $v = 1$) sont les règles 90 et 150. Si une cellule j suit la règle 90, alors celle-ci sera mise à jour par l'équation

$$x_t^{(j)} = x_{t-1}^{(j-1)} + x_{t-1}^{(j+1)}, \tag{29}$$

donc $a_{j,-1} = 1, a_{j,0} = 0, a_{j,1} = 1$ pour cette cellule. Si c'est la règle 150 qui est appliquée à la cellule j, alors celle-ci sera mise à jour par l'équation est

$$x_t^{(j)} = x_{t-1}^{(j-1)} + x_{t-1}^{(j)} + x_{t-1}^{(j+1)}, \tag{30}$$

donc $a_{j,-1} = 1, a_{j,0} = 1, a_{j,1} = 1$ pour cette cellule. Cette manière d'affecter un numéro à une règle a été introduite par Wolfram [48].

Dans la littérature, il existe beaucoup de types d'automates cellulaires à 1 dimension dont les règles sont linéaires. Certains ont pleine période $2^k - 1$ [47, 24].

Dans le cas où $n > 1$, on s'intéressera aux automates cellulaires linéaires où chaque cellule évolue selon une récurrence de la forme

$$m_{i,j}(t) = a_{i,j,0,0} m_{i,j}(t-1) + \sum_{(v,w)\in V} a_{i,j,v,w} m_{i+v,j+w}(t-1), \;\; 0 \le i < n, 0 \le j < m \tag{31}$$

où $V = \{(0,1),(1,0),(-1,0),(0,-1)\}$, les $a_{i,j,v,w} \in \mathbb{F}_2$, $m_{i,j}(t) = 0$ pour $i \notin \{0,1,\ldots,n-1\}$ ou $j \notin \{0,1,\ldots,m-1\}$ et $a_{i,j,v,w} = 0$ si $(i+v) \notin \{0,1,\ldots,n-1\}$ ou si $(j+w) \notin \{0,1,\ldots,m-1\}$. Dans cette récurrence, la taille du voisinage d'une cellule est 5 cellules, soit la cellule de gauche, de droite, du haut, du bas et elle-même. On appelle ce type d'automate cellulaire *automate cellulaire en 2 dimensions* [35], puisqu'il correspond à un tableau de valeurs en 2 dimensions qui évolue dans le temps.

Plus récemment, on a introduit des automates cellulaires dont la valeur que peut prendre chacune des cellules n'est pas dans $\mathbb{F}_2$, mais dans $\mathbb{F}_{2^p}$ pour une valeur de $p > 1$ [41, 38, 37]. Je m'intéresserai également à ce type d'automate cellulaire.

3 Équidistribution

Dans l'introduction, on explique brièvement ce qu'est le critère d'équidistribution. Comme mentionné, l'équidistribution est une mesure de l'uniformité de points générés dans $[0,1)^t$. Dans cette section, nous traitons plus en détails cette mesure d'uniformité.

Considérons l'ensemble Ω_t de tous les points à t dimensions produits par les valeurs successives d'un générateur, à partir de tous les états initiaux possibles $\mathbf{x}_0$, défini par

$$\Omega_t = \{\vec{u}_{0,t} = (u_0, \ldots, u_{t-1}) : \mathbf{x}_0 \in \mathbb{F}_2^k\}. \tag{32}$$

Partitionnons l'hypercube $[0,1)^t$ en $2^{t\ell}$ hypercubes de même grandeurs. La plus grande valeur de ℓ pour laquelle chacun des hypercubes contient $2^{k-t\ell}$ points est appelée la *résolution* en dimension t et est notée ℓ_t. On dit que Ω_t (et par le fait même, le générateur) est *(t,ℓ)-équidistribué* si tous les petits hypercubes contiennent $2^{k-t\ell}$ points. Ceci est possible seulement si $k \geq t\ell$, parce que la cardinalité de Ω_t est au plus 2^k. La séquence $u_0, u_1, \ldots$ (ainsi que le générateur qui la génère) est dite *équidistribuée au maximum (ME)*, si ℓ_t atteint sa borne supérieure pour toutes les valeurs de t possibles, c'est-à-dire $\ell_t = \min(L, \lfloor k/t \rfloor)$ pour $t = 1, \ldots, k$. Un ensemble de nombres complémentaires sont les $t_\ell, 1 \leq \ell \leq L$, où t_ℓ représente la plus grande valeur de t pour lequel Ω_t est (t,ℓ)-équidistribué pour une valeur de ℓ fixée. On a les bornes supérieures

$$\ell_t \leq \ell_t^* \stackrel{\text{def}}{=} \min(L, \lfloor k/t \rfloor) \quad \text{pour } t = 1, \ldots, k \tag{33}$$

et

$$t_\ell \leq t_\ell^* \stackrel{\text{def}}{=} \lfloor k/\ell \rfloor \quad \text{pour } l = 1, \ldots, L. \tag{34}$$

On définit aussi *l'écart en dimension t* par $\Lambda_t \stackrel{\text{def}}{=} \ell_t^* - \ell_t$ et *l'écart en résolution ℓ* par $\Delta_l \stackrel{\text{def}}{=} t_\ell^* - t_\ell$. On dit que le générateur a une *résolution maximale en dimension t* si $\Lambda_t = 0$. De plus, celui-ci sera dit de *dimension maximale en résolution ℓ*, si $\Delta_\ell = 0$. Selon ces définitions, on peut dire que le générateur est équidistribué au maximum (ME), si $\Lambda_t = 0$ pour $t = 1, \ldots, k$ ou, de façon équivalente, si $\Delta_\ell = 0$ pour $\ell = 1, \ldots, \min(k, L)$. L'équivalence peut être vue en constatant que, si pour une valeur de t on a $\ell_t < \ell_t^*$ il s'ensuit que $\Delta_{\ell_t^*} > 0$ et que si, pour une certaine valeur de ℓ, $t_\ell < t_\ell^*$, alors $\Lambda_{t_\ell^*} > 0$.

On a considéré l'ensemble de points Ω_t et la signification de l'équidistribution du générateur générant cet ensemble de points. Cet ensemble de points est généré par les vecteurs de valeurs successives produites par le générateur. L'équation (35) définit un critère d'uniformité plus rigoureux que la simple équidistribution et tente de mesurer l'uniformité d'ensembles de points lorsqu'on considère des projections dans des dimensions non successives. L'utilité de ce critère est bien illustré dans [18, 17] pour l'intégration quasi-Monte Carlo : afin de réduire l'erreur d'intégration, il est préférable d'obtenir une bonne équidistribution dans plusieurs ensembles de points formés par des t-tuplets de valeurs non successives du générateur. Dans [18], on définit le critère (35) qui est une généralisation du critère d'équidistribution et que l'on nomme *équidistribution des projections*.

On définit

$$\tilde{\Delta}_{t_1,\ldots,t_d} = \max\left(\max_{I=\{1,\ldots,s\},s=1,\ldots,t_1}(\ell^*_{|I|}(n)-\ell_I), \max_{I\in S(t_s,s),s=2,\ldots,d}(\ell^*_{|I|}(n)-\ell_I)\right), \tag{35}$$

où $S(t_s,s)=\{\{i_1,\ldots,i_s\}, 1=i_1<\ldots<i_s\leq t_s\}$. La quantité $\ell^*_{|I|}(n)$ est donnée par $\lfloor k/|I|\rfloor$ et correspond à la résolution maximale pour un ensemble de $n=2^k$ points dans $[0,1)^{|I|}$. L'ensemble I est un ensemble d'indices de dimensions.

La première partie du critère, soit

$$\max_{I=\{1,\ldots,s\},s=1,\ldots,t_1}(\ell^*_{|I|}(n)-\ell_I),$$

détermine l'écart maximal entre la résolution maximale, $\ell^*_{|I|}(n)$, et la résolution obtenue, ℓ_I, en prenant les ensembles de points produits par des s-tuplets de valeurs successives du générateur, soit

$$\{\vec{u}_s=(u_0,\ldots,u_{s-1}) : \mathbf{x}_0\in \mathbb{F}_2^k\}.$$

pour $s=1,\ldots,t_1$.

Une autre façon d'exprimer la première partie est par

$$\max_{t=1,\ldots,t_1}\Lambda_t.$$

La deuxième partie du critère, soit

$$\max_{I\in S(t_s,s),s=2,\ldots,d}(\ell^*_{|I|}(n)-\ell_I),$$

détermine également l'écart maximal entre la résolution maximale et la résolution obtenue mais en prenant des ensembles de points de valeurs non successives. Ces ensembles de points sont

$$\{\vec{u}=(u_{i_1},\ldots,u_{i_s}) : \mathbf{x}_0\in \mathbb{F}_2^k\},$$

pour $s=2,\ldots,d$ et $1=i_1<\ldots<i_s\leq t_s$. Le paramètre t_s permet de choisir, pour les s-tuplets, la valeur maximale de i_s et par le fait même, du point de vue pratique, de contrôler le nombre d'ensembles de points à vérifier. La valeur $\tilde{\Delta}_{t_1,\ldots,t_d}$ est la valeur maximale de $\ell^*_{|I|}(n)-\ell_I$ qu'on obtient dans les deux parties.

Il est important d'observer que ce critère $\tilde{\Delta}_{t_1,\ldots,t_d}$ généralise la propriété ME parce que le générateur est ME si et seulement si $\tilde{\Delta}_k=0$. Dans ce cas, on obtient

$$\tilde{\Delta}_k = \max_{I=\{1,\ldots,s\},s=1,\ldots,k}(\ell^*_{|I|}(n)-\ell_I) = \max_{t=1,\ldots,t_1}\Lambda_t.$$

Dans la définition que l'on a donnée d'un générateur ME, on dit qu'il est ME si $\Lambda_t=0$ pour $t=1,\ldots,k$. Puisque $\Lambda_t\geq 0$, alors l'équivalence entre $\tilde{\Delta}_k=0$ et la propriété ME devient évidente.

Dans les prochaines sous-sections, il est question des différentes façons de calculer l'équidistribution.

3.1 Calcul de l'équidistribution par la méthode matricielle

Le calcul de l'équidistribution d'un ensemble de points

$$\{\vec{u} = (u_{i_1}, \ldots, u_{i_t}) : \mathbf{x}_0 \in \mathbb{F}_2^k\}$$

pour un ensemble d'indices (successifs ou non successifs) $I = \{i_1, \ldots, i_t\}$ revient au calcul du rang de la matrice $B_{t,\ell}$ qui satisfait l'équation

$$\mathbf{s}_{t,\ell} = B_{t,\ell}\mathbf{x}_0 \tag{36}$$

où $\mathbf{s}_{t,\ell} = (\text{trunc}_\ell(\mathbf{y}_{i_1}), \ldots, \text{trunc}_\ell(\mathbf{y}_{i_t}))$ et la fonction $\text{trunc}_\ell(\mathbf{x})$ retourne le vecteur de ℓ bits $(x^{(0)}, \ldots, x^{(\ell-1)})$. La proposition 1 de [11] affirme que le générateur est (t, ℓ)-équidistribué si et seulement si la matrice $B_{t,\ell}$ est de plein rang $t\ell$. Dans [11], l'auteur explique comment calculer la matrice $B_{t,\ell}$. La méthode utilisée pour calculer le rang de la matrice $B_{t,\ell}$ est par l'élimination gaussienne. Le temps de calcul par cette méthode est dans $O(k^3)$. Ceci est utilisé pour des générateurs d'ordre plus petit que 1000.

3.2 Calcul de l'équidistribution par des réseaux

Le développement de cette méthode de calcul de l'équidistribution se trouve en détails dans Couture et al. [5]. Suivant ces auteurs et Mahler [22], on définit sur $\mathbb{F}_2[[z]]$, le corps des séries formelles de Laurent (voir section 2.1), la norme

$$|\pi(z)| \;=\; \begin{cases} 0 & \text{si } \pi(z) = 0 \\ 2^n & \text{si } \pi(z) \neq 0 \text{ et } \pi(z) \text{ est donné par (11) avec } \alpha_n \neq 0. \end{cases} \tag{37}$$

Par conséquent, l'espace vectoriel $\mathbb{F}_2[[z]]^t$, où t est un entier, est normé par

$$\|\Pi\| = \max_{1 \leq i \leq t} |\pi_i(z)|$$

où $\Pi = (\pi_1(z), \ldots, \pi_t(z))$.

On considère maintenant, dans $\mathbb{F}_2[[z]]$, le sous-anneau de polynômes $A = \mathbb{F}_2[z]$ et les A-sous-modules de $\mathbb{F}_2[[z]]^t$. On appelle un A-sous-module de $\mathbb{F}_2[[z]]^t$ un *réseau polynômial* (voir définition B.5).

Théorème 3.1 *(Mahler [23])*
Soit $\Pi_1, \ldots, \Pi_h$, des points dans le réseau polynômial $\mathcal{L} \subset \mathbb{F}_2[[z]]^t$ de rang h avec les propriétés suivantes :

1. *Π_1 est un plus court vecteur dans $\mathcal{L}$;*
2. *pour $i = 2, \ldots, h$, Π_i est un plus court vecteur parmi l'ensemble des vecteurs Π dans $\mathcal{L}$ tel que $\Pi_1, \ldots, \Pi_{i-1}, \Pi$ sont linéairement indépendants sur A.*

Alors $\Pi_1, \ldots, \Pi_h$ forme une base de $\mathcal{L}$ sur A.

Puisque n'importe quel cube $C_r^{(t)} = \{ \Pi \mid \|\Pi\| < 2^r \}$, $r \in \mathbb{Z}$, contient un nombre fini de points dans $\mathcal{L}$, ceci implique qu'un système tel que défini dans le théorème 3.1 existe toujours [5]. Ce système est une *base réduite* au sens de Minkowski. Les nombres $\sigma_i = \|\Pi_i\|$ sont uniquement déterminés par le réseau et appelés *minima successifs*. Un algorithme de Lenstra [19] permet de calculer les vecteurs Π_i de efficacement.

Ce type de réseau dans l'espace des séries formelles de Laurent nous permet de déterminer l'équidistribution. Il existe deux catégories de réseaux que décrivent les générateurs à récurrence linéaire modulo 2. Le premier réseau, soit le réseau en dimension, n'est engendré que pour certains types de générateurs, tel le générateur de Tausworthe. Le deuxième type de réseau, le réseau en résolution est produit par tous les générateurs à récurrence linéaire modulo 2. Ces deux catégories de réseaux et la manière de calculer l'équidistribution est le sujet des prochaines sous-sections.

3.2.1 Réseau en dimension

Dans le contexte des générateurs de Tausworthe, il est possible de décrire l'ensemble des points produits par un réseau.

Dans la section 2.2.2, on décrit la récurrence du générateur de Tausworthe par les équations (14) et (15). Si on prend tous les t-tuplets $(q_i, q_{i+1}, \ldots, q_{i+t-1}) \in \mathbb{F}_2[[z]]^t, 0 \leq i \leq 2^k - 1$, alors ceux-ci décrivent un réseau dans $\mathbb{F}_2[[z]]^t$. Ce réseau a pour base

$$\begin{aligned} B_1 &= (1, a(z), a(z)^2, \ldots, a(z)^{t-1})/P(z), \\ B_2 &= (0, 1, \ldots, 0), \\ &\cdots \\ B_t &= (0, 0, \ldots, 1). \end{aligned}$$

C'est un réseau polynômial de type Korobov [18]. Le théorème suivant montre comment ce réseau permet de déterminer l'équidistribution d'un générateur de Tausworthe.

Théorème 3.2 *(Tezuka [44])*
Un générateur de la forme (14) et (15) dans $\mathbb{F}_2[[z]]$ défini par $a(z)$ et un $P(z)$ irréductible sur $\mathbb{F}_2$ est (t, ℓ)-équidistribué si et seulement si $log_2 \sigma_t \leq -\ell$.

Bien que ce réseau soit utile pour déterminer l'équidistribution de générateurs de Tausworthe (et d'autres générateurs décrits dans le livre de Tezuka [46]), il ne permet pas de déterminer l'équidistribution pour tous les générateurs qui suivent le cadre défini par les équations (1)-(4). C'est pourquoi nous utilisons le réseau qui est défini dans la sous-section suivante.

3.2.2 Réseau en résolution

Dans la section 2.1, pour un initialisation donnée $\mathbf{x}_0$, on définit les séquences $\gamma_i = (z_0^{(i)}, z_1^{(i)}, \ldots), 1 \leq i \leq k$. Pour chacune des séquences γ_i (c'est-à-dire, pour chaque bit de la séquence $\{\mathbf{z}_n, n \geq 0\}$), on associe une fonction génératrice de la forme $g_i(z)/P(z)$. On peut donc associer à l'état $\mathbf{x}_0$, un vecteur de fonctions génératrices

$$(g_0(z), \ldots, g_{k-1}(z))/P(z).$$

En fait, pour chaque état $\mathbf{x}_n$, on peut associer un vecteur de fonctions génératrices qui caractérise la séquence des vecteurs $\mathbf{x}_m, m \geq n$. Soit

$$Q_i = ((g_{i,0}(z), \ldots, g_{i,k-1}(z))/P(z)$$

le vecteur de fonctions génératrices qui est associé à $\mathbf{x}_i$.

Soit $1 \leq \ell \leq k$, un entier. Selon [45], si $g_{0,0}(z) \neq 0$, tous les points $Q_i \in \mathbb{F}_2[[z]]^t, i \geq 0$ sont éléments du réseau polynômial (voir définition B.5) dont une base possible est

$$\begin{array}{rcl} B_1 & = & (g_{0,0}(z), \ldots, g_{0,\ell-1}(z))/P(z), \\ B_2 & = & (0, 1, \ldots, 0), \\ & \ldots & \\ B_\ell & = & (0, 0, \ldots, 1). \end{array} \tag{38}$$

Ce réseau est appelé *réseau en résolution* ℓ du générateur [45]. Grâce à ce type de réseau et à la prochaine proposition, il est possible de calculer l'équidistribution d'un générateur pour le cas où l'on ne considère que les vecteurs de valeurs successives. Pour calculer l'équidistribution des vecteurs de valeurs non successives, si le générateur n'est pas un générateur de Tausworthe (ou le réseau en dimension décrit par le générateur n'est pas un réseau polynômial [18]), il faut se rabattre sur la méthode matricielle puisqu'aucune méthode de détermination de l'équidistribution des ensembles de points produits par des vecteurs de valeurs non successives utilisant le réseau en dimension n'est connue.

Proposition 3.1 *(Tezuka [45])*
Un générateur est $(-v_\ell, \ell)$-équidistribué, où $v_\ell = \log_2 \sigma_\ell$ et σ_ℓ est le ℓ-ième minimum successif du réseau associé en résolution ℓ.

Il est aussi à noter que si le polynôme $P(z)$ est primitif, alors il est possible de définir le vecteur B_1 par

$$B_1 = (g_{0,0}(z)(1, z^{j_1}, \ldots, z^{j_{\ell-1}}) \bmod P(z))/P(z) \tag{39}$$

où les $j_i, 1 \leq i < \ell$ sont des entiers. Cette observation sera utile ultérieurement dans ce document.

3.2.3 Réseau dual

Dans cette sous-section, nous définissons le réseau dual du réseau en résolution. Ce réseau peut s'avérer utile dans certaines situations puisque la réduction du réseau dual est, la plupart du temps, plus rapide que la réduction du réseau en résolution [4].

Tout d'abord, on définit un produit scalaire sur $\mathbb{F}_2[[z]]^\ell$, pour $v = (v_1, \ldots, v_\ell)$ et $w = (w_1, \ldots, w_\ell) \in \mathbb{F}_2[[z]]^\ell$, par

$$\langle v, w \rangle = \sum_{i=1}^{\ell} v_i w_i.$$

Soit $\mathcal{L}_\ell$, un réseau en résolution ℓ d'un générateur dont la base est $\{\Pi_1, \ldots, \Pi_\ell\}$. On définit [18] le réseau dual $\mathcal{L}'_\ell$ par un réseau dont une base $\{\Phi_1, \ldots, \Phi_\ell\}$ satisfait

$$\langle \Pi_i, \Phi_j \rangle = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{sinon} \end{cases} \tag{40}$$

pour $1 \leq i, j \leq \ell$. Par exemple, si une base du réseau $\mathcal{L}_\ell$ est $\{B_1, \ldots, B_\ell\}$ où

$$\begin{array}{rcl} B_1 & = & (1, g_2(z), \ldots, g_\ell(z))/P(z), \\ B_2 & = & (0, 1, \ldots, 0), \\ & \ldots & \\ B_\ell & = & (0, 0, \ldots, 1), \end{array} \tag{41}$$

alors une base du réseau dual $\mathcal{L}'_\ell$ est définie par

$$\begin{array}{rcl} B'_1 & = & (P(z), 0, \ldots, 0), \\ B'_2 & = & (g_2(z), 1, \ldots, 0), \\ & \ldots & \\ B'_\ell & = & (g_\ell(z), 0, \ldots, 1). \end{array}$$

On remarque que pour ce réseau dual, les vecteurs définissant la base sont strictement dans $\mathbb{F}_2[z]^\ell$.

Dans le cas du réseau dual, on utilise une norme différente de celle définie précédemment. Soit $B = (b_1(z), \ldots, b_\ell(z)) \in \mathcal{L}'_\ell$, on définit

$$\|B\|' = \max_{i=1,\ldots,\ell} 2^{\deg(b_i(z))},$$

comme étant cette autre norme sur $\mathbb{F}_2[[z]]^\ell$.

Soit σ'_i, le i-ième minimum successif du réseau $\mathcal{L}'_\ell$ (mesuré avec la norme $\|\cdot\|'$). Le théorème suivant permet de déterminer l'équidistribution d'un générateur.

Théorème 3.3 *(Couture [4])*
Soit $\mathcal{L}_\ell$, le réseau en résolution ℓ induit par un générateur et $\mathcal{L}'_\ell$, le réseau dual de $\mathcal{L}_\ell$. Le générateur est (t, ℓ)-équidistribué si et seulement si $t \leq log_2 \sigma'_i$ pour $i = 1, \ldots, \ell$.

3.3 "Tempering" permettant d'améliorer l'équidistribution

Dans la section 2.2, on a discuté de différents types de générateurs de nombres aléatoires qui existent dans la littérature (c'est-à-dire de différents types de matrices X). Lors de l'implantation de ces générateurs, la matrice B est très souvent la matrice identité. Matsumoto et Kurita [26] utilisent pour la première fois une matrice B différente de l'identité sur leurs TGFSR. Les matrice A des TGFSR qu'ils utilisent sont de la forme

$$A = \begin{pmatrix} & & & a_0 \\ 1 & & & a_1 \\ & 1 & & a_2 \\ & & \ddots & \vdots \\ & & 1 & a_{w-1} \end{pmatrix}. \tag{42}$$

Le problème avec cette matrice A (avec $B = I_w$) est que l'équidistribution pour les deux premiers bits est $t_2 \leq r$, bien loin de la borne supérieur $t_2^* = \lfloor rw/2 \rfloor$. Afin d'améliorer l'équidistribution, ils introduisent une transformation linéaire, le tempering de Matsumoto-Kurita. Malgré cette transformation linéaire, l'équidistribution de ce type de générateur est borné par

$$t_\ell \leq r\lfloor w/\ell \rfloor.$$

Un autre exemple d'utilisation d'un autre type de tempering peut être trouvé dans [28]. Ils appliquent une transformation linéaire similaire au tempering de Matsumoto-Kurita. Mais encore une fois, les auteurs n'arrivent pas à rencontrer la borne supérieur pour plusieurs valeurs de ℓ.

Dans [36], on applique diverses transformations linéaires sur un générateur de type GCL polynômial et on obtient un générateur qui est équidistribué au maximum (ME). Mais ce générateur est deux fois plus lent que s'il n'y avait pas de tempering.

Dans le cadre de ma recherche, je vais tenter de trouver de nouvelles récurrences (c'est-à-dire de nouvelles matrice X) qui donnent de bons résultats en utilisant le moins possible de tempering (si possible, $B = I$). Le problème majeur du tempering est qu'il ralentit l'implantation. Il se pourrait que cela soit un mal nécessaire, mais j'éviterai, autant que possible, l'utilisation du tempering.

3.4 Réseaux digitaux

L'équidistribution est un critère qui permet de mesurer l'uniformité d'un ensemble de points Ω_t tel que défini par l'équation (32). Il existe un autre critère qui ne fait pas que regarder le nombre de points exclusivement dans des boîtes cubiques comme c'est le cas pour l'équidistribution. Celui-ci est la q-valeur d'un (q, k, t)-réseau. Voici une définition de ce concept.

Définition 3.1 *(Niederreiter [29], Schmid [40])*
Soit $b \geq 2, t \geq 1$ et $0 \leq q \leq k$ des entiers. Un ensemble de points composé de b^k points de $[0,1)^t$ forme un (q,k,t)-réseau en base b si chaque sous-intervalle $J = \prod_{i=1}^{t} = [a_i b^{-d_i}, (a_i + 1)b^{-d_i})$ de $[0,1)^t$ avec les entiers $d_i \geq 0$ et $0 \leq a_i \leq b^{d_i}$ pour $1 \leq i \leq t$ et de volume b^{q-k} contient exactement b^q points de l'ensemble de points.

Il est important de faire la distinction entre un (q,k,t)-réseau et un réseau polynômial. La confusion entre les noms n'est pas présente en anglais où un (q,k,t)-réseau est appelé "(q,k,t)-net" et un réseaux polynômial est appelé "polynomial lattice". On définit la q-valeur d'un ensemble de points comme étant la plus petites valeur de q telle que l'ensemble de points est un (q,k,t)-réseau. Cette q-valeur constitue une mesure de l'uniformité d'un ensemble de points. Plus elle est petite, plus l'ensemble de points est uniforme dans $[0,1)^t$. Il s'agit en fait d'un critère plus exigeant que l'équidistribution puisque ce critère observe le nombre de points dans plusieurs petits rectangles et non seulement dans des petits cubes.

Définition 3.2 *(Niederreiter [29], Schmid [40])*
Soit p une puissance d'un nombre premier et $t \geq 1, k \geq 1$ des entiers. Soit $C^{(1)}, \ldots, C^{(t)}$ des matrices de dimension $k \times k$ sur $\mathbb{F}_p$. Soit $n = \sum_{j=0}^{k-1} a_j p^j$, la représentation p-adique de n en base p pour $0 \leq n \leq p^k$. Prenons $a_0, \ldots, a_{k-1}$ comme des éléments de $\mathbb{F}_p$. Soit

$$(y_1^{(i)}(n), \ldots, y_k^{(i)}(n))^T = C^{(i)}(a_0, \ldots, a_{k-1})^T$$

pour $i = 1, \ldots, t$ et

$$V_n = (x_n^{(1)}, \ldots, x_n^{(t)}) \in [0,1)^t \text{ avec } x_n^{(i)} = \sum_{j=1}^{k} y_j^{(i)}(n) p^{-j}.$$

Si pour un entier q avec $0 \leq q \leq k$ l'ensemble de points

$$V_n = (x_n^{(1)}, \ldots, x_n^{(t)}) \in [0,1)^t \quad \text{pour } n = 0, \ldots, p^k - 1$$

est un (q,k,t)-réseau en base p, alors il est appelé (q,k,t)-réseau digital construit sur $\mathbb{F}_p$.

Un générateur qui utilise une récurrence linéaire modulo 2 décrit un (q,k,t)-réseau digital avec $p = 2$. Pour un ensemble Ω_t produit par un générateur donné, le réseau digital résultant est celui où $C^{(i)} = BX^i$ pour $i = 1, \ldots, t$ et k est la dimension du vecteur d'état.

Dans [39], les auteurs donnent deux techniques qui permettent de trouver la q-valeur d'un réseau digital donné. Une est basée sur un code Gray et l'autre sur l'élimination gaussienne. Ils montrent que la méthode par le code Gray est plus rapide dans le cas où $p = 2$. Nous utiliserons cette méthode pour déterminer la q-valeur des générateurs à évaluer. Puisque le nombre de boîtes à observer croît de manière exponentielle en k, il n'est possible de déterminer la q-valeur que pour de petits générateurs. Les générateurs avec de petites valeurs de k peuvent être utilisés pour l'intégration quasi-Monte Carlo.

4 Description du projet de recherche

Nous voici donc dans la section où je spécifie les endroits où je crois pouvoir faire des contributions importantes dans le domaine de la génération de nombres pseudo-aléatoires et d'ensembles de points uniformes.

Le but du projet de recherche est de construire des générateurs de nombres pseudo-aléatoires qui produisent des ensembles de points bien équidistribués pour la simulation Monte Carlo et pour l'intégration quasi-Monte Carlo.

Le problème majeur des générateurs qui sont proposés dans la littérature (et qui sont sommairement expliqués dans la section 2.2) est que la récurrence est souvent trop simple. Ceci fait en sorte qu'il n'y a pas assez de "brassage" de bits d'une itération à l'autre, causant une mauvaise équidistribution. Le tempering permet de résoudre ce problème partiellement, mais celui-ci ne peut pas enrayer les limitations induites par la matrice X choisie (voir l'exemple du TGFSR en section 3.3).

Il faut donc trouver de nouvelles récurrences (matrices X) qui sont rapides à exécuter sur un ordinateur et qui nécessitent le minimum de tempering. De plus, ces nouvelles récurrences doivent être facilement séparables afin de pouvoir produire plusieurs séquences de nombres pseudo-aléatoires indépendantes en séparant la période en segments. Chacun des segments produit une séquence de nombres $u_{r_j}, u_{r_j+1}, \ldots$ où la valeur r_j est propre à la j-ième séquence.

Voici une liste détaillée de mes principaux objectifs de recherche.

Objectif 1 : Étude de générateurs MRMM basés sur des polynômes dans $\mathbb{F}_{2^w}$

Je propose d'utiliser une récurrence dans $\mathbb{F}_{2^w}$ de la forme

$$m_n = \sum_{i=1}^{r} b_i m_{n-i} \tag{43}$$

où $m_n \in \mathbb{F}_{2^w}, n \geq 0$, r est un entier positif, $b_i \in \mathbb{F}_{2^w}$ et l'arithmétique se fait dans $\mathbb{F}_{2^w}$ (Voir l'annexe A pour plus de détails sur l'arithmétique dans $\mathbb{F}_{2^w}$). Cette récurrence est la récurrence (30) de [30]. Le polynôme caractéristique dans $\mathbb{F}_{2^w}$ de cette récurrence est le polynôme

$$Q(z) = z^r - \sum_{i=1}^{r} b_i z^{r-1}.$$

Afin que ce type de générateur ait pleine période, il faut que le polynôme $Q(z)$ soit primitif dans $\mathbb{F}_{2^w}$ [30]. Comme dans le cas de la récurrence de base (5), on peut associer une fonction génératrice

$$H(z) = h(z)/Q(z) = m_0 z^{-1} + m_1 z^{-2} + \cdots$$

à une récurrence (43) où $h(z) \in \mathbb{F}_{2^w}[z]/Q(z)$, étant donné une initialisation $m_0, m_1, \ldots, m_{r-1}$. Il est facile de sauter de j itérations en avant dans la récurrence (43) en effectuant $g(z) = z^j h(z) \bmod Q(z)$. La fonction génératrice $G(z) = g(z)/Q(z)$ nous donne explicitement l'état du générateur.

Afin d'implanter la récurrence (43), on peut choisir la base polynômiale $(1, \zeta, \ldots, \zeta^{w-1})$ où ζ est un élément primitif de $\mathbb{F}_{2^w}$ pour représenter les éléments de $\mathbb{F}_{2^w}$ (voir l'annexe A pour une courte introduction sur l'arithmétique dans $\mathbb{F}_{2^w}$. Dans ce cas, on peut l'implanter par

$$\mathbf{v}_n = \sum_{i=0}^{r} I(b_i \neq 0) A^{c_i} \mathbf{v}_{n-i} \tag{44}$$

où les c_i sont déterminés par $b_i = \zeta^{c_i}$ (puisque ζ est primitif, n'importe quel élément de $\mathbb{F}_{2^w}$ peut être exprimé par une puissance de ζ), I est la fonction indicatrice et la matrice A de dimension $w \times w$ est la matrice compagnon de ζ. Cette récurrence utilise la "Multiple Recursive Matrix Method" telle que décrite à la section 2.2.6. L'état de ce type de générateur est $\mathbf{x}_n = (\mathbf{v}_n^T, \mathbf{v}_{n-1}^T, \ldots, \mathbf{v}_{n-r+1}^T)^T$. Puisqu'il s'agit d'un générateur MRMM, on peut se servir du théorème 2.3 pour déterminer si une récurrence (43) est de pleine période.

On peut voir que le TGFSR avec matrice A comme dans (42) est un cas spécial de cette récurrence. Le TGFSR utilise la base polynômiale, $q = m - r$, $c_q = 0$, $c_r = 1$ et $b_i = 0$ sauf pour $i = r$ et $i = q$. Le GCL polynômial utilise une variante de cette récurrence. La récurrence est $m_n = \zeta^s m_{n-1}$.

Il est à remarquer que lorsqu'on applique un tempering sur ce type de générateur de la manière suivante : $\mathbf{y}_n = ((R\mathbf{v}_n)^T, (R\mathbf{v}_{n-1})^T, \ldots, (R\mathbf{v}_{n-r+1})^T)^T$ où R est une matrice $w \times w$ de plein rang (comme c'est fait pour le TGFSR dans [26]), tout ce que l'on fait, c'est changer la base de représentation des éléments de $\mathbb{F}_{2^w}$. Ceci suggère que le choix de la base est important dans la performance de ce type de générateur par rapport à l'équidistribution.

Proposition de recherche
Je propose d'explorer plus à fond le choix de la base et de son implication pour l'équidistribution. Dans [26], en appliquant le tempering, le changement de base est fait de façon arbitraire et par essais-erreurs. Je propose de voir s'il est possible de faire une recherche mieux guidée du choix de la base de $\mathbb{F}_{2^w}$. Également, je veux trouver de bons générateurs de nombres pseudo-aléatoires et de bons petits générateurs pour l'intégration quasi-Monte Carlo. Ceux-ci seront choisis par rapport à l'équidistribution, la q-valeur, ou possiblement d'autres critères. ∎

Objectif 2 : Étude de générateurs à congruence linéaire dans $\mathbb{F}_{2^w}[z]/P(z)$

Je propose d'étudier les récurrences du type

$$q_n(z) = a(z)q_{n-1}(z) \bmod Q(z) \tag{45}$$

où $Q(z) = z^r - b_1 z^{k-1} - \cdots - b_r$ est un polynôme dans $\mathbb{F}_{2^w}[z]$ et $q_n(z), a(z) \in F_{2^w}[z]/Q(z)$. L'état de ce type de générateur est le polynôme $q_n(z)$. Ce polynôme est, à l'itération n,

$$q_n(z) = \alpha_{n,w-1} z^{w-1} + \alpha_{n,w-2} z^{w-2} + \cdots + \alpha_{n,0}$$

où $\alpha_i \in \mathbb{F}_{2^w}, i = 0, \ldots, w-1$. Sur un ordinateur, on peut aussi représenter $q_n(z)$ par un vecteur de bits

$$\mathbf{x}_n = (\mathbf{x}_n^{w-1}, \mathbf{x}_n^{w-2}, \ldots, \mathbf{x}_n^0)$$

où $\mathbf{x}_n^i, i = 0, \ldots, w-1$ représente $\alpha_{n,i}$ selon une base choisie de $\mathbb{F}_{2^w}$.

Supposons que l'on utilise la base ordonnée polynômiale $(1, \zeta, \ldots, \zeta^{w-1})$ pour $\mathbb{F}_{2^w}$ où ζ est primitif. Pour les valeurs de $i = 1, \ldots, r$ tels que $b_i \neq 0$, on défini les c_i par $b_i = \zeta^{c_i}$ et $A_i = A^{c_i}$, pour les autres valeurs de i, on met $A_i = 0$. La récurrence (45) peut être exprimée sous la forme matricielle par

$$\mathbf{x}_n = X\mathbf{x}_{n-1}$$

où

$$X = \begin{pmatrix} A_1 & I_w & & \\ A_2 & & I_w & \\ \vdots & & & \ddots \\ A_{r-1} & & & I_w \\ A_r & & & \end{pmatrix}$$

est une matrice $rw \times rw$. On peut voir cette récurrence aussi sous la forme

$$\mathbf{x}_n = (\mathbf{x}_{n-1}^{w-2}, \mathbf{x}_{n-1}^{w-3}, \ldots, \mathbf{x}_{n-1}^0, 0) + (A_1, \ldots, A_r)\mathbf{x}_{n-1}^{w-1}.$$

Dans le cas où $Q(z)$ est un trinôme dans $\mathbb{F}_{2^w}[z]$, il n'y a que deux valeurs de $1 \leq i \leq r$ telles que $A_i \neq 0$.

En comparant la forme de cette matrice X et celle d'un générateur MRMM (24), on voit qu'elles sont semblables : l'une est la transposée de l'autre. La même remarque avait été faite dans une comparaison entre les matrice $\bar{X}$ des générateurs de Tausworthe et des GCL polynômiaux dans la section 2.2.3. On pourrait dire que le générateur MRMM basé sur un polynôme dans $\mathbb{F}_{2^w}$ est un générateur de Tausworthe dans $\mathbb{F}_{2^w}$ avec $s = 1$ et le GCL polynômial dans $\mathbb{F}_{2^w}$ est une extension du simple GCL polynômial défini dans la section 2.2.3.

Afin de sauter j itérations en avant dans la récurrence (45), il suffit de multiplier $q_n(z)$ par $a(z)^j$ dans $\mathbb{F}_{2^w}[z]/P(z)$ pour obtenir $q_{n+j}(z)$.

Le cadre défini par le GCL polynômial nous restreint à choisir des formes particulières des matrices A_i. Souvent, ces matrices sont complexes et la multiplication par ces matrices sont complexes. Pour obtenir d'autres générateurs (qui ne sont pas nécessairement des GCL polynômiaux) qui soient plus rapides, rien ne nous empêche de choisir des matrices A_i quelconques. Dans ce cas, pour avoir une pleine période, il faut que la matrice A_r soit de plein rang et que le polynôme défini par l'équation (25) soit primitif dans $\mathbb{F}_2$.

Proposition de recherche
Comme pour le MRMM basé sur un polynôme dans $F_{2^w}[z]$, je veux étudier le choix de la base dans $\mathbb{F}_{2^w}$ et de son implication pour l'équidistribution. Également, je veux trouver de bons générateurs de nombres pseudo-aléatoires et de bons petits générateurs pour l'intégration quasi-Monte Carlo selon différents critères. Je veux explorer les similitudes entre les générateurs MRMM basés sur des récurrences du type (43) et les GCL dans $\mathbb{F}_{2^w}[z]/P(z)$. Je veux comparer les vitesses d'exécution et l'équidistribution. Je veux également construire de bons générateurs avec des matrices A_i quelconques. ∎

Objectif 3 : Variations

En observant la manière dont le MRMM et le GCL polynômial dans $\mathbb{F}_{2^w}$ modifient leur vecteur d'état $\mathbf{x}_n$, on remarque une chose : la récurrence du MRMM modifie un seul bloc de w bits de $\mathbf{x}_n$ en combinant linéairement plusieurs blocs de w bits. Le GCL polynômial dans $\mathbb{F}_{2^w}$, de son côté, modifie plusieurs blocs en tenant compte de la valeurs d'un seul bloc de w bits. Cette observation, faite par Makoto Matsumoto lors d'une conversation privée, suggère la généralisation suivante : une récurrence qui modifie plusieurs blocs de w bits en tenant compte de la valeurs de plusieurs blocs de w bits. Ceci se résume par la matrice de transition

$$X = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1r} \\ A_{21} & A_{22} & \dots & A_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ A_{r1} & A_{r2} & \dots & A_{rr} \end{pmatrix} \tag{46}$$

où les matrices $A_{ij}, 1 \leq i, j \leq r$, sont $w \times w$ et ont leurs coefficients dans $\mathbb{F}_2$. L'état de ce générateur est le vecteur

$$\mathbf{x}_n = (\mathbf{x}_n^0, \mathbf{x}_n^1, \dots, \mathbf{x}_n^{w-1}).$$

La période de ce type de générateur est de $2^{rw} - 1$ si et seulement si le polynôme caractéristique de la matrice X est primitif et de degré rw.

Afin d'avoir des générateurs rapides, on peut choisir peu de matrices A_{ij} non nulles. Pour celles-ci, on peut choisir des matrices A_{ij} avec peu de coefficients non nuls.

Proposition de recherche
Je veux trouver des générateurs rapides qui utilisent des récurrences de type (46). Je trouverai des générateurs pour l'intégration quasi-Monte Carlo et pour la simulation numérique. Ils seront bons par rapport à l'équidistribution, la q-valeur, ou d'autres critères. ∎

Objectif 4 : Étude de l'équidistribution des automates cellulaires

Dans la littérature, on ne retrouve rien sur l'équidistribution des automates cellulaires. Il y a une propriété qui s'apparente à l'équidistribution qui est utile . On dit qu'un automate cellulaire à k cellules est *exhaustif* s'il peut générer les 2^k vecteurs possibles à partir de l'état initial. On dit qu'il est *pseudo-exhausif* pour $k' < k$ cellules si tous les ensembles de k' cellules adjacentes sont exhaustifs. Ces propriétés sont importantes pour la détection de défauts dans un circuit électronique. Soit un circuit avec k entrées binaires. On vérifie ce circuit en donnant à chacune des entrées un bit provenant d'un automate cellulaire. En ayant un automate cellulaire exhaustif, on peut vérifier le circuit pour toutes les combinaisons possibles d'entrées. Mais de nos jours, les circuits sont très gros et il est impossible de les vérifier de façon exhaustive. En ciblant certains groupes d'entrées de tailles k' qui sont dépendantes entre elles dans le fonctionnement du circuit, on peut utiliser des automates cellulaires qui sont pseudo-exhaustifs pour ces groupes de k' bits.

Proposition de recherche
Pour les automates cellulaires à une dimension dans $\mathbb{F}_2$, des vérifications sommaires sur quelques automates me permettent d'énoncer la conjecture suivante : il n'existe pas d'automates cellulaires à une dimension, avec un voisinage de $v = 1$ qui soit équidistribué au maximum. Je désire prouver cette conjecture.

Pour les automates cellulaires en deux dimensions dans $\mathbb{F}_2$, je désire étudier l'équidistribution de ces générateurs ainsi que leur q-valeur.

Dans le cas des automates cellulaires en une dimension dans $\mathbb{F}_{2^p}(p > 1)$, je désire dans un premier temps en étudier l'équidistribution. Dans un deuxième temps, je voudrais comparer ceux-ci aux automates cellulaires en deux dimensions où $m = p$. Dans ces 2 types d'automates cellulaires, on doit utiliser np bits pour emmagasiner l'état, mais celui dans $\mathbb{F}_{2^p}$ utilise une circuiterie plus complexe. La question est de savoir si la complexité de l'automate cellulaire justifie le gain obtenu en terme d'équidistribution. ∎

Objectif 5 : Recherche de règles de réseaux digitaux extensibles

Dans cet objectif, on décrit une méthode pour construire des ensembles de n points pour l'intégration Monte Carlo, mais où n n'est pas déterminé à l'avance.

Une règle de réseau polynômial de rang 1 en dimension t est un ensemble de $n = 2^k$ points dans $\mathbb{F}_2[[z]]^t$ défini par

$$P_n = \{\text{frac}(a(z)H(z)/P(z)) : \deg(a(z)) < k\},$$

où $\deg(P(z)) = k$, $H(z) = (g_0(z), \ldots, g_{t-1}(z))$ et $\deg(g_i(z)) < k$ pour $i = 0, \ldots, t-1$. Pour obtenir un ensemble de points $\tilde{P}_n$ dans $(0, 1]^t$, il suffit de remplacer les z par 2 dans

l'expansion formelle de Laurent de chaque coordonnée de chaque point de P_n. On choisit les polynômes $g_i(z)$ et le polynôme $P(z)$ afin que l'ensemble de points $\tilde{P}_n$ soit uniforme dans l'hypercube $(0,1]^t$. L'équidistribution est une mesure possible de cette uniformité. L'ensemble de points $\tilde{P}_n$ est utilisé pour effectuer une intégration quasi-Monte Carlo. Lorsqu'on utilise une règle de réseau polynômial pour évaluer une intégrale, il existe une théorie qui permet de borner l'erreur d'intégration pour certaines classes de fonctions [18]. Cette théorie rend l'intégration par des règles de réseau polynômial intéressante.

On peut implanter de bonnes règles de réseau polynômial grâce à des générateurs de Tausworthe [16, 18]. Le problème avec ce type d'ensembles de points est que le nombre de points n est fixe et on doit le déterminer au début de l'intégration quasi-Monte Carlo. Si, après évaluation, on se rend compte que l'intégration n'est pas assez précise, alors il faut augmenter le nombre de points. On peut choisir un autre ensemble avec le même nombre de points n et faire la moyenne des deux résultats, ou recommencer avec une autre règle de réseau ayant $2n$ points et rejeter le résultat obtenu avec la première intégration. Dans le premier cas, l'union des deux ensembles de points ne produit pas nécessairement une règle de réseau. Dans l'autre cas, on utilise un ensemble de points qui est une règle de réseau, mais on gaspille l'effort de calcul pour la première évaluation.

Une solution à ce problème est la construction de règles de réseau polynômial extensibles [33]. Cette idée a été introduite dans [8] où l'on construit des ensembles de points qui s'emboîtent l'un dans l'autre, un peu à la manière des poupées russes. Une règle de réseau polynômial extensible est en fait une suite de règles de réseau polynômial qui se contiennent l'une dans l'autre. Afin d'utiliser ce type de règle, on produit une première règle de réseau. Si on veut augmenter le nombre de points, alors on produit la prochaine règle de réseau dans la suite de règles (qui contient plus de points que la première règle). Puisque les points déjà produits par la première règle font partie de cette autre règle, alors on sauve le temps de calcul nécessaire à calculer les points déjà produits.

Voici quelques définitions permettant de définir une règle de réseau polynômiale extensible.

Définition 4.1 *(Niederreiter [33])*
Une séquence $F = (f_k)_{k=1}^{\infty}$ de polynômes dans $\mathbb{F}_2[z]$ est appelée chaîne de divisibilité dans $\mathbb{F}_2[z]$ si f_k divise f_{k+1} et $1 \leq \deg(f_k) < \deg(f_{k+1})$ pour tout $k \geq 1$.

Également, on associe à F, l'ensemble Y_F des *polynômes F-adiques* qui est l'ensemble des sommes formelles

$$A = \sum_{j=0}^{\infty} a_j f_j$$

avec $f_0 = 1$, $a_j \in \mathbb{F}_2[z]$ et $\deg(a_j) < \deg(f_{j+1}) - \deg(f_j)$ pour $j \geq 0$. Définissons maintenant la fonction $\tau : \mathbb{F}_2[[z]] \to [0,1]$ par

$$\tau\left(\sum_{j=w}^{\infty} b_j z^{-j}\right) = \sum_{j=\max(1,w)}^{\infty} b_j 2^{-j}.$$

Soit $F = (f_k)_{k=1}^{\infty}$, une chaîne de divisibilité quelconque, alors n'importe quel polynôme g dans $\mathbb{F}_2[z]$ peut être écrit dans la forme

$$g(z) = \sum_{j=0}^{k-1} g_j(z) f_j(z)$$

avec $f_0(z) = 1$, $g_j(z) \in \mathbb{F}_2[z]$ et $\deg(g_j(z)) < \deg(f_{j+1}(z)) - \deg(f_j(z))$ pour $0 \leq j \leq k-1$. Cette représentation est unique, sauf pour l'addition de coefficients nuls. C'est pourquoi la fonction

$$\phi_F(g(z)) = \sum_{j=0}^{k-1} \frac{g_j(z)}{f_{j+1}(z)} \in \mathbb{F}_2[[z]]$$

est bien définie.

Finalement, pour un vecteur $V = (v_1, \ldots, v_t) \in Y_F^t$, on obtient une séquence infinie $\sigma(V, F)$ composée par les points

$$(\tau(\phi_F(g(z))b_1), \ldots, \tau(\phi_F(g(z))b_t)) \in [0,1]^t,$$

où $g(z)$ prend comme valeur tous les polynômes sur $\mathbb{F}_2[z]$ en ordre de degré non décroissant. Cette séquence $\sigma(V, F)$ est ce qu'on appelle *une règle de réseau polynômial extensible*.

Comme il est expliqué dans [33], si on considère les $N_k = 2^{\deg(f_k)}$ premiers points de la séquence $\sigma(V, F)$, ceux-ci constituent un réseau polynômial avec $P(z) = f_k(z)$ et $H(z) = V \bmod f_k$ où l'opération mod est exécutée composante par composante.

Un exemple de construction de règles de réseau polynômial extensibles est par la combinaison de générateurs de Tausworthe. Soit $G(f(z), s)$, un générateur de Tausworthe de paramètre de saut s et dont $f(z)$ est le polynôme caractéristique de la matrice $\bar{X}_{\text{taus}}$. Soit $k = \deg(f(z))$. En prenant la sortie définie par (10) et (4), l'ensemble de points $\{(u_0, u_1, \ldots, u_{t-1}) : \mathbf{x}_0 \in \mathbb{F}_2^k\}$ produit par un générateur de Tausworthe constitue une règle de réseau polynômial. Afin que la combinaison de plusieurs générateurs de Tausworthe constitue toujours une règle de réseau, on prend comme sortie du générateur combiné $\mathbf{y}_n = \sum_{j=1}^{J} \mathbf{y}_{j,n}$ et (4) où $\mathbf{y}_{j,n}$ est le vecteur de sortie de la j-ième composante et J est le nombre de composantes. Prenons, par exemple, un ensemble de générateurs de Tausworthe $\{G(f_i(z), s) : i = 1, 2, \ldots\}$ dont les polynômes $f_i(z)$ sont premiers entre eux. La première règle de réseau polynômial générée est produite par le générateur $G(f_1, s)$ sur tous ses cycles. La deuxième est construite avec la combinaison de $G(f_1(z), s)$ et $G(f_2(z), s)$ sur tous ses cycles. Il est à remarquer que, pour la deuxième règle, les cycles dont l'état du deuxième générateur est le vecteur nul n'ont pas besoin d'être générés puisqu'ils sont contenus dans la première règle. On construit les autres règles de la même manière. Avec cette construction, on obtient $V = (1, z^s, z^{2s}, \ldots, z^{(t-1)s})$.

Une généralisation des règles de réseau polynômial extensibles serait les règles de réseau digital extensibles. Une construction possible est par les séquences digitales. Ce type de construction est expliquée dans [15] et ses références.

D'autres constructions de (q, k, t)-réseaux digitaux extensibles basés sur des récurrences (c'est à dire, quand $C^{(i)} = BX^i$ pour une certaine matrice de transition X et une matrice de tempering B) peuvent être faites, mais, la théorie pour celles-ci est inexistante. Une construction possible serait à l'aide d'un générateur combiné à J composantes qui ont les matrices de transitions $X_1, \ldots, X_J$ où X_j est de dimension $k_j \times k_j$. Soit $k = k_1 + \cdots + k_J$ et $\mathbf{x}_n$ le vecteur d'état du générateur combiné de k bits. La récurrence de ce générateur combiné est

$$\mathbf{x}_n = \begin{pmatrix} X_1 & & & \\ & X_2 & & \\ & & \ddots & \\ & & & X_J \end{pmatrix} \mathbf{x}_{n-1} = X\mathbf{x}_{n-1}.$$

On produit la sortie avec

$$\mathbf{y}_n = B\mathbf{x}_n$$

et

$$u_n = \sum_{i=1}^{k} y_n^{(i-1)} 2^{-i}$$

où B est de dimension $k \times k$. Soit $k'_j = k_1 + \ldots + k_j$. La matrice B, pour cette construction, a la particularité que les sous-matrices composés des k'_j premières lignes et des k'_j premières colonnes sont de rang k_j. Ainsi, on définit l'ensemble de points R_j, l'ensemble des points $(u_0, u_1, \ldots, u_t - 1)$, à partir de tous les états $\mathbf{x}_0$ tels que seuls les k'_j premiers bits peuvent être non nuls et les autres bits sont nuls. On peut montrer (et ce sera fait dans ma thèse) que $R_j \subseteq R_{j+1}$ pour $j = 1, \ldots, J-1$. La condition qu'on impose sur certaines sous-matrices de B est nécessaire pour une bonne distribution des points de chaque ensemble R_j. Si celle-ci n'était pas respectée, il serait possible, par exemple, que tous les points aient comme première coordonnée une valeur plus petite que $1/2$. Dans ce cas, l'équidistribution des projections contenant cette coordonnée serait mauvaise. Les ensemble de points R_j constituent des (q, k, t)-réseaux digitaux où $C^{(i)} = BX^{i-1}$.

On peut montrer que les règles de réseau polynômial extensibles implantés par des générateurs de Tausworthe combinés entre dans le cadre de cette construction. Les détails de ceci se trouveront dans ma thèse.

Proposition de recherche

Malgré que l'existence de bonnes[3] règles de réseau polynômial a été démontré dans [33], aucun paramètre de bonnes règles n'a encore été publié. C'est pourquoi je désire trouver de bonnes séquences $\sigma(V, F)$ par rapport à l'équidistribution, à la q-valeur, ou d'autres critères. Ces séquences pourront être produites à l'aide soit de générateurs de Tausworthe combinés, soit de méthodes existantes ou soit d'autres méthodes que je développerai. De façon plus générale, je veux trouver des réseaux digitaux, basés sur des récurrences ou non, qui produisent des ensembles de points emboîtés l'un dans l'autre. Cette idée de réseaux digitaux extensibles basés sur des récurrences est nouvelle. Il s'agit d'une généralisation des règles de réseau polynômial extensibles avec $B = (b(z), b(z)^2, \ldots, b(z)^{t-1})$. Toute la théorie sur

[3]Selon le critère $R^{(s)}$, qui est décrit dans [33]

cette façon de produire des règles de réseaux digitaux extensibles est à faire et je propose de l'explorer.■

Objectif 6 : Trouver de bons (q, k, t)-réseaux digitaux

Puisque le fait d'utiliser de petits générateurs pour la génération d'ensemble de points uniformes est assez récente, peu de recherche de bons générateurs par rapport à ce critère n'a été fait jusqu'à présent.

Proposition de recherche
J'implanterai la méthode de détermination de la q-valeur qui utilise le code Gray dans le progiciel REGPOLY [4]. Aussi, je tenterai de trouver de bons petits générateurs par rapport à la q-valeur du réseau digital qu'ils décrivent. Je trouverai également des réseaux digitaux extensibles basés sur des récurrences qui possèdent une bonne q-valeur.■

Objectif 7 : Accélération du calcul de la base du réseau dual

Lorsqu'on utilise un réseau en résolution pour calculer l'équidistribution, le vecteur $Q_0 = (g_{0,0}(z), \ldots, g_{0,\ell-1}(z))/P(z)$ est directement disponible par la récurrence du générateur. Le réseau décrit est celui donné par l'équation (38) où $B_1 = Q_0$.

Pour déterminer l'équidistribution, en utilisant le dual du réseau en résolution, il faut calculer d'abord le dual du réseau. Cette tâche peut être longue dans le cas où le vecteur Q_0 n'a pas de forme particulière (pour un exemple, voir la section 3.2.3). Dans [4], les auteurs décrivent un algorithme efficace de réduction successive des réseaux en résolution ℓ, pour $\ell = 1, \ldots, L$, afin de trouver toutes les valeurs de t_ℓ. Mais cet algorithme requiert que B_1 soit de la forme $(1, f_2(z), \ldots, f_\ell(z))/P(z)$. Il existe plusieurs manières de trouver une base telle que B_1 soit de la forme voulue.

Une façon est par l'inversion d'un élément dans $\mathbb{F}_2[z]/P(z)$. Soit

$$Q_0 = G/P(z) = (g_1(z), \ldots, g_\ell(z))/P(z),$$

le vecteur obtenu de la récurrence. On peut obtenir la forme voulue de B_1 en multipliant composante par composante, dans $\mathbb{F}_2[z]/P(z)$, le vecteur G par l'inverse multiplicatif (dans $\mathbb{F}_2[z]/P(z)$) de g_1. Donc $B_1 = (1, g_2(z)g_1(z)^{-1}, \ldots, g_\ell(z)g_1(z)^{-1})/P(z)$. Il est à remarquer que le réseau décrit par la base $B_1, \ldots, B_\ell$ tel que donné par (41) est le même que celui décrit par la base $g(z)B_1, \ldots, B_\ell$ si la multiplication par $g(z)$ se fait dans $\mathbb{F}_2[z]/P(z)$ et si elle se fait composante par composante.

[4]Ce progiciel consiste en une librairie de fonctions qui permettent de déterminer l'équidistribution de générateurs. On en parle plus en détails à l'objectif 4.8.

Une autre manière de trouver le B_1 de la forme voulue se fait en analysant la matrice de transition X.

Proposition de recherche
Je propose de trouver des méthodes qui permettent de calculer, à partir de la matrice X de la récurrence, le vecteur $B_1 = (1, f_2(z), \ldots, f_k(z))/P(z)$ avec un algorithme qui ne nécessite pas l'inversion d'un élément. Quand k est très grand, l'inversion d'un élément dans $\mathbb{F}_2[z]/P(z)$ peut être long. Également, la multiplication de chacune des coordonnées de B_1 par l'inverse dans $\mathbb{F}_2[z]/P(z)$ n'est pas triviale.

Voici un exemple de ce type d'algorithme pour un LCG polynômial avec $s = 1$. On veut trouver le vecteur $B_1 = (f_1(z), \ldots, f_k(z))/P(z)$ tel que $f_1(z) = 1$. Supposons que $P(z)$ soit primitif sur $\mathbb{F}_2[z]$. Ceci implique que $a_k = 1$. En observant la récurrence (17), on remarque que $x_{n-1}^{(k-1)} = x_n^{(0)}$ pour $n \geq 0$. Ceci implique que

$$f_k(z) = z f_1(z) \bmod P(z) \tag{47}$$

puisque $j_k = j_1 + 1$ (en utilisant la notation de l'équation (39)). Pour les autres bits $0 \leq j < k - 1$ de la récurrence, on observe que

$$x_n^{(j)} = x_{n-1}^{(j-1)} + a_{k-j} x_{n-1}^{(k-1)} = x_{n-1}^{(j-1)} + a_{k-j} x_n^{(0)}$$

et on obtient

$$f_j(z) = z f_{j-1}(z) + a_{k-j+1} f_1(z) \bmod P(z). \tag{48}$$

On fixe $f_1(z) = 1$ et on utilise les équations (47) et (48) pour calculer les autres polynômes $f_j(z), 2 \leq j \leq k$. Si on calcule les $g_j(z)$ pour $j = 1, \ldots, L$, alors cet algorithme a un temps d'exécution dans $O(L)$. ∎

Objectif 8 : Implantation des méthodes dans le progiciel REGPOLY

Dans le cadre de ma maîtrise, j'ai commencé le développement du progiciel REGPOLY [36] qui permet de vérifier l'équidistribution de générateurs à récurrence linéaire modulo 2. Je vais continuer de travailler sur ce progiciel afin d'inclure les générateurs venant de ma recherche ainsi que les méthodes de vérification de l'équidistribution. Voici un résumé des nouveaux modules qui sont développés depuis mon inscription au doctorat :

- module **reduc** : implantation des méthodes de Couture et al. [4] et de Tezuka [45], pour déterminer l'équidistribution de générateurs.
- module **mersenne** : permet de vérifier l'équidistribution de générateurs de type Mersenne twister.
- module **doublepoly** : permet de vérifier l'équidistribution des GCL dans $\mathbb{F}_{2^w}[z]/P(z)$ où $P(z)$ est un trinôme..

- module **mrmmtrin** : permet de vérifier l'équidistribution des MRMM basés sur un trinôme dans $\mathbb{F}_{2^w}$.
- module **AC1D** : permet de vérifier l'équidistribution d'automates cellulaires à 1 et 2 dimensions.
- module **custom** : permet de vérifier l'équidistribution de n'importe quel type de générateurs en définissant sa matrice de transition X.
- module **polychar** : permet de déterminer le polynôme caractéristique de n'importe quel générateur à partir de son implantation. Ce module permet aussi de vérifier si un polynôme est irréductible et/ou primitif. On peut aussi afficher la matrice de transition de n'importe quel générateur.

J'ai également fait quelques ajouts dans certains modules :

- **mersenne**, **doublepoly**, **tausworthe**, **polynomial**, **tgfsr** : une fonction qui génère automatiquement des paramètres de générateurs qui sont de pleine période (i.e., dont le polynôme caractéristique est primitif).
- **mersenne**, **tausworthe**, **polynomial**, **tgfsr** : une fonction qui retourne de façon efficace le polynôme caractéristique des générateurs (i.e. plus rapidement que la méthode du module polychar).

Proposition de recherche
Je compte améliorer ce progiciel afin d'inclure les types générateurs que je développerai pendant ma recherche. Ce qui suit est une liste de modules que je vais inclure dans le cadre de mon doctorat (en plus de ceux mentionnés ci-haut).

- Un module pour calculer la q-valeur de réseaux digitaux.
- Un module pour inclure des réseaux digitaux qui ont des matrices $C^{(i)}$ arbitraires.
- Modifier le module **mecf** afin de pouvoir calculer l'équidistribution de réseaux digitaux qui ont des matrices $C^{(i)}$ arbitraires.
- Un module qui permet l'étude de réseaux digitaux extensibles, par rapport à la q-valeur des (q, k, t)-réseaux digitaux engendrés et à l'équidistribution.
- D'autres modules qui permettent de faire des recherches de générateurs selon d'autres critères de recherche, tel le critère P_α [42, 7].

■

Objectif 9 : Étude de l'utilisation de pré-calculs afin d'accélérer les implantations des générateurs

Dans [6], l'auteur utilise des tableaux de valeurs pré-calculées afin d'implanter de petits générateurs non linéaires. L'utilisation de ces tableaux permet de produire les nombres pseudo-aléatoires beaucoup plus rapidement, une fois toutes les valeurs calculées.

Dans ce projet de recherche, on explorera l'idée afin d'implanter les multiplications de vecteur de bits par des matrices à l'aide de tableaux de valeurs pré-calculées. Ceci pourra permettre d'implanter de façon rapide les récurrences et le tempering.

Par exemple, supposons que la multiplication par la matrice A, une matrice de bits de dimension $h \times w$, fait partie de l'implantation d'un générateur. Avant d'utiliser le générateur, on pré-calcule la multiplication $A\mathbf{v}$ pour toutes les 2^w valeurs de $\mathbf{v}$ possibles et on garde en mémoire les résultats (par exemple, dans un tableau). Par la suite, la multiplication par la matrice A se fait en un seul accès à la mémoire, quel que soit la complexité de la matrice A.

Évidemment, le gain en vitesse que procure ce genre d'implantation ne se fait pas sans coûts de mémoire. La mémoire nécessaire pour emmagasiner tous les 2^w résultats possibles est $h2^w$ bits. Par exemple, pour $h = w = 32$, il faut 16 Go de mémoire pour emmagasiner toutes les multiplications, ce qui est énorme. Pour des valeurs plus modeste comme $h = w = 16$, la mémoire nécessaire est tout de même de 128 Ko.

Une façon de pallier au problème causé par la grande quantité de mémoire nécessaire est d'effectuer la multiplication en un petit nombre c d'accès à la mémoire. Pour ce faire, on divise la matrice A en sous-matrices $A_1, A_2, \ldots, A_c$, la matrice A_i étant de dimension $h \times w_i$ de telle manière que

$$A = [A_1 : A_2 : \ldots : A_c]$$

et

$$\sum_{i=1}^{c} w_i = w.$$

Ainsi, la multiplication $A\mathbf{v}$ peut être effectuée par $A_1\mathbf{v}^1 + A_2\mathbf{v}^2 + \cdots + A_c\mathbf{v}^c$ où $\mathbf{v} = ({\mathbf{v}^1}^T, \ldots, {\mathbf{v}^c}^T)^T$ et $\mathbf{v}^i$ est un vecteur de w_i bits. Les multiplications par les matrices A_i sont pré-calculés. Dans ce cas, la mémoire nécessaire pour emmagasiner les multiplications par les matrices A_i est

$$h \sum_{i=1}^{c} 2^{w_i}$$

bits.

Prenons par exemple, le cas où $h = w = 32, c = 2$ et $w_1 = w_2 = 16$. La mémoire nécessaire est de 512 Ko et le nombre d'accès à la mémoire pour effectuer la multiplication est 2. On peut voir une nette amélioration de l'utilisation de la mémoire par rapport au cas où l'on ne fait qu'un seul accès à la mémoire. Dans le cas où $h = w = 32, c = 3, w_1 = w_2 = 11$ et $w_3 = 10$, la mémoire nécessaire est de 20 Ko seulement. Ces exemples nous montrent qu'il est possible d'utiliser des tables pré-calculés qui utilisent peu de mémoire et peu d'accès à la mémoire et ce, pour n'importe quelle matrice A.

Un autre problème des tables pré-calculées est le temps nécessaire pour les calculer. Une estimation optimiste voudrait que l'on puisse calculer 2×10^7 multiplications en 1 secondes sur un ordinateur de table courant. Dans le cas où $w = h = 32$ et $c = 1$, les 2^{32} multiplications se font en 215 secondes. Mais dans le cas où $w = h = 32, c = 3, w_1 = w_2 = 11$ et $w_3 = 10$, une

évaluation pessimiste (où l'on fait 10^5 multiplications par seconde) donne que les pré-calculs se font en 5×10^{-2} secondes. Grâce à ce type d'analyse, on peut déterminer s'il est mieux de pré-calculer à l'exécution ou d'effectuer les pré-calculs une seule fois, d'emmagasiner les résultat dans un fichier et de les charger à l'exécution.

Lors de l'atelier sur les générateurs de nombres pseudo-aléatoires qui s'est tenu au Centre de Recherche Mathématique du 17 au 28 juin 2002, Alexander Keller, un des participants de l'atelier, m'a fait la remarque suivante sur l'utilisation de tables de multiplications pré-calculées. Lors de l'exécution du générateur, il se peut que la performance du générateur dépende de l'application dans lequel le générateur est utilisé. En effet, afin d'être efficace, la table de multiplication doit être emmagasiner dans la mémoire cache de l'ordinateur. Si l'application utilise la mémoire cache de façon intensive, il se peut que les déplacements de la table de multiplication vers et hors de la mémoire cache ralentissent la vitesse de génération des nombres pseudo-aléatoires. Ce ralentissement éventuel sera étudié de plus près.

Proposition de recherche
Je propose d'étudier l'implantation de générateurs à l'aide de tables de valeurs pré-calculées. Je vais aussi étudier l'effet de l'utilisation de ce type de générateur sur l'efficacité d'une simulation qui utilise de façon intensive la mémoire cache. ∎

Objectif 10 : Développement de librairies contenant de bons générateurs

Pour la simulation Monte Carlo et l'intégration quasi-Monte Carlo, il est pratique d'avoir des librairies informatique contenant de bons générateurs. Par exemple, il existe la librairie de fonctions SSJ [14]. Cette librairie, implantée en langage Java, permet de faire de la simulation par événement ou par processus. Parmi les modules de SSJ, il y en a pour la génération de nombres pseudo-aléatoires. Par exemple, le module RandMRG implante un générateur à récurrence multiple dans $\mathbb{Z}/p$. Pour celui-ci, il existe une fonction qui permet d'initialiser le générateur à un état prédéterminé, une autre pour sauter dans la séquence d'un nombre s d'itérations, d'autres qui permettent de diviser la séquence en plusieurs sous-séquences (multi-séquencement) où chacune d'elles agissent comme générateurs indépendants, et plusieurs autres fonctions utiles pour la simulation. L'esprit de ce genre de fonctions est de rendre l'utilisation de générateurs facile et flexible à la fois. Il y a un autre librairie de fonctions qui est en cours de développement au laboratoire de simulation de l'Université de Montréal pour des ensembles de points hautement uniformes.

Proposition de recherche
Je propose de construire une librairie avec les bons générateurs que je trouverai dans le cadre de ma recherche. Cette librairie contiendra plusieurs utilitaires nécessaires pour ce genre d'applications (multi-séquencement, saut dans la récurrence, etc..). Il y aura des générateurs de petites périodes (de 2^7 à 2^{24}) et des générateurs de longues périodes ($> 2^{100}$). Cette librairie contiendra également des règles de réseaux extensibles (basés sur des récurrences ou non)

qui sont bons par rapport au critère d'équidistribution et à la q-valeur. Je propose aussi d'implanter les nouveaux générateurs et les nouveaux ensembles de points dans les deux librairies mentionnées ci-haut. ■

5 Conclusion

Je crois que les objectifs cités dans la dernière section de ce document constituent des axes de recherches originaux et que le résultat de cette recherche permettra de faire avancer la science pour ce qui est des méthodes de générations de nombres pseudo-aléatoires et d'ensembles de points uniformes.

Les générateurs basés sur des récurrences dans $\mathbb{F}_{2^w}$ semblent prometteurs. Ils possèdent une matrice de transition plus complexe que la plupart des générateurs connus. Durant mon travail de maîtrise, j'ai pu observer que plus la matrice de transition était complexe mieux était l'équidistribution. Durant mon doctorat, j'analyserai cette famille de récurrences et d'autres également qui, j'en suis convaincu, permettra d'obtenir de bons ensembles de points.

Parallèlement à ma recherche sur les récurrences, je continuerai le développement du progiciel REGPOLY. Celui-ci constitue un outil difficile à développer, mais important pour l'analyse d'ensembles de points. À la conclusion de mon doctorat, il offrira la possibilité de vérifier la qualité d'ensembles de points produits par tout nouveau design qui entre dans le cadre des (q, k, t)-réseaux digitaux en base 2, ceux-ci pouvant être basés sur des récurrences ou non. L'ensemble des designs possibles est vaste, d'où l'importance, pour la communauté scientifique, d'avoir un outil pour les analyser efficacement.

En parlant aux gens qui utilisent les générateurs de nombres pseudo-aléatoires, je me suis rendu compte à quel point ce domaine de la science est méconnu. En pratique, les gens utilisent le premier générateur de nombres pseudo-aléatoires qui leur tombe sur la main. Ce premier générateur n'est pas nécessairement bon. En fait, les générateurs les plus répandus sont souvent les plus mauvais [13]. Les utilisateurs ne semblent pas sensibilisés à l'importance d'un bon générateur de nombres pseudo-aléatoires. Pourtant, les conséquences peuvent être désastreuses dans leurs applications puisqu'ils assument, à tort, que les nombres produits par le générateur sont des variables aléatoires uniformes et indépendantes. Je vois un grand besoin d'éduquer la communauté scientifique à l'utilité des bons générateurs. Mais cet sensibilisation ne peut se faire que si de bons générateurs sont accessibles. Si un utilisateur désire utiliser un bon générateur, il faut lui rendre la vie facile en mettant ceux-ci à sa disposition. En fait, un autre but de ma recherche est de donner, à de potentiels utilisateurs qui refusent la médiocrité, la possibilité d'utiliser de bons générateurs sans trop d'efforts de leur part.

Annexe A : Arithmétique dans $\mathbb{F}_{2^w}$

Pour représenter un élément dans $\mathbb{F}_{2^w}$, il faut choisir une base ordonné $(\beta_1, \ldots, \beta_w)$ de $\mathbb{F}_{2^w}$ sur $\mathbb{F}_2$ où les $\beta_i \in \mathbb{F}_{2^w}, 1 \leq i \leq w$. Ceci permet d'avoir une représentation des éléments de $\mathbb{F}_{2^w}$ sur ordinateur. Grâce à cette base, on peut représenter n'importe quel élément $\alpha \in \mathbb{F}_{2^w}$ par

$$\alpha = \alpha_1\beta_1 + \cdots + \alpha_w\beta_w \tag{49}$$

où $\alpha_i \in \mathbb{F}_2, 1 \leq i \leq w$.

On peut maintenant représenter un élément $\alpha \in \mathbb{F}_{2^w}$ par le vecteur de bits $\mathbf{v} = (\alpha_1, \ldots, \alpha_w)^T$. A noter que cette représentation dépend de la base ordonnée choisie.

Il existe deux types de bases couramment utilisées dans la littérature : la base polynômiale et la base normale. Soit $\zeta \in \mathbb{F}_{2^w}$, un générateur de $\mathbb{F}_{2^w}$. La base polynômiale est $(1, \zeta, \zeta^2, \ldots, \zeta^{w-1})$. Si chacune des composantes de $(\zeta, \zeta^2, \zeta^{2^2}, \ldots, \zeta^{2^{w-1}})$ sont linéairement indépendantes, alors celui-ci constitue une base normale [21].

Soit $\alpha \in \mathbb{F}_{2^w}$, $\mathbf{v}$ son vecteur de bits correspondant sous la base polynômiale et $M(z) = z^w + \sum_{i=0}^{w-1} a_i z^i$, le polynôme minimal de ζ sur $\mathbb{F}_2$. Ce polynôme est le polynôme de moindre de degré tel que $M(\zeta) = 0$ dans $\mathbb{F}_{2^w}$. Si le polynôme $M(z)$ est primitif, alors n'importe quel élément de $\alpha \in \mathbb{F}_{2^w}$, sauf le zéro, peut être représenté par ζ^s pour une valeur de s.

La multiplication dans $\mathbb{F}_{2^w}$ est linéaire, ce qui veut dire qu'on peut effectuer la multiplication par ζ par une transformation linéaire $T : \mathbb{F}_{2^w} \to \mathbb{F}_{2^w}$. Si on représente les éléments de $\mathbb{F}_{2^w}$ par des vecteurs de bits dans $\mathbb{F}_2^w$, alors la multiplication par ζ se fait par la multiplication par une matrice A. Pour déterminer cette matrice A, il suffit de déterminer comment elle transforme les vecteurs de la base polynômiale. Soit $\mathbf{e}_i$, le vecteur représentant l'élément $\zeta^{i-1}, i = 1, \ldots, w$. Il est facile de voir que $A\mathbf{e}_i = \mathbf{e}_{i+1}$ pour $i = 1, \ldots, w-1$. Puisque $M(\zeta) = 0$, on sait que $\zeta^w = \sum_{i=0}^{w-1} a_i\zeta^i$. Donc $A\mathbf{e}_w = \mathbf{a} = (a_0, \ldots, a_{w-1})^T$. On peut maintenant déduire que

$$A = \begin{pmatrix} \mathbf{e}_2 & : & \ldots & : & \mathbf{e}_w & : & \mathbf{a} \end{pmatrix} = \begin{pmatrix} & & & & a_0 \\ 1 & & & & a_1 \\ & 1 & & & a_2 \\ & & \ddots & & \vdots \\ & & & 1 & a_{w-1} \end{pmatrix}. \tag{50}$$

La multiplication de η par ζ^s peut être effectuée par une composition de s transformations linéaires T appliquées à η dans $\mathbb{F}_{2^w}$. Ceci implique que, dans la représentation vectorielle, on effectue cette opération en multipliant le vecteur représentant η par A^s.

Annexe B : Définition de réseau polynômial

Cette annexe contient plusieurs définitions qui sont données sans commentaires et qui aboutissent à la définition de réseau polynômial.

Définition B.1 *(Connell [3])*
Soit R un anneau et M un groupe abélien additif. M est un R-module s'il existe une multiplication scalaire $M \times R \to M$ qui satisfait les propriétés suivantes

$$\begin{array}{ccccc} r(a_1+a_2) & = & (a_1+a_2)r & = & a_1r+a_2r \\ (r_1+r_2)a & = & a(r_1+r_2) & = & ar_1+a_r2 \\ (r_1r_2)a & = & a(r_1r_2) & = & (ar_1)r_2 \\ \underline{1}a & = & a\underline{1} & = & a \end{array}$$

où $a, a_1, a_2 \in M$, $r, r_1, r_2 \in R$ et $\underline{1}$ est l'élément neutre multiplicatif de R.

Définition B.2 *(Connell [3])*
Une base est un ensemble B d'éléments de M qui sont linéairement indépendants et qui génèrent M.

Définition B.3 *(Connell [3])*
Un R-module M est dit libre s'il existe une base pour M.

Définition B.4 *(Connell [3])*
Un sous-ensemble de N de M est un R-sous-module de M si N est un module.

Définition B.5 *(Tezuka [45])*
Soit $R = \mathbb{F}_2[z]$ et $M = \mathbb{F}_2[[z]]^t$. Un réseau polynômial est un R-sous-module libre de M.

Références

[1] A. Compagner. Definitions of randomness. *American Journal of Physics*, 59 :700–705, 1991.

[2] A. Compagner. The hierarchy of correlations in random binary sequences. *Journal of Statistical Physics*, 63 :883–896, 1991.

[3] E.H. Connell. *Elements of Abstract and Linear Algebra*. Sur l'internet, à l'adresse `http:www.math.miami.edu/~ec/book/`, 2001.

[4] R. Couture et P. L'Ecuyer. Lattice computations for random numbers. *Mathematics of Computation*, 69(230) :757–765, 2000.

[5] R. Couture, P. L'Ecuyer, et S. Tezuka. On the distribution of k-dimensional vectors for simple and combined Tausworthe sequences. *Mathematics of Computation*, 60(202) :749–761, S11–S16, 1993.

[6] J. Granger-Piché. Générateurs pseudo-aléatoires combinant des récurrences linéaires et non linéaires. Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, 2001.

[7] F. J. Hickernell. Lattice rules : How well do they measure up ? Dans P. Hellekalek et G. Larcher, éditeurs, *Random and Quasi-Random Point Sets*, volume 138 de *Lecture Notes in Statistics*, pages 109–166. Springer, New York, 1998.

[8] F. J. Hickernell, H. S. Hong, P. L'Ecuyer, et C. Lemieux. Extensible lattice sequences for quasi-Monte Carlo quadrature. *SIAM Journal on Scientific Computing*, 22(3) :1117–1138, 2001.

[9] A. M. Law et W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, troisième édition, 2000.

[10] P. L'Ecuyer. Uniform random number generation. *Annals of Operations Research*, 53 :77–120, 1994.

[11] P. L'Ecuyer. Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation*, 65(213) :203–213, 1996.

[12] P. L'Ecuyer. Random number generation. Dans Jerry Banks, éditeur, *Handbook of Simulation*, pages 93–137. Wiley, 1998.

[13] P. L'Ecuyer. Software for uniform random number generation : Distinguishing the good and the bad. Dans *Proceedings of the 2001 Winter Simulation Conference*, pages 95–105, Pistacaway, NJ, 2001. IEEE Press.

[14] P. L'Ecuyer. *SSJ : A Java Library for Stochastic Simulation*, 2001. Guide d'utilisateur.

[15] P. L'Ecuyer et C. Lemieux. Recent advances in randomized quasi-monte carlo methods. Dans M. Dror, P. L'Ecuyer, et F. Szidarovszki, éditeurs, *Modeling Uncertainty : An Examination of Stochastic Theory, Methods, and Applications*, pages 419–474. Kluwer Academic Publishers, Boston, 2002.

[16] P. L'Ecuyer et F. Panneton. Construction of equidistributed generators based on linear recurrences modulo 2. Dans K.-T. Fang, F. J. Hickernell, et H. Niederreiter, éditeurs, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 318–330, Berlin, 2002. Springer-Verlag.

[17] C. Lemieux. *L'utilisation de règles de réseau en simulation comme technique de réduction de la variance*. Thèse de doctorat, Université de Montréal, mai 2000.

[18] C. Lemieux et P. L'Ecuyer. Randomized polynomial lattice rules for multivariate integration and simulation. Disponible à `http://www.iro.umontreal.ca/~lecuyer`, 2001.

[19] A. K. Lenstra. Factoring multivariate polynomials over finite fields. *Journal of Computer and System Sciences*, 30 :235–248, 1985.

[20] T. G. Lewis et W. H. Payne. Generalized feedback shift register pseudorandom number algorithm. *Journal of the ACM*, 20(3) :456–468, 1973.

[21] R. Lidl et H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, Cambridge, édition revisée, 1994.

[22] K. Mahler. An analogue to Minkowski's geometry of numbers in a field of series. *Annals of Mathematics*, 42(2) :488–522, 1941.

[23] K. Mahler. On a theorem in the geometry of numbers in a space of laurent series. *J. Number Theory*, 17 :403–416, 1983.

[24] M. Matsumoto. Simple cellular automata as pseudo-random m-sequence generators for built-in self test. *ACM Transactions on Modeling and Computer Simulation*, 8(1) :31–42, 1998.

[25] M. Matsumoto et Y. Kurita. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation*, 2(3) :179–194, 1992.

[26] M. Matsumoto et Y. Kurita. Twisted GFSR generators II. *ACM Transactions on Modeling and Computer Simulation*, 4(3) :254–266, 1994.

[27] M. Matsumoto and Y. Kurita. Strong deviations from randomness in m-sequences based on trinomials. *ACM Transactions on Modeling and Computer Simulation*, 6(2) :99–106, 1996.

[28] M. Matsumoto et T. Nishimura. Mersenne twister : A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1) :3–30, 1998.

[29] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1992.

[30] H. Niederreiter. Factorization of polynomials and some linear-algebra problems over finite fields. *Linear Algebra Appl.*, 192 :301–328, 1993.

[31] H. Niederreiter. The multiple-recursive matrix method for pseudorandom number generation. *Finite Fields and their Applications*, 1 :3–30, 1995.

[32] H. Niederreiter. Pseudorandom vector generation by the multiple-recursive matrix method. *Mathematics of Computation*, 64(209) :279–294, 1995.

[33] H. Niederreiter. The existence of good extensible polynomial lattice rules. *À paraître*, 2002.

[34] T. Nishimura. Tables of 64-bit Mersenne twisters. *ACM Transactions on Modeling and Computer Simulation*, 10(4) :348–357, 2000.

[35] N. Packard et S. Wolfram. Two-dimensional cellular automata. *J. Statistical Physics*, 38,5/6 :901–946, 1985.

[36] F. Panneton. Générateurs de nombres aléatoires utilisant des récurrences linéaires modulo 2. Mémoire de maîtrise, Département d'informatique et de recherche opérationnelle, Université de Montréal, 2000.

[37] K. Paul, S.P. Chaudhuri, R. Ghosal, B. Sikdar, et D.R. Chowdhury. GF(2^p) CA based vector quantization for fast encoding of still images. Dans *Proc. of VLSI Design*, India, 2000.

[38] K. Paul et D.R. Chowdhury. Applications of GF(2^p) CA in burst error correcting codes generator. Dans *Proc. of VLSI Design*, India, 2000.

[39] G. Pirsic et W. Ch. Schmid. Calculation of the quality parameter of digital nets and application to their construction. *Journal of Complexity*, 2001. À paraître.

[40] W. Ch. Schmid. The exact quality parameter of nets derived from sobol' and niederreiter sequences. *Recent Advances in Numerical Methods and Applications*, pages 287–295, 1999.

[41] B.K. Sikdar, K. Paul, G.P. Biswas, C.Yang, V. Bopanna, S.Mukherjee, et P.P. Chaudhuri. Theory and applications of GF(2^p) cellular automata as on-chip test pattern generator. Dans *Proc. of VLSI Design*, pages 556–561, India, 2000.

[42] I. H. Sloan et S. Joe. *Lattice methods for multiple integration.* Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1994.

[43] R. C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19 :201–209, 1965.

[44] S. Tezuka. Random number generation based on the polynomial arithmetic modulo two. Rapport technique RT-0017, IBM Research, Tokyo Research Laboratory, octobre 1989.

[45] S. Tezuka. The k-dimensional distribution of combined gfsr sequences. *Mathematics of Computation*, 62(206) :809–817, 1994.

[46] S. Tezuka. *Uniform Random Numbers : Theory and Practice.* Kluwer Academic Publishers, Norwell, Mass., 1995.

[47] S. Tezuka et M. Fushimi. A method of designing cellular automata as pseudorandom number generators for built-in self-test for vlsi. *Contemp. Math.*, 168 :363–367, 1994.

[48] S. Wolfram. Statistical mechanics of cellular automata. *Rev. Modern Physics*, 55 :601–644, 1983.