Université de Montréal

Générateurs pseudo-aléatoires combinant des récurrences linéaires et non linéaires

par

Jacinthe Granger-Piché

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

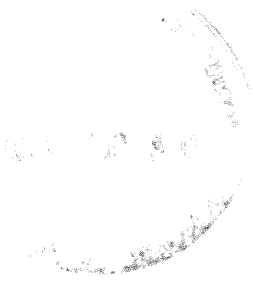
Mémoire présenté à la Faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique option recherche opérationnelle

juin 2001



Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé:

Générateurs pseudo-aléatoires combinant des récurrences linéaires et non linéaires

présenté par:

Jacinthe Granger-Piché

a été évalué par un jury composé des personnes suivantes:

Gilles Brassard

(président-rapporteur)

Pierre L'Écuyer

(directeur de recherche)

Balázs Kégl

(membre du jury)

Mémoire accepté le:

27 août 2001

Sommaire

Dans ce mémoire, nous nous intéressons aux générateurs pseudo-aléatoires combinant des récurrences de différents types, en portant une attention particulière aux combinaisons de récurrences linéaires et non linéaires. Des résultats généraux concernant deux façons de combiner des générateurs, soient par l'addition des sorties modulo 1 et par un ou-exclusif bit à bit des sorties, sont donnés. La structure de l'ensemble des vecteurs formés de t valeurs successives produites par deux types de générateurs combinés particuliers est aussi présentée. Nous nous intéressons d'abord à l'addition modulo 1 des sorties d'un générateur à récurrence linéaire multiple (MRG pour «multiple recursive generator») et d'un deuxième générateur quelconque, puis nous examinons la combinaison d'un générateur de Tausworthe (aussi appelé LFSR pour «linear feedback shift register») avec un autre générateur au moyen d'un ou-exclusif bit à bit. Nous sommes intéressés en particulier au cas où le deuxième générateur, qui peut être quelconque, est non linéaire.

Nous formons ensuite plusieurs familles de générateurs combinés, la plupart ayant une composante linéaire (un MRG ou un LFSR) et une petite composante non linéaire. Nous essayons aussi quelques combinaisons de MRG et de LFSR. Plusieurs tests statistiques considérés comme «costauds» sont appliqués aux différentes familles. Les résultats montrent que les combinaisons proposées permettent de faire des gains non négligeables, par rapport aux MRG ou aux LFSR combinés entre eux, et ce pour tous les tests appliqués.

Nous suggérons finalement quelques bons générateurs combinés, et discutons de leur implantation. Des temps d'exécution comparables à ceux de MRG ou de LFSR connus et considérés comme rapides sont obtenus dans certains cas.

Table des matières

Liste des figures

Liste des tableaux

Remerciements

J'aimerais d'abord remercier mon directeur de maîtrise, Pierre L'Écuyer, pour tout le temps qu'il a bien voulu me consacrer, et les bonnes idées qu'il m'a données. J'ai beaucoup apprécié sa disponibilité, et la qualité de son encadrement.

Je remercie aussi le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), la fondation Marc Bourgie et le bureau d'avocats Goodman Phillips & Vineberg, pour l'aide financière qu'ils m'ont apportée.

Je tiens à mentionner l'aide précieuse de Richard Simard, qui a su répondre avec justesse à plusieurs de mes questions, tant au point de vue informatique que mathématique. Merci aussi à Jasmin Frenette, pour ses nombreuses suggestions de programmation et ses encouragements constants, ainsi qu'à Renée Touzin, qui a bien voulu lire mon mémoire et me faire plusieurs commentaires très pertinents.

J'aimerais finalement remercier mes parents, Denise Granger et Victor Piché, ainsi que tous les autres membres de ma famille, pour leurs judicieux conseils et leur soutien si apprécié.

Chapitre 1

Introduction

En simulation, la génération de nombres aléatoires est nécessaire pour simuler des observations au hasard, provenant de différentes lois de probabilités. D'habitude, des algorithmes produisant des séquences périodiques de nombres ayant «l'air» indépendants et tirés d'une loi uniforme sur l'intervalle [0,1] (loi notée U(0,1)) sont utilisés, puis les nombres produits sont ensuite transformés, selon les besoins, pour simuler des variables aléatoires de différentes distributions. Comme ces algorithmes, appelés générateurs de nombres pseudo-aléatoires U(0,1), sont à la base de la génération de toutes les données aléatoires d'un problème, il est crucial de bien étudier leurs propriétés et de les choisir avec soin.

Les générateurs les plus connus et répandus se servent de récurrences linéaires, pour produire une nouvelle sortie à partir d'une séquence de nombres générée précédemment. Des séquences de longues périodes peuvent être produites avec ces générateurs et des algorithmes très rapides sont disponibles, ce qui est souhaitable. Les nombres générés ont cependant une structure très régulière, qui peut devenir visible si une trop grande quantité de sorties est utilisée. Dans ce cas, les nombres produits n'ont plus «l'air» aléatoire, ce qui peut être problématique.

D'autres générateurs utilisent plutôt des récurrences non linéaires pour produire de nouvelles sorties. Ces générateurs permettent de générer des nombres dont la structure est moins régulière, ou moins facilement détectable. Par contre, ils sont habituellement beaucoup plus lents, ce qui pose un problème dans le cas où, par exemple, la majorité du temps de calcul d'une application est due à la génération des nombres aléatoires.

Dans ce mémoire, nous étudions des générateurs combinant des récurrences linéaires

avec des récurrences non linéaires. Nous espérons ainsi améliorer la structure trop régulière des générateurs linéaires, tout en développant des algorithmes plus rapides que ceux des générateurs complètement non linéaires. Une étude des propriétés théoriques de plusieurs familles de générateurs combinant des récurrences différentes est présentée, et des résultats empiriques sont donnés.

Avant de donner un plan détaillé du mémoire, quelques notions de bases doivent être introduites. La prochaine section définit de façon plus formelle ce qu'est un générateur de nombres pseudo-aléatoires, et les caractéristiques d'un bon générateur sont détaillées à la section 1.2. Le plan du mémoire est ensuite donné, de même que la contribution apportée.

1.1 Définitions

Un générateur de nombres pseudo-aléatoires peut être défini de façon plus formelle comme suit :

Définition 1.1.1 [27] *Un générateur est une structure $\mathcal{G} = (S, \mu, T, U, G)$, où S est un ensemble fini d'états, μ est une loi de probabilité sur S, $T : S \to S$ est une fonction de transition, U est un ensemble fini de symboles de sorties et $G : S \to U$ est une fonction de sortie.*

L'état initial s_0 d'un générateur, souvent appelé germe, est d'abord choisi selon la loi de probabilité μ. Ensuite, l'état du générateur évolue selon $s_n = T(s_{n-1})$, pour $n \geq 1$. À chaque étape n, la sortie $u_n \in U$ produite par le générateur est obtenue en appliquant la fonction G à l'état courant : $u_n = G(s_n)$. Les u_n ainsi obtenus sont les nombres produits par le générateur. Il est important de remarquer qu'une fois le germe choisi, la séquence de nombres produite par le générateur est entièrement déterminée, d'où l'appelation générateurs de nombres «pseudo»-aléatoires.

Les sorties u_n, bien qu'entièrement fixées une fois le germe choisi, doivent se comporter, vues de l'extérieur, comme des variables aléatoires i.i.d. (indépendantes et iden-

tiquement distribuées) tirées d'une certaine distribution spécifiée. Dans ce mémoire, la distribution considérée est la loi uniforme sur l'intervalle [0,1], notée U(0,1). Cette loi a une très grande importance en simulation, puisqu'il existe plusieurs techniques pour convertir des variables aléatoires U(0,1) en variables aléatoires de différentes distributions (comme une normale, une gamma, une binomiale, etc.). Ainsi, un bon générateur U(0,1) peut facilement être transformé en bon générateur d'une autre loi, en suivant par exemple les procédures décrites dans [4, 10].

Le nombre d'états possibles appartenant à l'ensemble S de la définition 1.1.1 est fini. Ceci implique qu'après un certain nombre d'étapes, l'état du générateur prend une valeur déjà rencontrée, et une séquence périodique est obtenue. La longueur des séquences se répétant correspond à la «période» d'un générateur, comme l'indique la définition suivante.

Définition 1.1.2 [13] *La période d'un générateur $\mathcal{G}$ est le plus petit entier positif ρ tel qu'à partir d'un certain entier $\tau \geq 0$, $s_{\rho+n} = s_n$, pour tout $n \geq \tau$.*

Une conséquence directe du nombre fini d'états est le nombre fini de symboles de sorties. Dans le cas des générateurs U(0,1), ces symboles de sorties prennent des valeurs entre 0 et 1.

Regardons un exemple simple de générateur U(0,1), pour mieux comprendre les définitions données. Un type de générateur très connu, décrit en détail dans [9], se nomme «générateur à congruence linéaire» (ou LCG pour «linear congruential generator»), à cause de la forme particulière de sa fonction de transition T. Il utilise la récurrence $x_n = T(x_{n-1}) = ax_{n-1} \bmod m$, où mod désigne l'opération modulo, m est un grand entier positif, et a est une constante entière choisie judicieusement. Une valeur initiale entre 0 et $m-1$ est donnée à x_0, puis à chaque étape $n \geq 1$, les x_n sont obtenus avec la fonction de transition T. Ces x_n, qui prennent des valeurs entières entre 0 et $m-1$, sont ensuite divisés par m pour produire des nombres entre 0 et 1. La sortie obtenue, à l'étape n, est donc $u_n = G(x_n) = x_n/m$.

Dans ce cas, l'ensemble des valeurs de sorties possibles est donné par $U = \{0, 1/m, 2/m, ..., (m-1)/m\}$. L'état s_n du générateur coïncide avec la valeur de x_n, et l'ensemble des états possibles est $S = \{0, 1, ..., m-1\}$. Comme il n'y a que m états possibles, la période de ce générateur ne peut pas dépasser m.

1.2 Caractéristiques d'un bon générateur

Un bon générateur doit obéir à un certain nombre de critères. D'abord, les valeurs u_n produites par le générateur doivent se comporter, jusqu'à un certain point, comme des variables aléatoires i.i.d., dans notre cas provenant d'une loi U(0,1). Idéalement, nous voudrions qu'il soit impossible de faire la différence entre une séquence produite par le générateur, et une vraie séquence de nombres aléatoires. Entre autres, tout comme pour une vraie séquence de nombres aléatoires, nous aimerions qu'il soit pratiquement impossible, en observant une séquence particulière, de déterminer quel sera le prochain nombre généré. Or, comme la séquence produite par un générateur est périodique, il est clair que les nombres finiront par se répéter, devenant alors faciles à prédire. Il n'est donc pas réaliste de chercher des générateurs produisant des séquences de nombres, à tout le moins des séquences «infinies», ne pouvant pas être distinguées d'une vraie séquence de nombres aléatoires. Ce qui est recherché est plutôt le fait que le générateur puisse produire une séquence de nombres très difficile à distinguer d'une vraie séquence de nombres aléatoires, le plus «longtemps» possible.

En pratique, plusieurs tests statistiques ont été développés pour comparer les sorties u_n produites par un générateur avec des variables aléatoires i.i.d. de loi U(0,1). Encore une fois, le but recherché n'est pas de produire des séquences de nombres pouvant passer tous les tests statistiques possibles, peu importe la longueur de la séquence, puisque ceci est impossible. Nous aimerions plutôt qu'aucun algorithme simple ne puisse détecter une différence, avec un temps de calcul et une mémoire raisonnables. Bien sûr, cette notion de «raisonnable» évolue avec le temps, puisque les ordinateurs sont de plus en plus puissants, et possèdent de plus en plus de mémoire.

Une condition nécessaire pour qu'un générateur se comporte bien face aux tests statistiques est qu'il possède une longue période. En fait, pour plusieurs générateurs, une différence entre les valeurs produites et une vraie séquence aléatoire commence à être détectée avant que la période ne soit épuisée. Par exemple, pour un générateur de type LCG tel que présenté dans la section précédente, il existe des tests statistiques relativement simples qui commencent à détecter une différence lorsque seulement la racine cubique de la période est utilisée [30]. Il est donc crucial de développer des générateurs avec de très grandes périodes, particulièrement pour les applications qui nécessitent la génération d'un très grand nombre de variables aléatoires.

Un bon générateur doit aussi être efficace. Il doit être rapide, et utiliser le moins de mémoire possible. Encore une fois, lorsqu'un grand nombre de variables aléatoires doit être généré dans un programme, la rapidité du générateur peut avoir un impact important sur la vitesse d'exécution du programme. De même, si plusieurs générateurs sont utilisés en parallèle, la mémoire consommée peut devenir un facteur important.

Plusieurs autres critères sont mentionnés dans [13, 16, 18]. Entre autres, un générateur doit pouvoir être programmé facilement, dans un langage de haut-niveau, et être portable. Aussi, il est souvent utile de pouvoir répéter une séquence déjà produite, par exemple pour vérifier qu'un programme s'exécute correctement. Heureusement, tous les générateurs de nombres pseudo-aléatoires possèdent cette propriété, puisqu'il suffit de reprendre un même état initial s_0 pour obtenir une séquence identique.

Finalement, la construction de bons générateurs demande une bonne connaissance théorique. Knuth dit d'ailleurs dans [9] : «Les nombres aléatoires ne doivent pas être générés avec une méthode choisie aléatoirement». Il existe une très grande quantité de tests statistiques pouvant être appliqués à une séquence de nombres générés. Faire passer quelques tests statistiques augmente certainement notre confiance envers un générateur, mais ne garantit pas que les séquences produites ressemblent suffisamment à des séquences aléatoires pour être utilisées. En fait, le test ultime consisterait à vérifier, dans le contexte précis où sont utilisés les nombres générés, si la différence entre la séquence produite et une vraie séquence de valeurs aléatoires change de façon

significative les résultats obtenus. Cependant, ce test n'est pas réalisable en pratique, puisque les applications sont souvent très complexes. De plus, il est difficile de tenir compte de toutes les applications possibles, puisqu'elles sont très variées. Il faut donc se tourner vers des analyses plus théoriques pour augmenter notre confiance envers un générateur.

En général, la structure mathématique des nombres générés sur la période entière est étudiée. Plus spécifiquement, l'ensemble de tous les vecteurs de t valeurs successives pouvant être générées à partir de tous les états initiaux $s_0 \in S$ est formé et analysé. Dans le cas des générateurs U(0,1), cet ensemble doit être bien réparti dans l'hypercube $[0,1]^t$, et ce pour différentes valeurs de t. Pour plusieurs familles de générateurs, des mesures d'uniformité de cet ensemble ont été développées [9, 13, 15, 16, 19, 26], permettant de choisir adéquatement les paramètres des récurrences (par exemple, les valeurs de a, c et m dans le cas du LCG décrit dans la section précédente), et d'évaluer la qualité des générateurs.

Une fois la structure des valeurs générées connue, et les paramètres choisis selon certains critères, les tests statistiques deviennent importants. Il est suggéré de faire passer des tests variés, qui pourront être choisis en tenant compte des limites du générateur. Ces tests permettent de vérifier comment se comportent des séquences de nombres successifs générés, dans certaines situations empiriques.

1.3 Plan du mémoire

Les générateurs les plus utilisés et documentés possèdent une fonction de transition T linéaire, de même qu'une fonction de sortie G linéaire. Le chapitre suivant est consacré à ces types de générateurs, dits «linéaires». Deux familles de générateurs sont considérées : les MRG (pour «multiple recursive generator»), qui sont en fait une généralisation des LCG décrits à la section 1.1, et les générateurs LFSR (pour «linear feedback shift register»), aussi appelés générateurs de Tausworthe. La structure des nombres produits par ces générateurs est bien connue, et nous verrons les mesures

d'uniformité utilisées, en pratique, pour choisir les paramètres des récurrences. Ces générateurs sont généralement très rapides, à cause de la forme simple des fonctions T et G. Cependant, les nombres générés sur la période entière possèdent une structure très régulière. Ceci pose un problème si une grande portion de la période est utilisée, puisque la structure devient alors visible, et les nombres générés ne se comportent plus comme des valeurs aléatoires. Il faut donc développer des générateurs ayant de très grandes périodes, pour s'assurer que même dans les applications demandant une très grande quantité de valeurs aléatoires, la structure n'affectera pas les résultats calculés.

D'autres générateurs utilisent soit une fonction de transition T non linéaire, soit une fonction de sortie G non linéaire. Ces générateurs, dits «non linéaires», sont présentés au chapitre 3. Plusieurs familles sont considérées : les générateurs à congruence quadratique et cubique (dont le nom est tiré de la forme de la fonction de transition T), et les générateurs inversifs, qui «inversent» la valeur obtenue avant de produire une sortie. Ces générateurs sont généralement plus lents que les générateurs linéaires, à cause des fonctions T ou G qui sont plus compliquées, et plus longues à calculer. Aussi, la structure des nombres produits par ces générateurs n'est pas bien connue, puisqu'elle est très complexe. Cependant, plusieurs mesures théoriques nous indiquent que les nombres produits par certaines des familles présentées se comportent de façon très similaire à de vrais nombres aléatoires, même si la période entière est considérée. Les tests empiriques témoignent également des bonnes propriétés de ces générateurs.

Le chapitre 4 présente des exemples de générateurs combinés souvent utilisés et documentés. Ces générateurs sont construits en combinant plusieurs générateurs d'une même famille, comme c'est le cas pour les MRG combinés, les LFSR combinés, ou encore les générateurs cubiques combinés. Nous verrons que les combinaisons permettent d'atteindre des périodes beaucoup plus longues, tout en gardant une bonne vitesse de génération, et que la structure des nombres produits peut s'améliorer.

Bien que les combinaisons de générateurs d'une même famille soient très étudiées et utilisées, très peu de documentation est disponible sur des combinaisons de générateurs provenant de familles différentes. Nous avons donc développé de la nouvelle théorie, qui

est présentée au chapitre 5. La première section donne des résultats généraux concernant deux façons de combiner des générateurs, peu importe leur famille. Dans un premier temps, la sortie d'un générateur est additionnée à la sortie d'un deuxième générateur, puis le résultat est ramené entre 0 et 1 par l'opération modulo 1. Ensuite, les sorties de deux générateurs quelconques sont combinées à l'aide d'un ou-exclusif bit à bit. Les sections suivantes présentent la théorie plus spécifique à deux types de générateurs combinés que nous avons décidé d'étudier. Nous analysons d'abord la structure des nombres produits par un générateur combiné construit comme suit : la sortie d'un MRG est additionnée, modulo 1, à celle d'un deuxième générateur quelconque. Nous analysons ensuite la structure des nombres produits par un générateur combiné dont une des composantes est un LFSR. La sortie du générateur combiné est obtenue cette fois en faisant un ou-exclusif entre la sortie du LFSR et la sortie d'un deuxième générateur. Bien sûr, dans ce mémoire, nous nous sommes intéressés au cas où le deuxième générateur, qui peut être quelconque, est non linéaire.

Plusieurs familles de générateurs combinés sont ensuite formées, la plupart ayant une composante linéaire (soit un MRG, soit un LFSR) et une petite composante non linéaire. Quelques combinaisons de MRG avec des LFSR sont aussi essayées. Des tests statistiques sont appliqués à ces différentes familles, et les résultats sont présentés au chapitre 6. Les tests statistiques détectant le plus facilement des défauts dans les séquences produites par les générateurs linéaires ou non linéaires utilisés ont été sélectionnés. Nous verrons que certains générateurs combinés se comportent très bien par rapport à ces tests.

Des exemples de bons générateurs combinés sont ensuite présentés au chapitre 7. L'implantation de ces générateurs est discutée, et les temps d'exécution sont comparés avec ceux de générateurs considérés comme rapides.

Finalement, les principales conclusions sont résumées au dernier chapitre. Des pistes de recherche sont aussi suggérées, afin d'orienter les travaux futurs.

1.4 Contribution

Comme mentionné précédemment, les combinaisons de générateurs provenant de familles différentes ont été très peu étudiées jusqu'à maintenant. Les résultats théoriques et empiriques présentés aux chapitres 5 et 6 sont nouveaux, et constituent la contribution principale de ce mémoire. Ces résultats ont permis de développer de bons générateurs ayant de meilleures propriétés statistiques que les MRG ou les LFSR combinés souvent utilisés. Aussi, certaines des implantations suggérées au chapitre 7 permettent d'obtenir des temps d'exécution comparables aux temps obtenus avec des MRG ou des LFSR combinés considérés comme rapides. Les générateurs présentés sont donc d'un grand intérêt, particulièrement dans un contexte de simulation, où une très grande quantité de nombres ayant de bonnes propriétés statistiques doivent être générés.

Au niveau plus informatique, un logiciel nommé «TestU01», développé par une équipe du laboratoire de simulation de l'Université de Montréal, a été utilisé pour appliquer les tests statistiques présentés au chapitre 6. Deux modules ont été ajoutés à ce logiciel, pour compléter les familles de générateurs disponibles. Grâce à ces deux modules, il est désormais possible de sélectionner différents types de générateurs combinant des familles différentes. Le guide d'utilisation de ces deux modules est donné en annexe.

Chapitre 2

Générateurs linéaires

Dans ce chapitre, nous allons présenter des générateurs dont la fonction de transition T, de même que la fonction de sortie G, sont linéaires. Nous allons considérer deux familles de générateurs très utilisées et documentées : les générateurs à récurrence linéaire multiple, souvent désignés par MRG (pour «multiple recursive generator»), et les générateurs LFSR (pour «linear feedback shift register»), aussi appelés générateurs de Tausworthe. Nous discuterons de la période de ces générateurs, de leur implantation, de la structure des nombres qu'ils produisent, et des mesures d'uniformité utilisées, en pratique, pour choisir les paramètres des récurrences.

2.1 MRG

2.1.1 Définition

Les générateurs à récurrence multiple, ou MRG, sont basés sur la récurrence linéaire suivante :

$$x_n = (a_1 x_{n-1} + a_2 x_{n-2} + ... + a_k x_{n-k}) \bmod m, \tag{2.1}$$

où l'ordre de la récurrence k et le module m sont des entiers positifs, et les coefficients $a_1, a_2, ..., a_k$ sont des entiers entre $-(m-1)$ et $m-1$. L'état à l'étape n est donné par le vecteur $\mathbf{s}_n = (x_{n-k+1}, x_{n-k+2}, ..., x_n)$. La valeur de x_n, comprise entre 0 et $m-1$, peut ensuite être divisée par m pour produire une sortie entre 0 et 1. La fonction de sortie est alors : $u_n = G(s_n) = x_n/m$. Lorsque l'ordre de la récurrence est $k = 1$, nous retrouvons le cas particulier du LCG présenté à la section 1.1.

La période maximale pouvant être atteinte avec ce générateur est $\rho = m^k - 1$. En effet, le vecteur $\mathbf{s}_n$ est composé de k valeurs situées entre 0 et $m-1$, ce qui laisse m^k possibilités. Nous devons cependant éviter de générer l'état $\mathbf{s}_n = (0, 0, ..., 0)$, puisqu'alors tous les états suivants seront $(0, 0, ..., 0)$, et seule la sortie 0 sera produite par le générateur. Il reste donc $m^k - 1$ états possibles, d'où la période maximale mentionnée. Les coefficients de la récurrence $a_1, a_2, ..., a_k$, ainsi que le module m, doivent être choisis de façon appropriée pour que cette période maximale soit atteinte. Le module m doit être premier et les conditions données dans [9, 16] doivent être respectées. Il est à noter que seulement deux coefficients non nuls, a_k et a_r, $1 \leq r < k$, sont nécessaires pour satisfaire ces conditions. Dans ce cas, la récurrence devient

$$x_n = (a_r x_{n-r} + a_k x_{n-k}) \bmod m, \tag{2.2}$$

et le calcul d'un nouvel état est simplifié.

Le module m est généralement choisi de telle sorte que les m valeurs possibles pour x_n puissent être représentées de façon exacte avec un entier, sur un ordinateur. Ceci impose une limite sur la période maximale pouvant être atteinte, avec un ordre de récurrence k fixé.

2.1.2 Implantation

Plusieurs approches ont été suggérées pour calculer correctement et rapidement le produit $ax \bmod m$, lorsque m est grand. Il n'est généralement pas possible de faire les calculs directement avec des entiers, puisque la valeur de x peut devenir grande, et le produit ax risque de dépasser le plus grand entier représentable sur la machine utilisée. Une première méthode, décrite dans [4, 16, 25], permet de calculer le produit $ax \bmod m$ en trois étapes simples, tout en gardant les résultats intermédiaires entiers, entre $-m$ et m. Cette méthode, désignée par factorisation approximative, est valide si $a > 0$ et $a(m \bmod a) < m$. Une deuxième méthode, décrite dans [18, 19], calcule directement le produit ax et la division nécessaire pour l'opération mod m avec une arithmétique en virgule flottante. Ceci est possible si l'entier ax peut toujours être représenté exactement en virgule flottante. Sur la plupart des ordinateurs contemporains, un nombre en virgule

flottante est représenté avec 64 bits, et les entiers de -2^{53} jusqu'à 2^{53} peuvent être représentés exactement. Dans ce cas, cette méthode fonctionne si $|ax| \leq |a|m < 2^{53}$. Finalement, une troisième méthode, décrite d'abord dans [39], puis critiquée dans [29] et finalement améliorée dans [32], considère le cas où a peut s'exprimer comme une somme de quelques puissances de 2. Le produit ax est alors obtenu en faisant quelques décalages de bits vers la gauche (équivalents à une multiplication par une puissance de 2), et quelques additions. Une petite astuce permet, dans ce cas, de calculer efficacement l'opération mod m.

Ces différentes techniques, qui exploitent la forme particulière de la fonction de transition T, permettent de passer très rapidement d'un état à l'autre, rendant les MRG très efficaces.

2.1.3 Structure de réseau

L'ensemble des nombres pouvant être générés par un MRG possède une structure particulière, que nous allons décrire dans cette section. Nous allons former des vecteurs de t valeurs successives produites par un MRG, et analyser la structure de l'ensemble de ces vecteurs pouvant être générés, avec tous les états initiaux $\mathbf{s}_0$ possibles.

Mais d'abord, donnons la définition d'un réseau de dimension t, où t est un entier positif.

Définition 2.1.1 [26] *Un réseau de dimension t est un ensemble de la forme*

$$L_t = \left\{ \mathbf{v} = \sum_{i=1}^{t} z_i \mathbf{v}_i \mid \textit{chaque } z_i \textit{ entier} \right\}, \tag{2.3}$$

où $B = \{\mathbf{v}_1, ..., \mathbf{v}_t\}$ *est une base de* $\mathbb{R}^t$.

Un réseau est donc formé de l'ensemble des vecteurs obtenus à partir de combinaisons linéaires entières des vecteurs de base $\{\mathbf{v}_1, ..., \mathbf{v}_t\}$. La base B est souvent appelée base du réseau L_t.

Considérons maintenant l'ensemble Ψ_t formé de tous les vecteurs de t valeurs suc-

cessives pouvant être produits par un MRG, à partir de tous les états initiaux possibles. Nous avons $\Psi_t = \{\mathbf{u}_{n,t} = (u_n, ..., u_{n+t-1}) \mid n \geq 0, \mathbf{s}_0 \in S\}$, qui est équivalent à $\Psi_t = \{\mathbf{u}_{0,t} = (u_0, ..., u_{t-1}) \mid \mathbf{s}_0 \in S\}$. Le théorème suivant indique la structure particulière que forment les vecteurs de Ψ_t.

Théorème 2.1.1 [9, 26] *Les vecteurs de Ψ_t correspondent à l'intersection d'un réseau L_t avec l'hypercube $[0,1)^t$.*

Une façon de construire une base du réseau L_t pour un MRG est donnée dans [26]. Elle procède comme suit. Soit $\mathbf{e}_{i,t}$ le $i$ième vecteur unitaire en dimension t, et k l'ordre de la récurrence du MRG. Définissons $\mathbf{w}_i = (x_{i,1}, ..., x_{i,t})$ comme étant un vecteur dont les k premières coordonnées sont les mêmes que celles de $\mathbf{e}_{i,t}$, et les $t-k$ dernières coordonnées sont complétées en appliquant la relation de récurrence (2.1) : $x_{i,j} = (a_1 x_{i,j-1} + ... + a_k x_{i,j-k}) \bmod m$, pour $k < j \leq t$. Alors, une base possible est donnée par

$$\mathbf{v}_i = \begin{cases} \mathbf{w}_i/m & \text{pour } 1 \leq i \leq k, \\ \mathbf{e}_{i,t} & \text{pour } k < i \leq t. \end{cases} \tag{2.4}$$

Ceci nous permet de formuler le lemme 2.1.1, qui sera très utile au chapitre 5, lorsque de la nouvelle théorie concernant certains types de générateurs combinés sera présentée.

Lemme 2.1.1 *Un réseau L_t de base B donnée par (2.4) contient $\mathbb{Z}^t$.*

Preuve : Il est clair, en se rapportant à la définition 2.1.1, que la somme de deux vecteurs d'un réseau appartient encore au réseau, de même que le produit d'un scalaire entier avec un des vecteurs du réseau. Il suffit donc de montrer que chaque vecteur unitaire $\mathbf{e}_{i,t}$, pour $1 \leq i \leq t$, fait partie du réseau L_t de base B, et nous aurons montré que L_t contient $\mathbb{Z}^t$. Or, pour $k < i \leq t$, $\mathbf{e}_{i,t}$ appartient clairement au réseau, puisqu'il fait partie de la base B. Pour $i \leq k$, il suffit de remarquer que $\mathbf{e}_{i,t} = \mathbf{w}_i - \mathbf{x}_{i,k+1}\mathbf{e}_{k+1,t} - ... - \mathbf{x}_{i,t}\mathbf{e}_{t,t}$, qui est équivalent à $\mathbf{e}_{i,t} = m\mathbf{v}_i - \mathbf{x}_{i,k+1}\mathbf{v}_{k+1} - ... - \mathbf{x}_{i,t}\mathbf{v}_t$, une combinaison linéaire entière des vecteurs de base du réseau. □

L'exemple suivant permet d'illustrer la structure des points Ψ_t générés par un MRG.

Exemple 2.1.1 *Considérons un exemple simple en dimension $t = 2$. Soit un MRG d'ordre $k = 1$ (il s'agit ici d'un LCG), dont le module $m = 97$, et le coefficient $a_1 = 14$. L'ensemble des couples (u_n, u_{n+1}) pouvant être générés, à partir de tous les états initiaux, est illustré à la figure 2.1. La structure de réseau est bien visible. Notons que les points sont tous situés sur des droites parallèles équidistantes. Cette structure particulière est aussi présente pour des dimensions $t \geq 2$. Avec ces choix de paramètres, la période du LCG est maximale, et vaut $\rho = m-1 = 96$. Tous les couples de Ψ_2 peuvent être générés, à l'exception du point (0,0), lorsqu'une valeur initiale différente de 0 est choisie.*

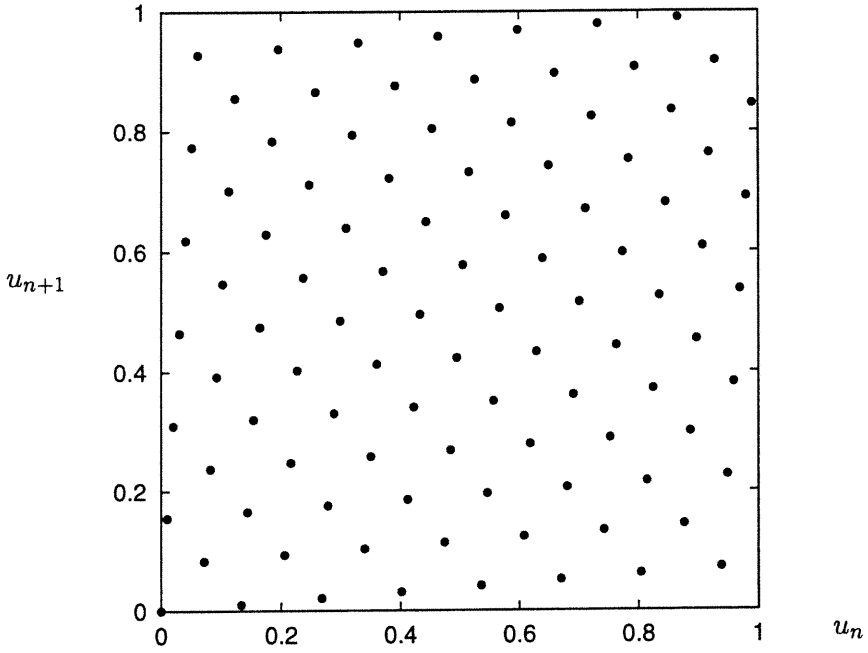


Figure 2.1: Points de Ψ_2 pour un LCG de paramètres $a_1 = 14$ et $m = 97$

2.1.4 Test spectral

Nous venons de voir que l'ensemble Ψ_t des vecteurs de t valeurs successives pouvant être produits par un MRG correspond à l'intersection d'un réseau L_t avec l'hypercube $[0,1)^t$. Ceci implique que les points de Ψ_t sont répartis sur un certain nombre d'hyperplans parallèles équidistants. En fait, il existe plusieurs familles de tels hyperplans

pouvant couvrir les points de Ψ_t. Choisissons celle maximisant la distance entre les hyperplans successifs, et notons par d_t la distance entre ces hyperplans. Une grande distance d_t signifie que les valeurs générées sont concentrées sur un nombre réduit d'hyperplans. Dans ce cas, de grandes portions de l'hypercube $[0,1)^t$ ne sont pas couvertes, ce qui n'est pas souhaitable. Nous cherchons plutôt des générateurs pour lesquels la distance d_t entre les hyperplans est petite, afin que les valeurs générées soient réparties le plus uniformémement possible sur l'hypercube $[0,1)^t$. Le calcul de cette distance d_t est appelé «test spectral».

Des bornes inférieures sur la distance d_t sont connues. Entre autres, nous avons la borne suivante (tirée de [16]), pour $t > k$:

$$d_t \geq \left(1 + \sum_{i=1}^{k} a_i^2\right)^{-1/2}. \tag{2.5}$$

Pour obtenir une petite distance d_t pour des dimensions $t > k$, les coefficients des MRG doivent donc être grands. Or, nous avons vu une forme simplifiée de la récurrence (2.1) qui n'utilisait que deux coefficients non nuls. Cette forme simplifiée réduit le nombre d'opérations à effectuer pour passer d'un état à un autre, mais augmente la valeur de la borne inférieure. Nous verrons cependant au chapitre 4 qu'en combinant plusieurs MRG ayant des récurrences simples, nous pouvons obtenir un générateur avec de bonnes propriétés au test spectral.

D'autres bornes inférieures sont données dans [9] pour des dimensions $t \leq 8$, et dans [19] pour des dimensions $t > 8$. Notons par d_t^* ces bornes inférieures, qui dépendent de la dimension t et de l'ordre de la récurrence k. Nous pouvons normaliser le résultat du test spectral, en calculant la quantité suivante :

$$S_t = \frac{d_t^*}{d_t}. \tag{2.6}$$

Le résultat normalisé S_t se situe entre 0 et 1, et une valeur près de 1 est recherchée. Cette normalisation permet de comparer les résultats obtenus dans différentes dimensions t. La valeur de M_τ suivante peut alors être calculée, permettant d'évaluer la qualité de la structure de réseau des points générés pour les dimensions $t \leq \tau$, où τ est une constante

choisie arbitrairement :

$$M_\tau = \min_{t \leq \tau} S_t. \tag{2.7}$$

2.2 LFSR

2.2.1 Définition

Les nombres produits par un MRG sont tous des multiples de $1/m$, où m est le module de la récurrence. Dans certains cas, une discrétisation plus fine est désirée pour les sorties produites. Une solution possible est de combiner L valeurs de x_n obéissant à la récurrence (2.1) pour produire une sortie u_n comme suit :

$$u_n = \sum_{i=1}^{L} x_{ns+i-1} m^{-i}, \tag{2.8}$$

où s et L sont des entiers positifs. Les sorties u_n deviennent alors des multiples de $1/m^L$, plutôt que de $1/m$. Une telle séquence de nombres peut avoir la même période ρ que le MRG de récurrence (2.1), à condition que ρ et s n'aient aucun facteur commun [16].

Les générateurs LFSR, aussi appelés générateurs de Tausworthe, correspondent au cas où $m = 2$. Les x_n obtenus par la récurrence (2.1) valent alors 0 ou 1, et la sortie est obtenue en prenant L bits successifs générés. La période maximale pouvant être atteinte dans ce cas est de $\rho = 2^k - 1$. Il faut choisir l'ordre de la récurrence k assez grand pour que la période obtenue soit longue.

2.2.2 Implantation

Lorsque la forme simplifiée (2.2) de la récurrence est utilisée, il est possible de calculer rapidement les valeurs $(x_{ns}, ..., x_{ns+L-1})$ nécessaires pour produire une sortie, à partir des valeurs $(x_{(n-1)s}, ..., x_{(n-1)s+L-1})$ précédentes. L'algorithme «QuickTaus», décrit dans [15, 21], se sert uniquement de décalages de bits, et des opérateurs binaires

ou-exclusif et «et» pour produire une nouvelle sortie. Certaines conditions sur s, k, r (voir équation (2.2)) et L, énoncées dans [15], doivent cependant être respectées. Entre autres, L doit correspondre au nombre de bits utilisés pour représenter un entier sur la machine considérée.

2.2.3 Équidistribution

La mesure d'uniformité utilisée pour juger de la qualité d'un générateur LFSR est décrite dans ce qui suit.

D'abord, supposons que l'hypercube $[0,1)^t$ est divisé en 2^{tl} cases de même taille. Plaçons dans ces cases les points de l'ensemble Ψ_t, contenant tous les vecteurs de t valeurs successives pouvant être produits par un générateur. Si les points produits sont bien répartis dans l'espace $[0,1)^t$, les 2^{tl} cases devraient contenir le même nombre de points. Comme l'ensemble Ψ_t contient 2^k points (dont possiblement des points qui se répètent), chacune des 2^{tl} cases devrait contenir 2^{k-tl} points. Cette remarque a donné lieu aux définitions suivantes.

Définition 2.2.1 [21] *L'ensemble Ψ_t est dit (t,l)-équidistribué si chacune des 2^{tl} cases contient le même nombre de points de Ψ_t.*

Définition 2.2.2 [15] *La résolution en dimension t, notée l_t, est la plus grande valeur de l pour laquelle l'ensemble Ψ_t est (t,l)-équidistribué.*

Pour que l'ensemble Ψ_t soit (t,l)-équidistribué, chaque case considérée doit contenir au moins un point. Comme chaque case doit aussi contenir 2^{k-tl} points, nous devons avoir $l \leq k/t$. La résolution l_t ne peut donc pas dépasser $\lfloor k/t \rfloor$. Aussi, la résolution l_t est bornée supérieurement par le nombre de bits L utilisé pour produire chaque nombre généré. Nous avons donc $l_t \leq l_t^*$, où $l_t^* = \min(L, \lfloor k/t \rfloor)$ désigne la résolution maximale possible en dimension t. De même, si une dimension $t > k$ est considérée, peu importe la valeur de l, il y aura plus de cases que de points, et certaines cases seront vides. La propriété de (t,l)-équidistribution n'est donc pas possible si $t > k$.

Définition 2.2.3 [15] *Un générateur est dit maximalement équidistribué (ME) si l_t atteint sa plus grande valeur possible dans toutes les dimensions, soit $l_t = l_t^*$, pour $t = 1, ..., k$.*

Une condition supplémentaire peut aussi être demandée, pour les valeurs de l dépassant $\lfloor k/t \rfloor$, ou pour des valeurs de $t > k$, comme l'indique la définition suivante.

Définition 2.2.4 [15] *Un générateur ME est dit sans-collision (CF, pour «collision-free») si pour $l_t < l \leq L$ et pour tout t, aucune des 2^{tl} cases ne contient plus d'un point.*

Une façon de déterminer si un générateur LFSR particulier est ME ou CF est expliquée dans [15]. Il est à noter qu'il est difficile d'obtenir des LFSR qui sont ME, lorsque la forme simplifiée (2.2) de la récurrence est utilisée. Par contre, nous verrons au chapitre 4 que ce problème peut être résolu en combinant plusieurs LFSR ayant des récurrences simples.

2.3 Autres générateurs linéaires

D'autres types de générateurs linéaires sont décrits dans [13, 16, 18]. Entre autres, une forme généralisée de LFSR est présentée. Ces générateurs, appelés GFSR pour «generalized feedback shift register», utilisent L copies de (2.1) avec $m = 2$ en parallèle pour générer les L bits de la sortie à produire. Une modification de ces GFSR, qui a donné naissance aux «twisted GFSR», a été proposée dans [34], et permet d'augmenter la période, pour un même ordre de récurrence k, par rapport aux LFSR et aux GFSR. Cependant, certains défauts dans la structure des nombres produits par ces «twisted GFSR» ont été mis en évidence dans [35]. Une transformation appliquée sur les valeurs de x_n, appelée «tempering», a été suggérée dans [35] pour remédier à ce problème de structure. Les articles [28, 35] donnent plus de détails sur ce dernier type de générateur, et fournissent des paramètres de générateurs ME.

2.4 Caractéristiques des générateurs linéaires

Les générateurs linéaires présentés dans ce chapitre ont plusieurs caractéristiques communes. D'abord, des conditions sur les coefficients $a_1, ..., a_k$ et le module m de la récurrence (2.1) permettent de vérifier rapidement si la période maximale est atteinte. Aussi, des mesures d'uniformité, comme le test spectral pour les MRG ou l'équidistribution pour les LFSR, permettent de choisir les paramètres de la récurrence (2.1) en s'assurant que les valeurs générées couvrent bien l'hypercube $[0,1]^t$, pour différentes dimensions t. Finalement, des algorithmes efficaces ont été développés pour générer la suite des états en exploitant la forme particulière de la récurrence (2.1). Ces différentes caractéristiques ont contribué à la grande popularité des générateurs linéaires, particulièrement dans des domaines où la rapidité de génération est un critère important.

Il faut cependant se rappeler que l'ensemble Ψ_t des vecteurs de t valeurs successives pouvant être produits par ces générateurs linéaires a une structure très régulière. Les points générés par un MRG sont tous situés sur un certain nombre d'hyperplans parallèles équidistants, alors que les points produits par un LFSR choisi ME, sur l'ensemble de sa période, sont trop uniformémement distribués par rapport à une vraie séquence de nombres aléatoires. Lorsqu'une petite fraction de ces points est utilisée, Ψ_t peut être vu comme un espace d'échantillonnage duquel sont tirées aléatoirement et sans remplacement les valeurs produites. Dans ce cas, la très grande uniformité des points de Ψ_t est justifiée. Il faut cependant être prudent, et éviter d'utiliser une trop grande partie de la période.

Une question surgit alors : quelle fraction de la période peut être utilisée de façon sécuritaire, sans que les résultats d'une application ne risquent d'être affectés par la structure des points générés ? Évidemment, la réponse dépend du type d'application considéré. Cependant, plusieurs tests statistiques simples ont été appliqués aux séquences produites par différents générateurs linéaires, et certains arrivent à détecter une différence entre les nombres produits et une vraie séquence aléatoire avant même que la racine carrée de la période ne soit utilisée [30]. Dans le cas des MRG, il serait

même possible de constuire un test plus complexe, à partir d'un algorithme décrit dans [2, 3], pouvant échouer dans l'ordre du logarithme de la période. Il est donc conseillé, en pratique, de ne générer qu'une petite fraction de la période, pour être certain de ne pas fausser les résultats calculés.

Chapitre 3

Générateurs non linéaires

Pour éliminer la structure très régulière des nombres produits par les générateurs linéaires décrits au chapitre précédent, de la non-linéarité peut être introduite dans la fonction de transition T, ou dans la fonction de sortie G. Dans ce chapitre, nous allons présenter trois familles de générateurs non linéaires assez documentés : les générateurs à congruence quadratique et cubique (dont les noms sont tirés de la forme de la fonction de transition T), et les générateurs inversifs, qui «inversent» la valeur obtenue avant de produire une sortie. Nous discuterons de la période de ces générateurs, de leur implantation et de la structure des nombres qu'ils produisent.

3.1 Générateurs à congruence quadratique

3.1.1 Définition

Les générateurs à congruence quadratique utilisent la fonction de transition non linéaire suivante pour passer d'un état à un autre :

$$x_n = (ax_{n-1}^2 + bx_{n-1} + c) \bmod m, \tag{3.1}$$

où m est un entier positif, et les coefficients a, b et c sont des entiers entre $-(m-1)$ et $m-1$. L'état du générateur correspond à la valeur de x_n, qui est comprise entre 0 et $m-1$. La sortie est obtenue en divisant le résultat par m, tout comme pour les MRG : $u_n = x_n/m$.

La période maximale pouvant être atteinte avec ce générateur est m. Cette période maximale est obtenue si et seulement si certaines conditions sur les coefficients a, b, c et

le module m, données dans [8, 9], sont respectées. Notons, entre autres, que a doit être un multiple de p, pour tous les entiers premiers impairs p divisant m. Contrairement aux MRG, m ne doit pas être choisi premier ici. En effet, si m est premier, pour avoir une période maximale, a doit être un multiple de m, et comme $-(m-1) \leq a \leq m-1$, la seule valeur possible pour a est 0. Dans ce cas, le générateur n'a plus une fonction de transition T quadratique, et nous retrouvons un générateur linéaire de type LCG juste un peu plus général que celui présenté aux sections 1.1 et 2.1.

Le module m est souvent choisi comme étant une puissance de 2. La période maximale peut alors être atteinte, et une implantation relativement rapide est disponible. Cependant, les vecteurs de deux ou trois valeurs successives pouvant être produits sont concentrés sur un petit nombre de droites parallèles [8], ce qui constitue un défaut de structure important. Il est donc préférable de choisir un module m composé, dont certains facteurs sont différents de 2.

Mentionnons finalement une variante des générateurs quadratiques décrits, souvent utilisée en cryptographie et connue sous le nom de générateurs «BBS». Une récurrence simplifiée $x_n = x_{n-1}^2 \bmod m$ est utilisée, avec un module $m = pq$ composé de deux nombres premiers distincts satisfaisant une condition supplémentaire donnée dans [1, 13]. En supposant que m est difficilement factorisable et suffisamment grand, il a été montré que ces générateurs produisent des sorties très difficiles à distinguer d'une vraie séquence aléatoire [1, 13]. Ces générateurs sont aussi discutés dans [9].

3.1.2 Implantation

Lorsque le module m est une puissance de 2, l'opération modulo m peut se faire rapidement. Si x est un entier représenté sur f bits, et que l'opération $x \bmod 2^e$ est à effectuer, avec $e \leq f$, il suffit de conserver les $f-e$ derniers bits de x pour retrouver le reste de la division par 2^e. Ceci peut être fait en appliquant l'opérateur logique «et» entre la représentation binaire de x et un masque dont les $f-e$ bits les moins significatifs sont mis à 1. Si e correspond au nombre de bits utilisés pour représenter un entier, et que les débordements ne sont pas vérifiés, l'opération modulo m peut

simplement être éliminée, ce qui donne une implantation encore plus rapide. Cependant, comme nous l'avons mentionné dans la section précédente, il n'est pas conseillé d'utiliser des modules qui sont des puissances de 2, à cause de certains défauts de structure dans les points qui sont alors générés.

Lorsque m n'est pas une puissance de 2, il n'existe pas d'algorithme particulier permettant de générer rapidement un nouvel état x_n. Le produit $ax^2 \bmod m$ peut être décomposé en $(x \times (ax \bmod m)) \bmod m$, et l'arithmétique en virgule flottante permet d'effectuer les calculs correctement, sur un ordinateur utilisant 64 bits pour représenter un nombre en virgule flottante, lorsque $|x \times (ax \bmod m)| \leq m^2 < 2^{53}$. Cette implantation, quoique relativement rapide, impose cependant une limite considérable sur la période m pouvant être atteinte. Aussi, des opérations supplémentaires (multiplications et modulo) doivent être effectuées, par rapport à l'implantation équivalente donnée au chapitre précédent pour les MRG, ce qui rend ces générateurs plus lents que les générateurs linéaires présentés. Des exemples de temps d'exécution permettant de comparer plusieurs générateurs sont donnés dans [22].

3.2 Générateurs à congruence cubique

3.2.1 Définition

La fonction de transition des générateurs à congruence cubique ressemble à celle des générateurs à congruence quadratique, mais un terme cubique est ajouté :

$$x_n = (ax_{n-1}^3 + bx_{n-1}^2 + cx_{n-1} + d) \bmod m, \tag{3.2}$$

où le module m est un entier positif, et les coefficients a, b, c et d sont des entiers entre $-(m-1)$ et $m-1$. Tout comme pour les générateurs à congruence quadratique, l'état du générateur correspond à la valeur de x_n, et la sortie est obtenue est divisant cet état par m.

La période maximale de ce générateur est aussi de m. Cependant, les coefficients et le module de la récurrence permettant d'atteindre cette période maximale sont difficiles à

trouver, puisqu'il n'existe pas de conditions simples garantissant une période maximale.

Certaines valeurs de m et de coefficients ont été proposées dans [7, 27], dans le cas où la récurrence (3.2) est réduite à :

$$x_n = (ax_{n-1}^3 + 1) \bmod m, \tag{3.3}$$

où m est un nombre premier. La période a été calculée en générant des nombres jusqu'à ce qu'un état soit rencontré de nouveau. Cette technique est très coûteuse, puisque les paires (a, m) formant un générateur de période maximale sont rares, et que plusieurs nombres doivent être générés pour chaque valeur essayée.

3.2.2 Implantation

Lorsque certains coefficients sont mis à 0, comme c'est le cas pour la récurrence simplifiée (3.3), le calcul d'un nouvel état peut être accéléré. Cependant, tout comme pour les générateurs à congruence quadratique, il n'y a pas d'algorithme efficace connu permettant d'exploiter la forme particulière de la récurrence pour générer plus rapidement de nouveaux nombres. La décomposition du produit $ax^3 \bmod m$ en $x \times (x \times (ax \bmod m) \bmod m) \bmod m$, et l'arithmétique en virgule flottante permettent toutefois d'obtenir une implantation correcte et relativement rapide, à condition que $|x \times (ax \bmod m)| \leq m^2$ puisse être représenté exactement sur un nombre en virgule flottante.

3.3 Générateurs inversifs

3.3.1 Définition

Il existe une famille de générateurs inversifs qui utilise la même fonction de transition T linéaire que les MRG, mais transforment l'état $\mathbf{s}_{n+1} = (x_{n-k+2}, ..., x_{n+1})$ différemment pour obtenir une sortie. Soit $\{x_n\}$ la séquence obtenue par la récurrence (2.1) avec un module m premier, et $\tilde{x}_n$ la $n$ième valeur non nulle de $\{x_n\}$. L'étape

intermédiaire non linéaire suivante est d'abord effectuée :

$$z_n = (\tilde{x}_{n+1}\tilde{x}_n^{-1}) \bmod m, \tag{3.4}$$

où $\tilde{x}_n^{-1}$ est l'inverse modulo m de $\tilde{x}_n$ (l'inverse de $\tilde{x}_n \neq 0$ modulo un nombre premier m est l'unique entier $\tilde{x}_n^{-1} \in \{1, ..., m-1\}$ tel que $\tilde{x}_n^{-1}\tilde{x}_n \bmod m = 1$). La sortie est ensuite obtenue en divisant le résultat par m : $u_n = z_n/m$. Il est à noter que dans le cas où l'ordre de la récurrence k est de 2 ou 3, la valeur de z_n peut être calculée directement à partir de sa valeur précédente. Les détails sont donnés dans [13, 16].

La période maximale d'un tel générateur est de m^{k-1}. Des conditions suffisantes sur les paramètres de la récurrence (2.1) permettent de trouver facilement des générateurs de période maximale [13].

Un autre type de générateur inversif plus simple se nomme «générateur inversif explicite». La récurrence T est simplement $x_n = (an + c) \bmod m$ pour $n \geq 0$, où $a \neq 0$ et c sont des entiers positifs inférieurs à m, un nombre premier. La sortie est ensuite obtenue en faisant $z_n = x_n^{-1}$ et $u_n = z_n/m$, où x_n^{-1} est encore l'inverse modulo m de x_n (sauf lorsque x_n est un multiple de m, où l'inverse est remplacé par la valeur 0). La période de ce générateur est toujours de m, peu importe le choix des paramètres [8, 13].

3.3.2 Implantation

L'inverse modulo m de x_n peut être calculé par l'algorithme d'Euclide [9], ou directement par la formule $x_n^{-1} = x_n^{m-2} \bmod m$ [13]. Dans les deux cas, le temps requis pour effectuer le calcul est dans $O(\log m)$ [13], ce qui est relativement lent lorsque m est grand. En général, ces générateurs sont plus lents que les générateurs à congruence quadratique et cubique décrits dans les sections précédentes.

3.4 Structure des nombres générés

La structure des nombres produits par les générateurs non linéaires décrits dans ce chapitre est beaucoup moins régulière que celle des générateurs linéaires du chapitre

précédent, même si une grande portion de la période est considérée. Plusieurs résultats théoriques ont été publiés, et certains d'entre eux sont discutés dans ce qui suit.

D'abord, des bornes inférieures et supérieures sur une mesure d'uniformité nommée «discrépance» ont été dérivées [7, 8]. Considérons l'ensemble Ψ_t de tous les vecteurs de t valeurs successives pouvant être produits par un générateur, et supposons que cet ensemble contient N points. Si $\mathcal{R}$ désigne l'ensemble des boîtes rectangulaires R de dimension t incluses dans $[0,1]^t$, $V(R)$ le volume de la boîte R, et $I(R)$ le nombre de points de Ψ_t tombant dans R, la discrépance peut être définie comme :

$$D_N^{(t)} = \max_{R \in \mathcal{R}} \left| V(R) - \frac{I(R)}{N} \right|. \tag{3.5}$$

Il s'agit donc de calculer, sur l'ensemble des boîtes rectangulaires R possibles, la différence entre le volume de la boîte et la fraction du nombre de points tombant dans cette boîte. Parfois, seules les boîtes ayant un coin à l'origine sont considérées, et la discrépance est alors notée $D_N^{*(t)}$. Plus la discrépance est faible, plus les points sont uniformément répartis sur l'hypercube $[0,1]^t$ (selon ce critère).

En pratique, lorsque la période d'un générateur est grande, il n'existe pas d'algorithme efficace permettant de calculer les valeurs exactes de la discrépance des points de Ψ_t produits. Par contre, des bornes inférieures et supérieures sur cette discrépance (ou sur une valeur moyenne de discrépance pour un ensemble de paramètres donnés) ont été dérivées dans [7] pour les générateurs à congruence cubique, et dans [8] pour les générateurs inversifs. Ces bornes montrent que les points de Ψ_t produits par ces générateurs ont une discrépance asymptotiquement (lorsque le nombre N de points tend vers l'infini) semblable à celle d'une vraie séquence aléatoire i.i.d. U(0,1), et ce pour toutes les dimensions $t \geq 2$.

Des bornes sur la discrépance ont aussi été obtenues pour les générateurs à congruence quadratique dans [8], pour des modules m qui sont des puissances de 2. Seules les dimensions 2 et 3 ont été analysées, et certaines bornes inférieures sur des valeurs moyenne de discrépance trouvées sont plus grandes que celles espérées. En fait, pour certains choix de paramètres, les points de Ψ_2 et de Ψ_3 sont concentrés sur un nombre restreint de droites parallèles, ce qui constitue un défaut de structure important. Dans

le cas plus général des modules m composés, la structure des points est connue, et correspond à une superposition d'un certain nombre de réseaux translatés [8]. Une forte structure linéaire est encore présente, rendant ces générateurs moins attrayants.

D'autres résultats concernant la structure des points produits ont été trouvés. Entre autres, les générateurs inversifs ont une propriété remarquable : aucun hyperplan de dimension $t \geq 2$ contient plus de t vecteurs de Ψ_t [8]. La structure des points produits, dans ce cas, est loin d'être linéaire. Cependant, des structures hyperboliques peuvent parfois être observées, lorsque les points de Ψ_2 sont dessinés. La figure 3.1, tirée de [8], donne un exemple de structure hyperbolique visible, lorsqu'un générateur inversif basé sur une récurrence d'ordre 2 de coefficients $a_1 = 212\ 709\ 106$ et $a_2 = 67\ 483\ 458$ et de module $m = 2^{31} - 1$ est utilisé.

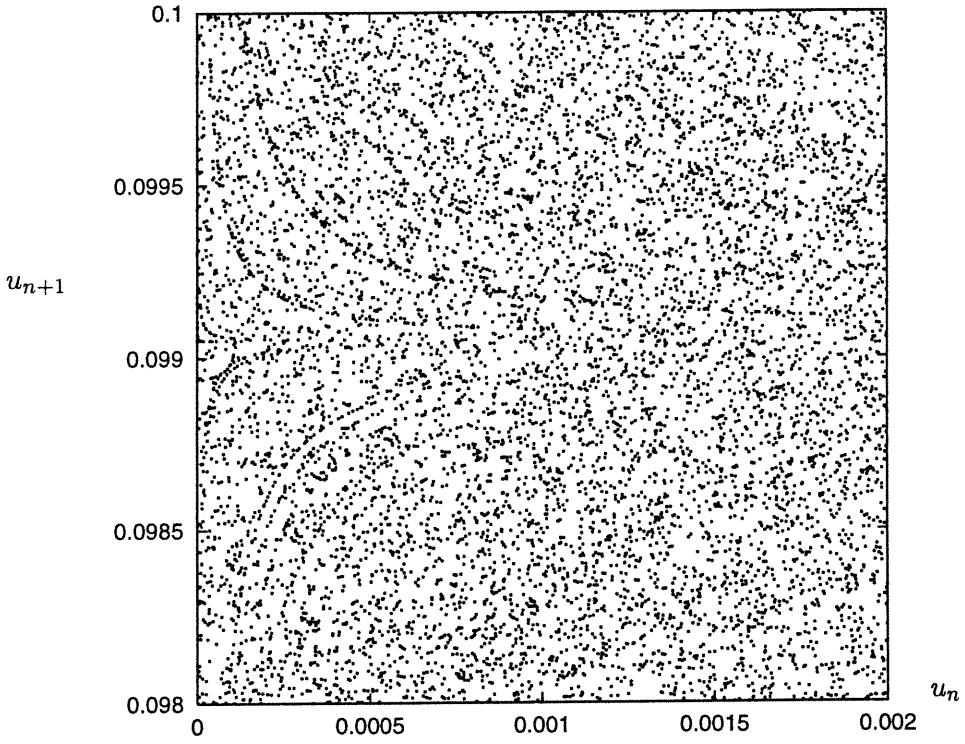


Figure 3.1: Points de Ψ_2 pour un générateur inversif basé sur une récurrence d'ordre 2 de paramètres $a_1 = 212\ 709\ 106$, $a_2 = 67\ 483\ 458$ et $m = 2^{31} - 1$

3.5 Caractéristiques des générateurs non linéaires

Les générateurs non linéaires présentés dans ce chapitre, à l'exception des générateurs à congruence quadratique, ont l'avantage de produire des points sans structure trop régulière, même si les nombres générés sur la période entière sont considérés. De plus, des bornes indiquent que la discrépance des nombres produits par ces générateurs est très semblable à celle d'une vraie séquence de nombres aléatoires i.i.d. U(0,1), lorsque le nombre N de points de Ψ_t est grand. Il semble donc que les points produits ont de bonnes propriétés d'uniformité. Il ne faut cependant pas oublier que ces résultats sont asymptotiques, et que certains d'entre eux sont basés sur une valeur moyenne de discrépance, pour un ensemble de paramètres donné. Ils ne nous permettent pas de savoir comment se comporte un générateur en particulier. Mais les tests statistiques appliqués jusqu'à maintenant à différents générateurs à congruence cubique ou inversifs ont montré qu'une grande partie de la période peut être utilisée sans problème. Certains tests ne commencent même à échouer que lorsque le nombre de points générés dépasse la période entière [27, 30, 31].

Les points produits par les générateurs à congruence cubique ou inversifs semblent donc avoir de bonnes propriétés. Cependant, ces générateurs sont relativement lents. Il n'y a pas d'algorithme rapide connu, permettant d'exploiter la forme particulière des récurrences utilisées. Dans le cas des générateurs à récurrence cubique, des implantations plus rapides sont disponibles, mais le module m doit rester relativement petit, ce qui impose des contraintes sur la période maximale pouvant être atteinte.

La période des générateurs cubiques est aussi limitée par le temps de calcul élevé que requiert la recherche de bons paramètres. Nous avons vu qu'il n'existe pas de conditions simples connues permettant de vérifier si un générateur particulier a une période maximale. Il faut donc générer tous les nombres de la période pour connaître celle-ci, ce qui est très coûteux en temps, particulièrement lorsqu'une longue période est recherchée. Le coût de la recherche est d'autant plus grand que les paramètres permettant d'atteindre une période maximale sont rares. Certains paramètres ont été proposés dans [7, 27], mais les générateurs sont tous trop petits pour être utilisés

individuellement (la plus grande période est près de 2^{17}).

Une solution possible, proposée dans [7, 27], est de combiner les sorties de plusieurs petits générateurs cubiques. Ceci permet au générateur «combiné» d'avoir une assez longue période, même si la période de chaque composante individuelle est petite. Aussi, chaque composante peut être implantée plus efficacement, et le générateur résultant peut être assez rapide. Ces générateurs combinés seront présentés dans le prochain chapitre.

Chapitre 4

Combinaison de générateurs d'une même famille

Dans les chapitres précédents, nous avons présenté plusieurs familles de générateurs linéaires et non linéaires. Dans certains cas, nous avons vu qu'un dilemme se posait lors du choix des paramètres des récurrences : privilégier une bonne structure des points à générer, ou privilégier une implantation plus rapide du générateur. Par exemple, la somme des carrés des coefficients $a_1, a_2, ..., a_k$ de la récurrence des MRG doit être grande pour que les points soient répartis sur des hyperplans rapprochés. Par contre, plus il y a de coefficients nuls, plus l'implantation peut être rapide. De même, il est difficile de trouver des générateurs LFSR qui ont de bonnes propriétés d'équidistribution, lorsque plusieurs coefficients de la récurrence de base sont nuls. Dans le cas des générateurs cubiques, le module m de la récurrence doit être grand pour que la période soit assez longue. Cependant, il n'y a pas d'implantation efficace connue lorsque m est trop grand.

Une façon de concilier la rapidité d'exécution avec la qualité des points produits est de combiner la sortie de plusieurs générateurs. Certains articles comme [5, 6, 33] s'appuient sur des résultats théoriques pour conclure que les propriétés statistiques des générateurs combinés sont meilleures que les propriétés de chacune des composantes individuelles. Il faut toutefois se méfier des conclusions tirées par ces auteurs, puisque les théorèmes sur lesquels ils s'appuient supposent que les nombres qui sont combinés sont des variables aléatoires indépendantes et tirées aléatoirement d'une distribution presque uniforme. Ces théorèmes ne s'appliquent pas aux séquences déterministes produites par les générateurs, comme l'explique l'article [11]. Il faut plutôt être prudent dans le choix des générateurs à combiner, et dans la façon de les combiner, puisqu'il n'y a pas nécessairement de garantie sur la qualité des points produits. Cependant, lorsque les générateurs sont combinés avec soin, d'excellents résultats peuvent être obtenus [12,

27, 33].

Dans ce chapitre, nous allons discuter de certains générateurs combinés dont les composantes individuelles sont issues de la même famille. Trois types de générateurs pour lesquels des résultats théoriques intéressants ont été trouvés seront présentés, soient les MRG combinés, les LFSR combinés et les générateurs à congruence cubique combinés.

4.1 MRG combinés

Considérons J MRG évoluant en parallèle, et supposons que le $j$ième générateur suit la récurrence d'ordre k_j et de module m_j suivante :

$$x_{j,n} = (a_{j,1}x_{j,n-1} + ... + a_{j,k_j}x_{j,n-k_j}) \bmod m_j, \tag{4.1}$$

pour $1 \leq j \leq J$. Dans ce qui suit, nous allons supposer que les modules m_j sont relativement premiers entre eux, et que la période ρ_j de chaque MRG est maximale, c'est-à-dire que $\rho_j = m_j^{k_j} - 1$.

Deux façons de combiner les valeurs produites par ces générateurs ont été proposées dans [14]. D'abord, une combinaison linéaire des $x_{j,n}$ est faite, puis le résultat est ramené entre 0 et $m_1 - 1$ comme suit :

$$z_n = \left(\sum_{j=1}^{J} \delta_j x_{j,n} \right) \bmod m_1, \tag{4.2}$$

où chaque δ_j est un entier relativement premier à m_j. La sortie, que nous noterons par u_n, est obtenue en divisant la valeur de z_n par m_1.

Ensuite, une combinaison linéaire des J sorties des MRG est proposée, puis le résultat est ramené entre 0 et 1 en effectuant une opération modulo 1. La sortie du générateur combiné, que nous noterons par w_n, est la suivante :

$$w_n = \left(\sum_{j=1}^{J} \frac{\delta_j x_{j,n}}{m_j} \right) \bmod 1, \tag{4.3}$$

où chaque δ_j est encore un entier relativement premier à m_j. Les δ_j sont souvent mis à 1, auquel cas la sortie w_n correspond à la somme des J sorties des MRG modulo 1.

La période de ces générateurs combinés est toujours égale au plus petit commun multiple des périodes de chacune de ses composantes : $\rho = \text{PPCM}(\rho_1, ..., \rho_J)$. Il est donc possible d'atteindre de grandes périodes, même si chaque MRG individuel est relativement petit.

Aussi, il est montré dans [14] que le générateur combiné (4.3) est équivalent à un MRG d'ordre $k = \max(k_1, ..., k_J)$ et de module $m = \prod_{i=1}^{J} m_j$. Les w_n obéissent donc à la récurrence

$$x_n = (a_1 x_{n-1} + ... + a_k x_{n-k}) \bmod m, \quad w_n = x_n/m, \tag{4.4}$$

où les coefficients $a_1, ..., a_k$ sont connus, et ne dépendent pas des valeurs de δ_j. Quant à la combinaison (4.2), elle est presque équivalente à (4.3), puisque les sorties u_n peuvent être exprimées comme $u_n = (w_n + \epsilon_n) \bmod 1$, où ϵ_n est une quantité bornée inférieurement et supérieurement par des constantes généralement très petites lorsque les m_j sont rapprochés.

La récurrence (4.4) peut avoir plusieurs grands coefficients, même si les J MRG utilisés dans la combinaison ont seulement deux petits coefficients non nuls. Il est donc possible d'obtenir des MRG combinés très rapides, tout en ayant une bonne valeur au test spectral. Des exemples de bons générateurs combinés sont donnés dans [19], et une implantation rapide y est suggérée.

4.2 LFSR combinés

Considérons maintenant J LFSR évoluant en parallèle. Notons par k_j l'ordre de la récurrence du générateur j, et par $x_{j,n}$ la $n$ième valeur produite par cette récurrence. Les sorties du $j$ième générateur sont alors données par

$$u_{j,n} = \sum_{i=1}^{L} x_{j,ns_j+i-1} 2^{-i}. \tag{4.5}$$

Une façon de combiner les sorties des J générateurs, présentée dans [15], est de faire un ou-exclusif bit à bit entre chacune des sorties : $u_n = u_{1,n} \oplus ... \oplus u_{J,n}$. Si les s_j sont choisis de telle sorte que le plus grand commun diviseur entre s_j et $2^{k_j} - 1$ est 1, et qu'une condition supplémentaire donnée dans [15] sur les coefficients des récurrences est respectée, la période du générateur combiné est de $\rho = (2^{k_1} - 1) \times ... \times (2^{k_J} - 1)$, qui correspond au produit des périodes de chacune des composantes. Encore une fois, de grandes périodes peuvent être atteintes, même si chaque générateur individuel est relativement petit.

Le générateur combiné présenté est équivalent à un LFSR dont la récurrence de base et la valeur de s de la fonction de sortie peuvent être trouvées dans [15, 38]. La récurrence de ce LFSR équivalent peut avoir plusieurs coefficients non nuls, même si les récurrences des J composantes de la combinaison ont seulement deux coefficients non nuls. Il devient alors beaucoup plus facile de trouver des générateurs avec de bonnes propriétés d'équidistribution, tout en conservant une bonne vitesse d'exécution. Des exemples de bons LFSR combinés peuvent être trouvés dans [21], de même qu'une implantation rapide de ces générateurs.

4.3 Générateurs à congruence cubique combinés

Les générateurs à congruence cubique combinés ont été présentés dans [7]. Dans cet article, J générateurs cubiques de la forme (3.3) sont considérés. La sortie u_n du générateur combiné est obtenue en additionnant modulo 1 les sorties des J générateurs cubiques : $u_n = (u_{1,n} + ... + u_{J,n}) \bmod 1$, où $u_{j,n}$ représente la $n$ième sortie du $j$ième générateur.

Si les modules m_j des générateurs cubiques sont des nombres premiers distincts, et que chaque générateur a une période maximale $\rho_j = m_j$, le générateur combiné a une période de $\rho = \text{PPCM}(\rho_1, ..., \rho_J) = m_1 \times ... \times m_J$. Encore une fois, la combinaison permet d'obtenir un générateur de grande période, même si chacune de ses composantes demeure relativement petite.

Des bornes supérieures sur la discrépance des points produits par ces générateurs combinés sont obtenues dans [7]. En fait, les bornes dont il a été question au chapitre précédent pour les générateurs cubiques non combinés sont un cas particulier de ces bornes dérivées pour la combinaison de J générateurs. Rappelons que ces bornes indiquent que la discrépance des nombres produits par les générateurs cubiques (combinés ou non) est très semblable à celle d'une vraie séquence aléatoire i.i.d. U(0,1), lorsque la période de ces générateurs est très grande. La borne sur la discrépance obtenue est plus petite lorsque le nombre J de générateurs à combiner et la dimension t considérée sont petits. Ceci pourrait suggérer qu'un nombre restreint de générateurs devraient être combinés, si une bonne uniformité des points est recherchée. Il faut toutefois être prudent dans l'interprétation des résultats, puisque la seule indication sur la discrépance que nous avons est une borne supérieure.

Des générateurs combinés à deux composantes ont été testés empiriquement dans [27, 30], et les résultats sont très encourageants : en général, la période complète doit être générée avant que les tests appliqués ne commencent à détecter une différence entre les nombres produits et une vraie séquence de nombres aléatoires.

Chapitre 5

Combinaison de générateurs de familles différentes

Dans le chapitre précédent, nous avons vu plusieurs avantages à combiner des générateurs. Des périodes beaucoup plus longues peuvent être atteintes, tout en conservant une bonne vitesse de génération. Aussi, la structure des nombres produits peut s'améliorer. Jusqu'à maintenant, plusieurs combinaisons ont été suggérées, mais elles impliquent toutes des générateurs issus d'une même famille, comme pour les cas présentés au chapitre précédent.

Dans ce mémoire, nous nous sommes intéressés à la combinaison de générateurs issus de familles différentes. Plus spécifiquement, nous voulons étudier des générateurs combinés dont une des composantes est un générateur linéaire, et l'autre composante est non linéaire. Nous espérons ainsi améliorer la structure trop régulière des générateurs linéaires, tout en développant des algorithmes plus rapides que ceux des générateurs complètement non linéaires.

Comme il a été mentionné au chapitre précédent, il faut toutefois être prudent dans la façon combiner des générateurs, puisque rien ne garantit que la qualité des points produits sera meilleure. Nous avons donc développé, dans ce chapitre, de la théorie qui permet de mieux comprendre la structure des points produits par les générateurs qui nous intéressent, de façon à s'assurer que l'ensemble des vecteurs de t valeurs successives pouvant être générés couvre bien uniformément l'espace $[0,1]^t$.

La prochaine section présente des résultats généraux sur la structure des points obtenus en combinant les sorties $u_{1,n}$ et $u_{2,n}$ de deux générateurs quelconques. Deux types de combinaisons sont considérés, soit l'addition des sorties modulo 1 ($(u_{1,n} + u_{2,n}) \bmod 1$), et un ou-exclusif bit à bit entre les sorties ($u_{1,n} \oplus u_{2,n}$). La théorie plus

spécifique à deux types de générateurs que nous avons décidé d'étudier est ensuite présentée. D'abord, la sortie d'un MRG est additionnée, modulo 1, avec la sortie d'un autre générateur quelconque. Ensuite, des générateurs LFSR sont combinés avec un autre générateur quelconque, au moyen d'un ou-exclusif bit à bit. Évidemment, nous sommes intéressés au cas où le deuxième générateur quelconque est non linéaire.

Les combinaisons considérées sont du même type que lorsque les MRG ou les LFSR sont combinés avec un autre générateur de leur famille, principalement à cause des résultats théoriques intéressants pouvant être déduits dans ce cas. Notons que la théorie présentée s'applique aussi lorsque des LFSR ou des MRG combinés sont utilisés dans la combinaison, et qu'elle peut facilement être généralisée au cas où plusieurs générateurs quelconques sont aussi combinés.

5.1 Combinaison de deux générateurs quelconques

Soit $u_{1,n}$ la $n$ième sortie d'un générateur G_1 de période ρ_1, dont l'état initial est $s_{1,0} \in S_1$, et $u_{2,n}$ la $n$ième sortie d'un deuxième générateur G_2 de période ρ_2, dont l'état initial est $s_{2,0} \in S_2$.

Notons par $\Psi_{1,t} = \{\mathbf{u}_{1,0,t} = (u_{1,0}, u_{1,1}, ..., u_{1,t-1}) \mid s_{1,0} \in S_1\}$ l'ensemble des points de t valeurs successives pouvant être produits par le générateur G_1, pour tous les états initiaux $s_{1,0}$ possibles, et par $\Psi_{2,t} = \{\mathbf{u}_{2,0,t} = (u_{2,0}, u_{2,1}, ..., u_{2,t-1}) \mid s_{2,0} \in S_2\}$ l'ensemble des points de t valeurs successives produits par le générateur G_2, pour tous les états initiaux $s_{2,0}$ possibles.

Considérons un générateur combiné G, de période ρ, dont la n*ième* sortie est soit (5.1) ou (5.2) :

$$u_n = (u_{1,n} + u_{2,n}) \bmod 1, \tag{5.1}$$

$$u_n = (u_{1,n} \oplus u_{2,n}). \tag{5.2}$$

Notons par $\Psi_t = \{\mathbf{u}_{0,t} = (u_0, ..., u_{t-1}) \mid s_{1,0} \in S_1, s_{2,0} \in S_2\}$ l'ensemble des points de

t valeurs successives produits par le générateur combiné G, pour tous ses états initiaux possibles.

Nous aimerions connaître la structure des points formés par Ψ_t.

5.1.1 Combinaison avec une addition modulo 1

Si la combinaison (5.1) est utilisée, nous avons $\Psi_t = \{\mathbf{u}_{0,t} \mid s_{1,0} \in S_1, s_{2,0} \in S_2\} = \{(\mathbf{u}_{1,0,t} + \mathbf{u}_{2,0,t}) \bmod 1 \mid s_{1,0} \in S_1, s_{2,0} \in S_2\}$, où l'opération modulo s'applique à chaque coordonnée des points. Si $\ll + \gg$ dénote la somme directe de deux ensembles (l'ensemble obtenu en additionnant deux à deux chaque élément du premier ensemble avec chaque élément du second ensemble), et $(\Psi_{1,t} + \mathbf{u}_t) \bmod 1$ désigne l'ensemble des points de $\Psi_{1,t}$, auxquels nous avons ajouté le vecteur $\mathbf{u}_t$, puis réduit modulo 1 chaque coordonnée, nous pouvons réécrire Ψ_t comme suit :

$$\begin{aligned} \Psi_t &= \{(\mathbf{u}_{1,0,t} + \mathbf{u}_{2,0,t}) \bmod 1 \mid s_{1,0} \in S_1, s_{2,0} \in S_2\} \\ &= (\Psi_{1,t} \ll + \gg \Psi_{2,t}) \bmod 1 \\ &= \bigcup_{s_{2,0} \in S_2} (\Psi_{1,t} + \mathbf{u}_{2,0,t}) \bmod 1, \end{aligned}$$

d'où la proposition suivante.

Proposition 5.1.1 *Si la combinaison (5.1) est utilisée, alors les points de Ψ_t sont obtenus en superposant ρ_2 ensembles de points de la forme $(\Psi_{1,t} + \mathbf{u}_{2,0,t}) \bmod 1$.*

5.1.2 Combinaison avec un ou-exclusif bit à bit

Si la combinaison (5.2) est utilisée, nous avons plutôt $\Psi_t = \{\mathbf{u}_{0,t} \mid s_{1,0} \in S_1, s_{2,0} \in S_2\} = \{(\mathbf{u}_{1,0,t} \oplus \mathbf{u}_{2,0,t}) \mid s_{1,0} \in S_1, s_{2,0} \in S_2\}$, où le ou-exclusif est fait bit à bit sur chaque coordonnée des points. Si nous notons par $\ll \oplus \gg$ la somme directe de deux ensembles, où la somme est remplacée par un ou-exclusif bit à bit, et si nous désignons par $\Psi_{1,t} \oplus \mathbf{u}_t$ l'ensemble des points de $\Psi_{1,t}$ auxquels nous avons ajouté le vecteur $\mathbf{u}_t$ en

faisant un ou-exclusif bit à bit, nous obtenons

$$\begin{aligned}\Psi_t &= \{\mathbf{u}_{1,0,t} \oplus \mathbf{u}_{2,0,t} \mid s_{1,0} \in S_1, s_{2,0} \in S_2\} \\ &= (\Psi_{1,t} \ll \oplus \gg \Psi_{2,t}) \\ &= \bigcup_{s_{2,0} \in S_2} \Psi_{1,t} \oplus \mathbf{u}_{2,0,t},\end{aligned}$$

d'où la prochaine proposition.

Proposition 5.1.2 *Si la combinaison (5.2) est utilisée, alors les points de Ψ_t sont obtenus en superposant ρ_2 ensembles de points de la forme $\Psi_{1,t} \oplus \mathbf{u}_{2,0,t}$.*

Regardons maintenant certaines propriétés de l'ensemble Ψ_t, d'abord lorsque la combinaison (5.1) est utilisée avec un générateur G_1 de type MRG, puis lorsque la combinaison (5.2) est utilisée avec un générateur G_1 de type LFSR.

5.2 Combinaison d'un MRG avec un autre générateur

Nous supposons ici que la combinaison (5.1) est utilisée avec un générateur G_1 de type MRG, et un générateur G_2 quelconque.

Avant d'analyser la structure des points produits par ce MRG combiné, certaines propriétés de la structure de réseau des MRG doivent être énoncées. Voici donc une proposition et un lemme qui nous seront utiles par la suite.

Proposition 5.2.1 *Soit $\Psi_{1,t}$ l'ensemble formé de tous les vecteurs de t valeurs successives pouvant être produits par un MRG, et L_t le réseau tel que $\Psi_{1,t} = L_t \cap [0,1)^t$. Soit $\tilde{\Psi}_{1,t} = (\Psi_{1,t} + \mathbf{w}_t) \bmod 1$ l'ensemble des points de $\Psi_{1,t}$ translatés par le vecteur $\mathbf{w}_t = (w_1, w_2, ..., w_t)$, dont chaque coordonnée a été réduite modulo 1. Alors, $\tilde{\Psi}_{1,t} = (L_t + \mathbf{w}_t) \cap [0,1)^t$, où $(L_t + \mathbf{w}_t)$ correspond au réseau L_t translaté de $\mathbf{w}_t$.*

Preuve : D'abord, montrons que $\tilde{\Psi}_{1,t} \subseteq (L_t + \mathbf{w}_t) \cap [0,1)^t$. Soit $\mathbf{x}_t \in \tilde{\Psi}_{1,t} = (\Psi_{1,t} + \mathbf{w}_t) \bmod 1$. Par l'opération modulo, nous savons que $\mathbf{x}_t \in [0,1)^t$. De plus,

$$\begin{aligned}
\mathbf{x}_t &= (\boldsymbol{\nu}_t + \mathbf{w}_t) \bmod 1, \text{ pour un certain } \boldsymbol{\nu}_t \in \Psi_{1,t} \\
&= ((\nu_1 + w_1) \bmod 1, (\nu_2 + w_2) \bmod 1, ..., (\nu_t + w_t) \bmod 1) \\
&= (\nu_1 + w_1 - i_1, \nu_2 + w_2 - i_2, ..., \nu_t + w_t - i_t), \\
&\quad \text{où } i_1, i_2, ..., i_t \text{ sont entiers} \\
&= (\nu_1, \nu_2, ..., \nu_t) - (i_1, i_2, ..., i_t) + (w_1, w_2, ...w_t) \\
&= (\nu'_1, \nu'_2, ..., \nu'_t) + (w_1, w_2, ...w_t) \\
&\quad \text{où } \nu'_1, \nu'_2, ..., \nu'_t \in L_t \text{ par le lemme 2.1.1} \\
&\in L_t + \mathbf{w}_t.
\end{aligned}$$

Nous avons bien $\mathbf{x}_t \in (L_t + \mathbf{w}_t) \cap [0,1)^t$, d'où $\tilde{\Psi}_{1,t} \subseteq (L_t + \mathbf{w}_t) \cap [0,1)^t$. Il reste à montrer que $(L_t + \mathbf{w}_t) \cap [0,1)^t \subseteq \tilde{\Psi}_{1,t}$. Si $\mathbf{x}_t \in (L_t + \mathbf{w}_t) \cap [0,1)^t$, alors nous avons $\mathbf{x}_t = \boldsymbol{\nu}_t + \mathbf{w}_t$ pour un certain $\boldsymbol{\nu}_t \in L_t$, et chaque composante de $\mathbf{x}_t$ est entre 0 et 1. Nous pouvons faire un modulo 1 sur chaque coordonnée sans changer le résultat, d'où $\mathbf{x}_t = (\boldsymbol{\nu}_t + \mathbf{w}_t) \bmod 1$ pour un certain $\boldsymbol{\nu}_t \in L_t$. Nous avons bien $\mathbf{x}_t \in \tilde{\Psi}_{1,t}$. □

Exemple 5.2.1 *Regardons un exemple en deux dimensions. Prenons un LCG de paramètre* $a_1 = 7$ *et de module* $m = 17$. *Les points* (u_0, u_1) *de* $\Psi_{1,2}$ *produits par ce générateur sont illustrés à la figure 5.1. La structure de réseau est bien visible, et deux vecteurs de base possibles* $\mathbf{v}_1$ *et* $\mathbf{v}_2$ *ont été dessinés. La figure 5.2 montre ce qui se produit lorsque les points de* $\Psi_{1,2}$ *sont translatés par* $\mathbf{w}_2 = (0.3, 0.4)$. *Nous obtenons alors l'intersection du réseau initial translaté par* $(0.3, 0.4)$ *avec le carré* $[0.3, 1.3) \times [0.4, 1.4)$. *En faisant l'opération modulo* 1 *sur chaque coordonnée des points de* $\Psi_{1,2} + \mathbf{w}_2$, *les points débordant du carré unité sont ramenés dans les régions* $[0, 0.3) \times [0, 1)$ *et* $[0.3, 1) \times [0, 0.4)$ *pour compléter les points du réseau translaté* $L_2 + \mathbf{w}_2$ *situés dans* $[0,1)^2$, *comme illustré sur la figure 5.3.*

Lemme 5.2.1 *Translater un réseau* L_t *de base* $B = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_t)$ *par* $\mathbf{w}_t$ *est équivalent à le translater par* $\mathbf{w}_t + (x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + ... + x_t\mathbf{v}_t)$, *où* $x_1, x_2, ..., x_t$ *sont des entiers arbitraires.*

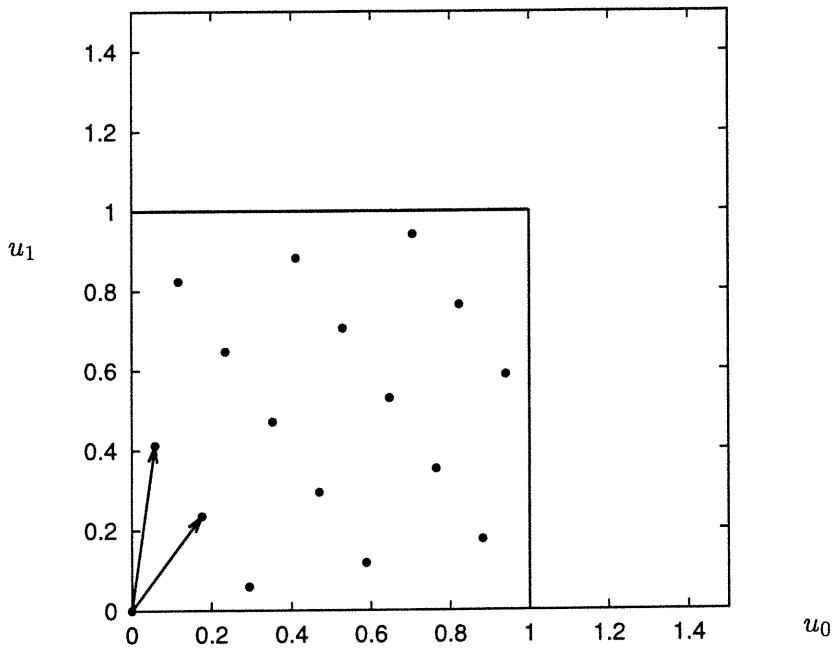


Figure 5.1: Points de $\Psi_{1,2}$ générés par un LCG avec $a_1 = 7$ et $m = 17$

Preuve : Il suffit de translater le réseau L_t d'abord par $(x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + ... + x_t\mathbf{v}_t)$, pour constater que le même réseau est obtenu. En effet, les points de L_t sont $\{y_1\mathbf{v}_1 + y_2\mathbf{v}_2 + ... + y_t\mathbf{v}_t \mid y_1, y_2, ..., y_t$ entiers$\}$, et les points de $L_t + (x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + ... + x_t\mathbf{v}_t)$ sont $\{y_1\mathbf{v}_1 + y_2\mathbf{v}_2 + ... + y_t\mathbf{v}_t + (x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + ... + x_t\mathbf{v}_t) \mid y_1, y_2, ..., y_t$ entiers$\}$ = $\{y_1'\mathbf{v}_1 + y_2'\mathbf{v}_2 + ... + y_t'\mathbf{v}_t \mid y_1', y_2', ..., y_t'$ entiers$\}$ puisque $x_1, x_2, ..., x_t$ sont des entiers. □

Nous avons maintenant les outils nécessaires pour analyser la structure des points de Ψ_t produits par un MRG combiné avec un autre générateur selon (5.1).

Par la proposition 5.1.1, nous savons que les points de Ψ_t sont obtenus en superposant ρ_2 ensembles de points de la forme $(\Psi_{1,t} + \mathbf{u}_{2,0,t})$ mod 1. Nous savons aussi, par le théorème 2.1.1, que l'ensemble des points $\Psi_{1,t}$ générés par le MRG peut être décrit par l'intersection d'un réseau de base B avec l'hypercube $[0,1)^t$. En translatant, modulo 1, l'ensemble de ces points, nous obtenons l'intersection d'un réseau translaté avec $[0,1)^t$ (proposition 5.2.1). Nous avons donc le résultat suivant :

Proposition 5.2.2 *Les points de Ψ_t sont obtenus en superposant les intersections de*

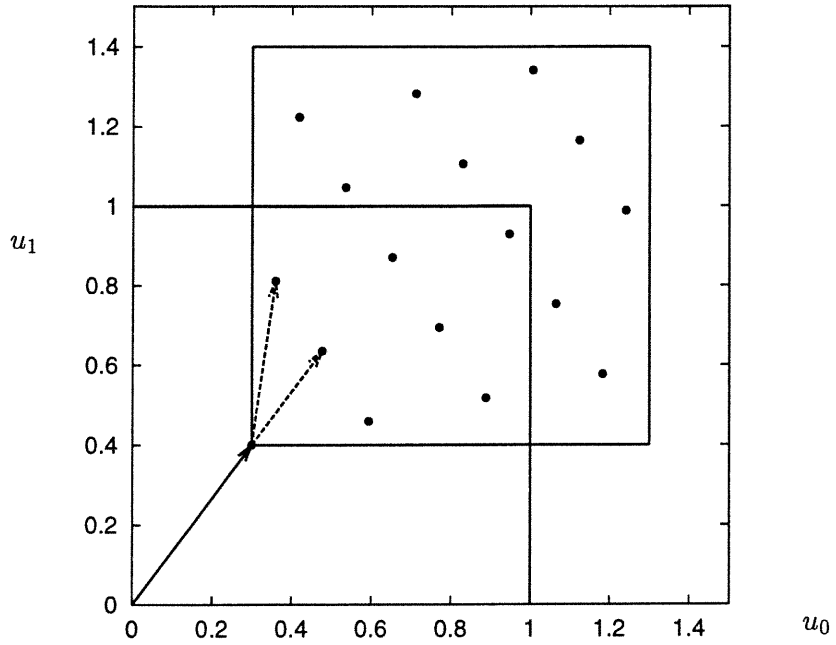


Figure 5.2: Points de $\Psi_{1,2}$ translatés par $(0.3, 0.4)$, lorsque $a_1 = 7$ et $m = 17$

ρ_2 réseaux translatés avec $[0,1)^t$. Chacun des ρ_2 réseaux est translaté par un vecteur $\mathbf{u}_{2,0,t} \in \Psi_{2,t}$.

Il est à noter que par le lemme 5.2.1, nous pouvons remplacer un point $\mathbf{u}_{2,0,t}$ de $\Psi_{2,t}$ par un autre point $\tilde{\mathbf{u}}_{2,0,t}$ obtenu en ajoutant à $\mathbf{u}_{2,0,t}$ une combinaison linéaire entière de la base B du réseau décrivant $\Psi_{1,t}$, sans changer l'ensemble $(\Psi_{1,t} + \mathbf{u}_{2,0,t}) \bmod 1$. Le générateur combiné G demeure donc identique.

Exemple 5.2.2 *Considérons un exemple avec $t = 2$. Prenons comme générateur G_1 un LCG de paramètre $a_1 = 8$ et de module $m = 29$. Ce LCG a une période maximale $\rho_1 = m - 1 = 28$. Si nous dessinons dans le plan l'ensemble des points $(u_{1,0}, u_{1,1})$ possibles, nous obtenons les 29 points (•) de la figure 5.4. La structure de réseau des points est bien visible sur cette figure, et deux vecteurs de base possibles $\mathbf{v}_1 = (4/29, 3/29)$ et $\mathbf{v}_2 = (1/29, 8/29)$ sont illustrés. Un parallélogramme P formé par les deux vecteurs de base a aussi été dessiné. Supposons que le générateur G_2, de période $\rho_2 = 3$, produise les trois points $(0.6, 0.6)$, $(0.3, 0.6)$ et $(0.6, 0.3)$, tel qu'illustré à la figure 5.4 (□). Alors, les points obtenus par le générateur combiné sont ceux de la figure 5.5.*

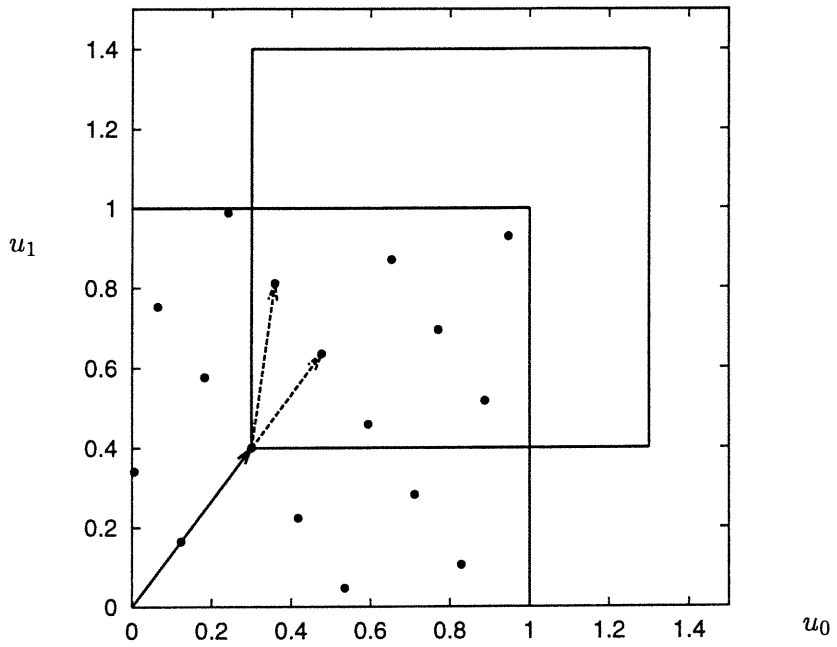


Figure 5.3: Points de $(\Psi_{1,2} + (0.3, 0.4)) \bmod 1$, lorsque $a_1 = 7$ et $m = 17$

Nous savons qu'il est possible d'ajouter aux points de G_2 une combinaison linéaire entière de la base du réseau du LCG, sans changer les points du générateur combiné. Par exemple, remplaçons le point $(0.6, 0.6)$ de G_2 par $(0.6, 0.6) - 4 \times v_1 \approx (0.0483, 0.186)$. Nous obtenons alors un nouveau point équivalent situé dans le parallélogramme P. De même, nous pouvons remplacer les deux autres points de G_2 par des points équivalents situés dans le parallélogramme P, tel qu'illustré sur la figure 5.6. Nous voyons bien ici que les points du générateur combiné correspondent à la superposition de trois réseaux translatés, chacun translaté par un des trois points situés dans le parallélogramme P.

Nous pouvons généraliser la démarche de l'exemple précédent en dimension t. Soit P l'hyperparallélogramme formé par les vecteurs de base $B = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_t)$, et soit $\tilde{\Psi}_{2,t}$ les points de $\Psi_{2,t}$ auxquels nous avons ajouté une combinaison linéaire entière de la base B pour les ramener dans l'hyperparallélogramme P. Les points de Ψ_t correspondent à la superposition des intersections de ρ_2 réseaux translatés, chacun translaté par un point de $\tilde{\Psi}_{2,t}$, avec $[0, 1)^t$.

Le fait de ramener les points du générateur G_2 dans l'hyperparallélogramme P nous

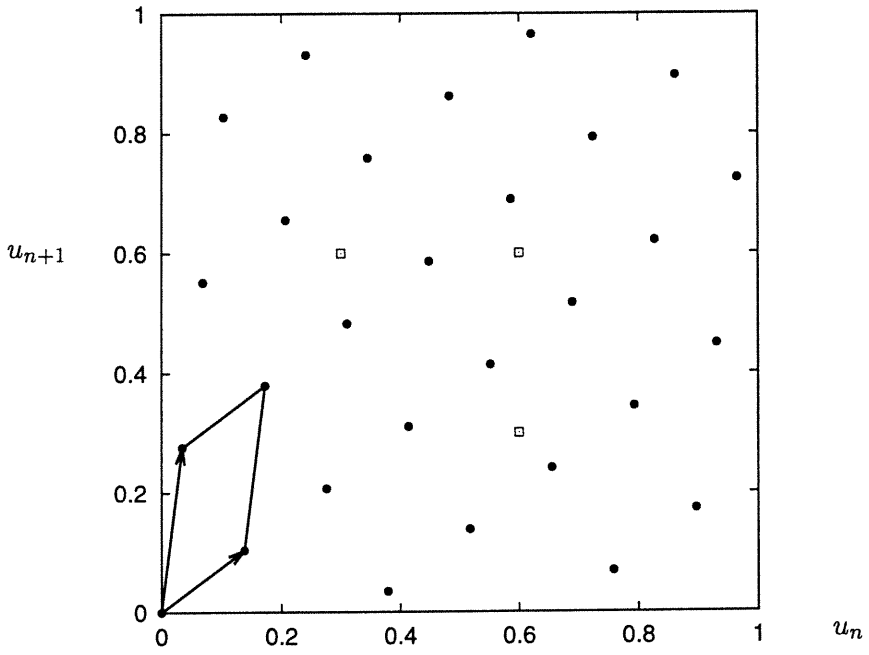


Figure 5.4: Points générés par un LCG avec $a_1 = 8$ et $m = 29$ (•) et par un deuxième générateur G_2 (□)

permet d'évaluer la distribution des points obtenus par le générateur combiné. En effet, une petite distance entre deux points de G_2 dans P signifie que deux réseaux seront translatés d'une valeur semblable, et que les points obtenus seront près l'un de l'autre. Si les points sont bien distribués dans l'hyperparallélogramme, alors des translations relativement différentes seront faites, et les points du générateur combiné couvriront bien l'espace $[0,1]^t$.

Ceci nous donne un critère pour choisir G_2 de telle sorte que les points de Ψ_t soient le mieux répartis possibles.

Critère de sélection 5.2.1 *Nous cherchons des générateurs G_2 pour lesquels les points de $\tilde{\Psi}_{2,t}$ sont bien distribués dans P.*

Ce critère, qui semble simple et attrayant, pose toutefois quelques difficultés en pratique. En fait, la base du réseau associé à un LCG particulier peut être obtenue facilement, par exemple en utilisant la méthode décrite dans [26]. Mais une fois cette

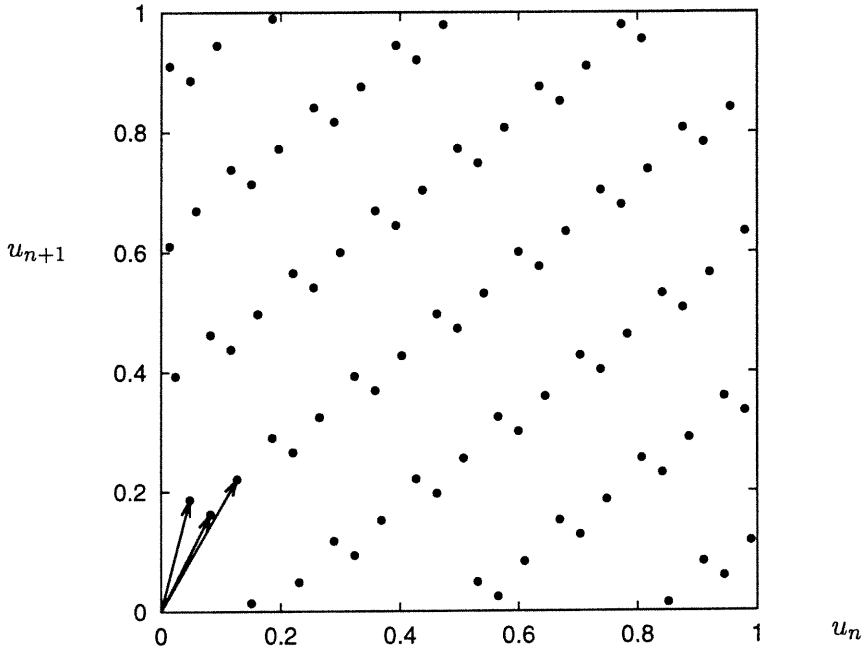


Figure 5.5: Points générés par le générateur combiné de l'exemple 5.2.2

base calculée, il n'est pas évident de trouver efficacement la combinaison linéaire entière permettant de ramener un point quelconque dans l'hyperparallélogramme désiré, même en dimension aussi petite que $t = 2$. Et comme ce calcul doit être fait pour chaque point des générateurs G_2 à considérer, et idéalement pour différentes dimensions t, une quantité de travail (et de temps) très élevée serait nécessaire pour choisir un seul générateur G_2 de petite période. Or, nous aimerions obtenir un critère de sélection assez efficace pour pouvoir être appliqué à plusieurs types de générateurs G_2, ayant différentes périodes relativement grandes. Malheureusement, nous n'avons pas réussi à trouver un tel critère, lorsque la combinaison est faite en additionnant, modulo 1, les sorties d'un MRG et d'un autre générateur. Cependant, si les points du MRG sont eux mêmes bien distribués, la théorie développée nous suggère qu'un certain minimum d'uniformité sera conservé suite à la combinaison, puisque plusieurs copies translatées du «bon» réseau du MRG sont obtenues. Des combinaisons impliquant plusieurs types de générateurs G_2 seront essayées au prochain chapitre, et des tests empiriques permettront de vérifier le degré d'uniformité des points produits.

Il est à noter que le critère 5.2.1 développé, quoique difficile à appliquer avec la

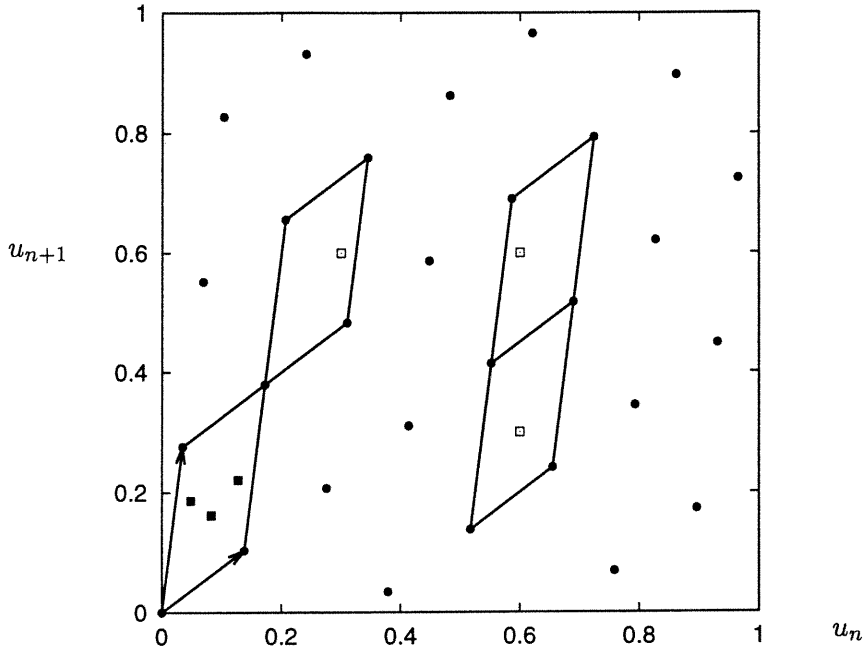


Figure 5.6: Points générés par un LCG avec $a_1 = 8$ et $m = 29$ (•), par un deuxième générateur G_2 (□) et points équivalents dans le parallélogramme P (■)

combinaison présentée, peut nous être utile pour envisager d'autres types de combinaisons prometteuses. En effet, ce critère indique que l'uniformité des points produits par le générateur G_2 dans l'hypercube $[0,1)^t$ importe peu, puisque c'est la répartition des points ramenés dans l'hyperparallélogramme P qui affecte le résultat final. Aussi, il pourrait être avantageux de construire un générateur G_2 produisant uniquement des points dans l'hyperparallélogramme P, ce qui nous permettrait d'appliquer le critère 5.2.1 directement. Par exemple, si $(\mathbf{v}_1, \mathbf{v}_2)$ est la base du réseau du MRG considéré en dimension $t = 2$, et que $(u_{2,n}, u_{2,n+1})$ représentent deux sorties consécutives d'un deuxième générateur, la combinaison linéaire $u_{2,n}\mathbf{v}_1 + u_{2,n+1}\mathbf{v}_2$ pourrait être formée, pour donner un point en deux dimensions situé dans l'hyperparallélogramme P formé par les vecteurs $(\mathbf{v}_1, \mathbf{v}_2)$. Ce point pourrait ensuite être additionné, modulo 1, avec deux sorties consécutives produites par le MRG, pour donner une sortie en deux dimensions du générateur combiné. Cette technique aurait l'avantage de produire des points de dimension 2 bien répartis dans l'espace $[0,1)^2$, à condition que les points du deuxième générateur soient eux-mêmes bien distribués dans $[0,1)^2$. Le même processus pourrait

être appliqué dans des dimensions supérieures, et des points en dimension t ayant une bonne uniformité dans l'espace $[0,1)^t$ pourraient être obtenus.

Cependant, le problème principal de cette approche vient du fait que les points produits ont une dimension t fixée. Or, dans la majorité des programmes de simulation, les vecteurs aléatoires utilisés ont des dimensions variables, et souvent inconnues. Cette façon de combiner ne sera donc pas retenue dans le reste de ce mémoire. Néanmoins, elle pourrait s'avérer utile pour les applications nécessitant la génération d'un grand nombre de vecteurs aléatoires d'une même dimension t connue.

5.3 Combinaison d'un LFSR avec un autre générateur

Regardons maintenant ce qui se produit si la combinaison (5.2) est utilisée avec un générateur G_1 de type LFSR, et un générateur G_2 quelconque.

Par la proposition 5.1.2, nous savons que les points de Ψ_t sont obtenus en superposant ρ_2 ensembles de points de la forme $\Psi_{1,t} \oplus \mathbf{u}_{2,0,t}$. Dans ce qui suit, nous allons montrer que si $\Psi_{1,t}$ est (t,l)-équidistribué, alors $\Psi_{1,t} \oplus \mathbf{w}_t$ l'est aussi, quelque soit le vecteur à t composantes $\mathbf{w}_t$. Pour ce faire, le lemme suivant, dont la preuve est évidente, sera utile.

Lemme 5.3.1 *$w_1 = w_2 \Leftrightarrow w_1 \oplus w = w_2 \oplus w$, pour w, w_1 et w_2 valant 0 ou 1.*

Proposition 5.3.1 *Si $\Psi_{1,t}$ est (t,l)-équidistribué, alors $\Psi_{1,t} \oplus \mathbf{w}_t$ l'est aussi.*

Preuve : D'abord, seuls les l premiers bits des coordonnées des points de $\Psi_{1,t}$ détermineront la case dans laquelle les points se retrouveront. Sans perte de généralité, supposons donc que les coordonnées des points ont l bits. Par hypothèse, $\Psi_{1,t}$ est (t,l)-équidistribué, et donc chacune des 2^{tl} cases possibles contient le même nombre de points. Par le lemme 5.3.1, nous savons que les points de $\Psi_{1,t} \oplus \mathbf{w}_t$ qui auront les mêmes coordonnées (et donc tomberont dans la même case) sont ceux qui avaient les mêmes coordonnées (et donc étaient dans la même case) avant de faire le ou-exclusif.

La propriété d'équidistribution est donc conservée suite au ou-exclusif. □

Si le LFSR G_1 est tel que $\Psi_{1,t}$ est (t,l)-équidistribué, chaque ensemble $\Psi_{1,t} \oplus \mathbf{u}_{2,0,t}$ sera également (t,l)-équidistribué. Comme Ψ_t est formé de l'union d'ensembles (t,l)-équidistribués, nous obtenons la proposition suivante :

Proposition 5.3.2 *Si $\Psi_{1,t}$ est (t,l)-équidistribué, alors Ψ_t l'est aussi.*

La combinaison présentée ici a donc l'avantage de préserver l'équidistribution du générateur LFSR utilisé.

Exemple 5.3.1 *Considérons un exemple en deux dimensions. Prenons un générateur G_1 de type LFSR dont la récurrence suit la forme simplifiée (2.2). Supposons que les paramètres de ce générateur sont donnés par $k = 4$, $r = 3$ et $s = 2$. Ce générateur a une période maximale $\rho_1 = 15$. Si nous dessinons dans le plan l'ensemble des points $(u_{1,0}, u_{1,1})$ possibles, nous obtenons les 16 points (•) de la figure 5.7. Nous voyons sur cette figure que les points sont $(2,2)$-équidistribués. Soit un générateur G_2 de période 4, dont les points sont $(0.2, 0.8)$, $(0.8, 0.1)$, $(0.1, 0.4)$ et $(0.4, 0.2)$, tel qu'illustré à la figure 5.7 (□). Alors, les points obtenus par le générateur combiné sont ceux de la figure 5.8. Nous voyons bien que les points sont encore $(2,2)$-équidistribués, puisque chaque case contient exactement quatre points.*

Il serait intéressant de trouver des conditions sur $\Psi_{1,t}$ ou $\Psi_{2,t}$ permettant d'augmenter l'équidistribution du générateur combiné de (t,l) à (t,l'), où $l' > l$, lorsque $l' \leq l_t^*$, la résolution maximale possible en dimension t.

Il peut paraître simple, à première vue, de trouver de telles conditions. D'abord, un raisonnement similaire à celui utilisé dans la preuve de la proposition 5.3.2 peut être appliqué à l'ensemble $\Psi_{2,t}$, pour donner la proposition suivante :

Proposition 5.3.3 *Si les bits $l+1$, ..., l' de $\Psi_{2,t}$ sont équidistribués en dimension t (si chaque combinaison possible pour les bits $l+1$, ..., l' de $\Psi_{2,t}$ revient le même nombre de fois), alors les bits $l+1$, ..., l' de Ψ_t sont aussi équidistribués.*

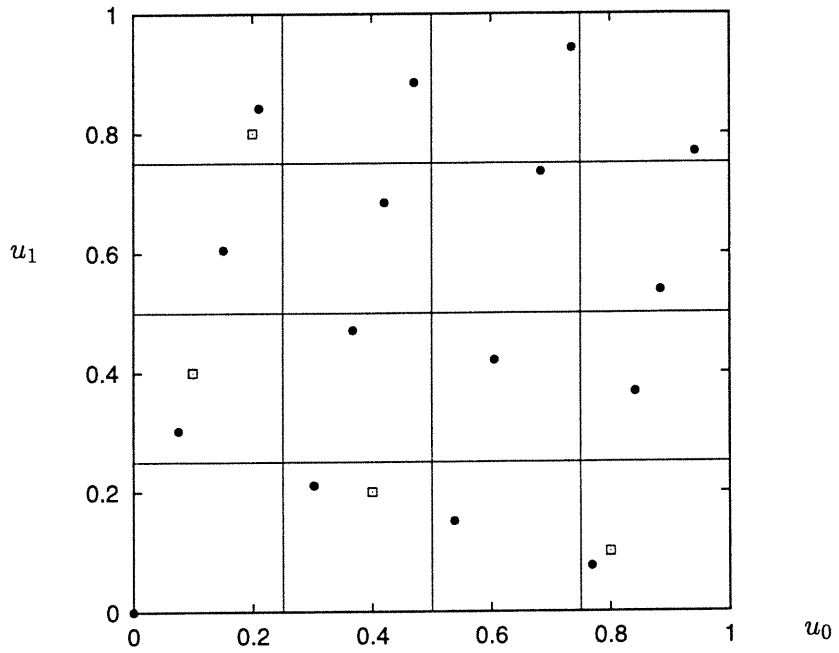


Figure 5.7: Points générés par un LFSR avec $k = 4$, $r = 3$ et $s = 2$ (•), et points d'un deuxième générateur G_2 (□)

Preuve : Par le même raisonnement qu'à la proposition 5.3.1, les bits $l + 1$, ..., l' de l'ensemble $\Psi_{2,t} \oplus \mathbf{w}_t$ demeurent équidistribués, et ce quelque soit le vecteur à t composantes $\mathbf{w}_t$. Comme Ψ_t peut aussi s'écrire comme $\bigcup_{s_{1,0} \in S_1} \Psi_{2,t} \oplus \mathbf{u}_{1,0,t}$, il est formé de l'union d'ensembles ayant leurs bits $l+1$, ..., l' équidistribués, d'où la proposition. □

À partir de ce résultat, il serait tentant de généraliser la proposition 5.3.2 de la façon suivante : *Si $\Psi_{1,t}$ est (t,l)-équidistribué, et que les bits $l+1,...,l'$ de $\Psi_{2,t}$ sont équidistribués en dimension t, alors Ψ_t est (t,l')-équidistribué.*

Cette généralisation est cependant fausse. Pour avoir la (t,l')-équidistribution, il faudrait que chaque combinaison possible, pour les l' premiers bits des coordonnées des points de Ψ_t, revienne le même nombre de fois. Or, par la (t,l)-équidistribution de Ψ_t, nous savons que chaque combinaison possible, pour les l premiers bits des points de Ψ_t, revient le même nombre de fois. Nous savons aussi, par l'équidistribution des bits $l+1,...,l'$, que chaque combinaison possible pour les bits $l+1,...,l'$ des points de Ψ_t revient le même nombre de fois. Cependant, rien ne nous assure, en considérant les l'

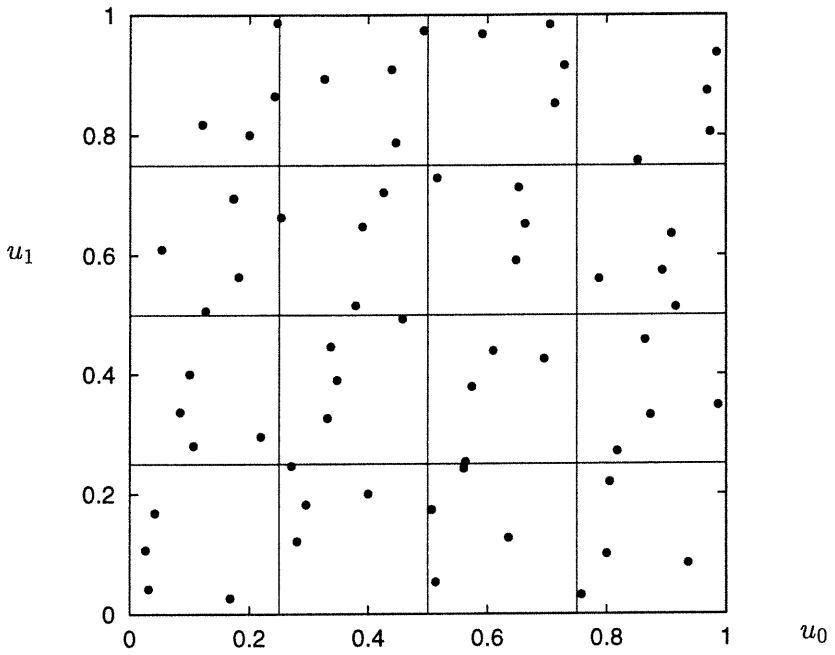


Figure 5.8: Points générés par le générateur combiné de l'exemple 5.3.1

premiers bits, que chaque combinaison possible pour les l premiers bits sera suivie de chaque combinaison possible pour les bits $l+1, ..., l'$, et encore moins que si tel est le cas, chaque combinaison se produira le même nombre de fois.

Exemple 5.3.2 *Considérons un exemple en une dimension. Supposons qu'un générateur produise huit nombres, dont les trois premiers bits sont* 000, 000, 011, 011, 100, 100, 111, 111. *Ces nombres sont* $(1,2)$*-équidistribués, puisque chaque combinaison possible pour les deux premiers bits revient deux fois. Le troisième bit est aussi équidistribué en une dimension, puisqu'il y a autant de* 0 *que de* 1. *Cependant, ces nombres ne sont pas* $(1,3)$*-équidistribués, puisque, entre autres, les combinaisons* 001, 010, 101 *et* 110 *ne sont jamais présentes.*

L'exemple suivant poursuit l'exemple 5.3.1, et permet de voir graphiquement ce qui se produit.

Exemple 5.3.3 *Reprenons l'exemple 5.3.1. Les points* $\Psi_{1,2}$ *du générateur LFSR sont* $(2,2)$*-équidistribués. De plus, le troisième bit des points de* $\Psi_{2,2}$ *est équidistribué en*

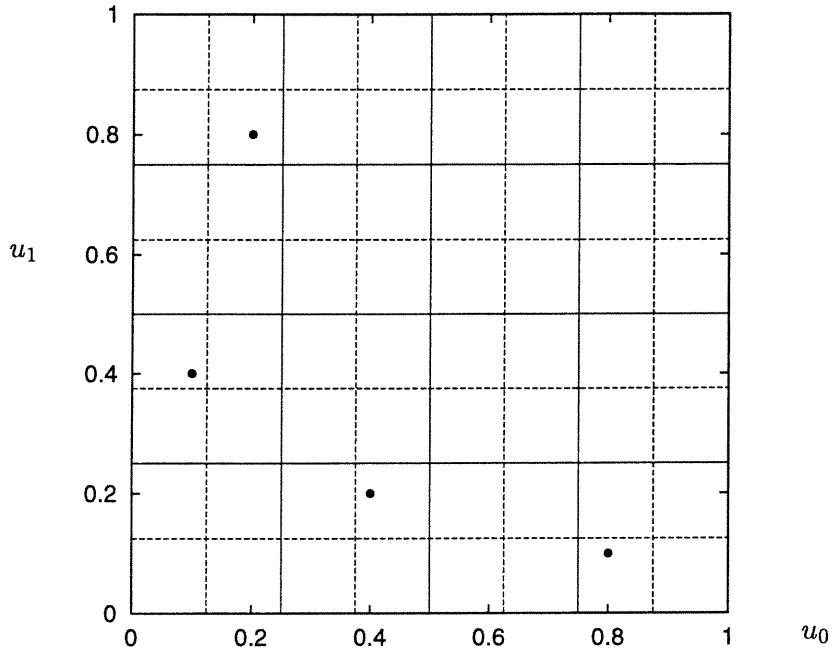


Figure 5.9: Points du générateur G_2 de l'exemple 5.3.1

dimension 2. *Pour bien voir cette équidistribution, les quatre points du générateur* G_2 *ont été redessinés sur la figure 5.9, avec une division plus fine. Les deux premiers bits de chaque coordonnée déterminent dans quelle case délimitée par une ligne solide un point tombe, alors que le troisième bit de chaque coordonnée détermine l'endroit, dans ces cases, où le point se retrouve. Plus précisément, il détermine dans quelle case dessinée en pointillé le point tombe. Ainsi, comme le point* $(0.2, 0.8)$ *se retrouve dans la case pointillée du bas à droite de sa case solide, ses troisièmes bits sont* 1 *et* 0. *Comme chacun des quatre points tombe dans une case pointillée différente, à l'intérieur de leur case solide respective, les quatre possibilités pour le troisième bit de chaque coordonnée se présentent une fois, et le troisième bit est équidistribué en dimension* 2.

Cependant, en dessinant les points du générateur combiné avec une division plus fine, comme à la figure 5.10, nous constatons que les 64 *points ne sont pas* $(2, 3)$*-équidistribués. La* $(2, 2)$*-équidistribution assure qu'il y a autant de points dans chaque case solide, alors que l'équidistribution du troisième bit assure que le nombre de points situés dans toutes les cases pointillées en bas à gauche d'une case solide (cas où les troisième bits de chaque coordonnée sont* $(0, 0)$*) est le même que le nombre de points*

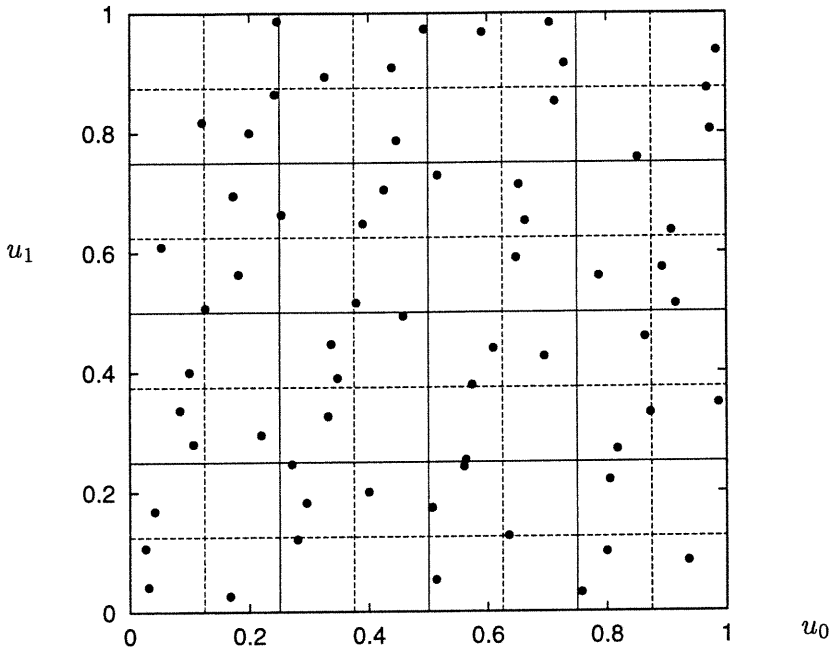


Figure 5.10: Points générés par le générateur combiné de l'exemple 5.3.1, avec une division du carré $[0,1]^2$ plus fine

situés dans toutes les cases pointillées en haut à gauche d'une case solide (cas où les troisième bits de chaque coordonnée sont $(0,1)$*). Le même raisonnement s'applique pour les cases pointillées situées à droite d'une case solide. Ceci n'implique cependant pas que chaque case pointillée pris individuellement contient le même nombre de points.*

Il est difficile de trouver des conditions sur les points de $\Psi_{1,t}$ ou de $\Psi_{2,t}$ nous assurant une meilleure équidistribution des points Ψ_t du générateur combiné. De telles conditions seraient utiles, puisqu'elles permettraient de sélectionner un premier générateur selon certains critères, puis de sélectionner un deuxième générateur selon d'autres critères, tout en assurant un certain niveau d'équidistribution pour le générateur combiné. Cependant, nous n'avons pas réussi à garantir une meilleure équidistribution que ne le fait la proposition 5.3.2, en considérant seulement les générateurs à combiner séparément. En fait, une fois que nous avons la (t,l)-équidistribution, d'après les conditions de la proposition 5.3.2, nous savons que si nous divisons l'espace $[0,1]^t$ en 2^{tl} cases, chaque case contient le même nombre de points de Ψ_t. Cependant, l'endroit où un point se retrouve, dans chaque case, dépend des bits $l+1,...,L$ de chaque point combiné. Il est

difficile de contrôler à la fois la case dans laquelle un point se retrouve, et l'endroit dans cette case où il se situe, sans analyser directement la structure des points du générateur combiné.

Pour obtenir une meilleure distribution des points, nous recommandons toutefois de s'assurer que le nombre de bits produits par la sortie $u_{2,n}$ du deuxième générateur soit supérieur ou égal au nombre L de bits successifs générés par le LFSR utilisé. En fait, si seuls les bits les plus significatifs varient, les points produits par le générateur combiné forment des motifs répétés dans certaines dimensions. Par exemple, un problème survient en dimension t, lorsque le LFSR est (t,l)-équidistribué, que chacune des 2^{tl} cases divisant l'espace $[0,1)^t$ contient plus d'un point, et que les sorties $u_{2,n}$ ne font varier que les $\tilde{l} \leq l$ premiers bits, avec $l < L$. Dans ce cas, les points situés dans une même des 2^{tl} cases se retrouvent exactement au même endroit à l'intérieur d'une autre case, suite au ou-exclusif avec un même point du deuxième générateur. En superposant les points obtenus après avoir effectué quelques ou-exclusifs, des motifs se répètent d'une case à l'autre, et une forte structure devient visible.

Exemple 5.3.4 *Prenons un LFSR dont la récurrence suit la forme simplifiée (2.2), de paramètres* $k = 3$, $r = 2$ *et* $s = 2$. *Ce générateur a une période maximale de* $\rho_1 = 7$. *L'ensemble des huit points* $(u_{1,0}, u_{1,1})$ *possibles sont dessinés à la figure 5.11* (•), *et l'espace* $[0,1]^2$ *est divisé en quatre, pour bien montrer que chaque case contient exactement deux points. Il s'agit donc d'un LFSR* $(2,1)$*-équidistribué. Supposons que les sorties du deuxième générateur* G_2 *ne font varier que le premier bit, de façon à ce que* $\tilde{l} = 1 \leq l = 1$. *En deux dimensions, ceci laisse quatre possibilités pour les points de* G_2, *soient* $(0,0)$, $(0,0.5)$, $(0.5,0)$ *et* $(0.5,0.5)$. *Ces quatre points sont illustrés à la figure 5.11* (□). *Lors de la combinaison avec un point de* G_2, *les points du LFSR situés à l'intérieur d'une même case vont se retrouver ensemble, possiblement dans une autre case. Cependant, comme le seul bit modifié est le premier, les points vont conserver leur même place relative, à l'intérieur de la nouvelle case. Si toutes les combinaisons possibles sont effectuées, chacun des points du LFSR visite chacune des cases, tout en conservant sa place relative. L'emplacement des points dans chacune des quatre cases devient alors identique, comme l'indique la figure 5.12.*

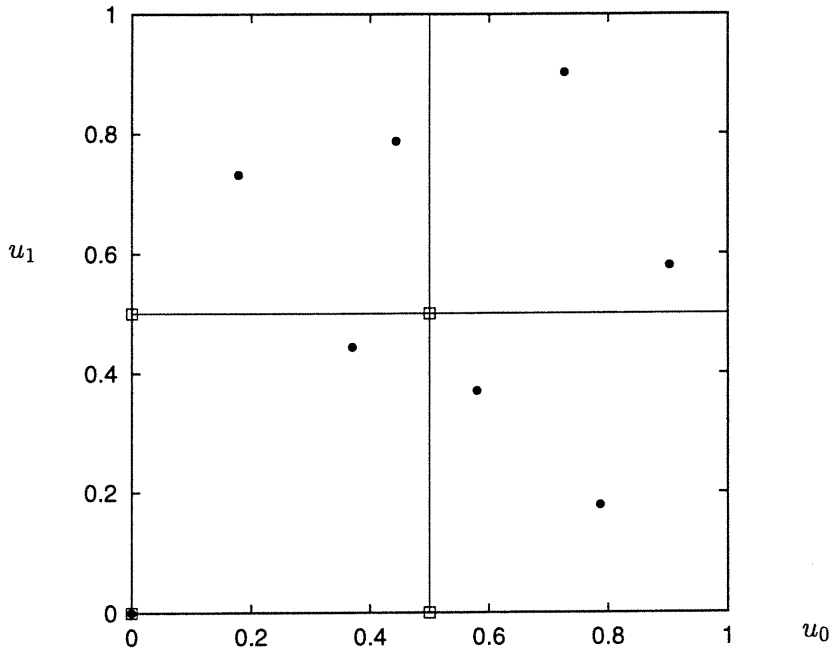


Figure 5.11: Points générés par un LFSR avec $k = 3$, $r = 2$ et $s = 2$ (•), et points d'un deuxième générateur G_2 (□)

À l'inverse, si seuls les bits les moins significatifs varient, les points produits par le générateur combiné n'atteignent pas certaines régions de l'hypercube $[0,1)^t$, pour des dimensions t assez grandes. Par exemple, un problème évident survient lorsque la dimension t nous permet de diviser l'espace $[0,1)^t$ en 2^{tl} cases, avec $l_t^* < l < L+1$, où l_t^* est la résolution maximale possible du LFSR en dimension t, et que les sorties $u_{2,n}$ ne font varier que les bits $\tilde{l} \geq l+1$. Dans ce cas, suite à la combinaison, les points du LFSR restent toujours dans une même des 2^{tl} cases, puisque les l premiers bits demeurent inchangés. Comme la valeur de l dépasse la résolution maximale l_t^*, certaines cases ne contiennent aucun point du LFSR, et demeurent toujours vides suite à la combinaison. Lorsque la dimension t considérée est suffisamment grande, ce problème se produit dès que $\tilde{l} \geq 2$. Il faut donc éviter de faire varier uniquement les derniers bits, et surtout s'assurer que le premier bit le plus significatif puisse prendre les valeurs 0 et 1.

L'utilisation de sorties $u_{2,n}$ faisant varier les différents bits du LFSR permet aussi d'augmenter la période de chacun des L bits les plus significatifs du générateur combiné, un avantage à ne pas négliger.

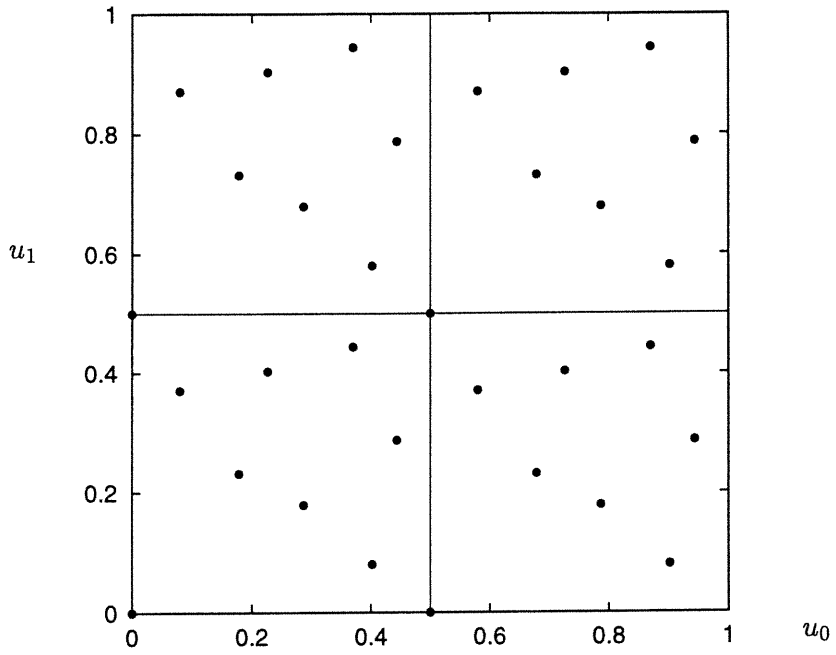


Figure 5.12: Points générés par le générateur combiné de l'exemple 5.3.4

L'uniformité de différents générateurs combinant des sorties de 31 bits est testée empiriquement au prochain chapitre. Les modules m_2 des générateurs G_2 sont tous choisis premiers (et donc impairs puisqu'ils sont plus grands que 2), ce qui augmente les chances de voir chacun des 31 bits les plus significatifs de $u_{2,n}$ varier, même avec de petits modules. Les sorties $u_{2,n}$ sont alors de la forme i/m_2, avec m_2 impair et $0 < i < m_2$, et ont une représentation binaire infinie.

Chapitre 6

Tests empiriques sur des familles de générateurs

La théorie présentée au chapitre précédent nous permet d'avoir une idée de la répartition des points produits par deux types de générateurs combinés : un MRG dont la sortie est additionnée, modulo 1, avec la sortie d'un deuxième générateur, et un LFSR combiné avec un générateur quelconque au moyen d'un ou-exclusif bit à bit. Nous savons qu'en considérant l'ensemble de tous les points pouvant être produits par ces générateurs combinés, une relativement bonne uniformité est assurée si les points du générateur linéaire de base (le MRG ou le LFSR) sont eux-mêmes bien distribués.

Mais plusieurs questions demeurent sans réponse, si nous nous fions uniquement à la théorie présentée. Par exemple, la «relativement bonne» uniformité de ces générateurs est-elle suffisante en pratique ? Comment se comportent les vecteurs de t valeurs successives $(u_{ti}, ..., u_{ti+t-1})$ générés ? Quel est le meilleur choix pour le deuxième générateur à combiner ? Un deuxième générateur non linéaire permet-il d'éliminer la structure trop régulière des points produits par le générateur linéaire de base ? Si oui, quelle doit être la taille de ce générateur non linéaire, par rapport à celle du générateur linéaire ? Et quelle fraction de la période peut-on utiliser sans risque, dépendant de la famille du deuxième générateur utilisé ?

Dans ce chapitre, nous avons voulu répondre à ces questions d'une manière empirique. Pour ce faire, nous avons soumis les sorties produites par plusieurs familles de générateurs combinés à différents tests statistiques. Plus spécifiquement, nous avons sélectionné, pour chaque famille testée, des petits générateurs de période $\rho \approx 2^e$, pour différents entiers e. Nous avons ensuite cherché, pour chaque test appliqué, à trouver à partir de quelle fraction de la période les générateurs d'une même famille commencent

à échouer le test.

Seule une quantité limitée de tests ont été appliqués, le but de cette étude empirique étant surtout d'explorer et d'essayer une grande variété de générateurs combinés. Néanmoins, ces tests ont été choisis avec soin, et les résultats obtenus nous permettent de mieux évaluer les gains et les risques potentiels pouvant survenir avec l'utilisation de différents générateurs, et nous permettent d'augmenter notre confiance envers certaines familles particulières de générateurs.

Il est à noter que les générateurs présentés dans ce chapitre sont beaucoup trop petits pour être utilisés en pratique. Le but ici n'est pas de suggérer un bon générateur particulier, mais bien de comprendre comment se comportent les séquences de nombres produites par les générateurs d'une même famille, face à certains tests statistiques. L'utilisation de petits générateurs permet d'effectuer les tests statistiques plus rapidement, et donc d'en appliquer un plus grand nombre, à une plus grande variété de générateurs. Suite aux résultats obtenus, nous avons pu développer de bons générateurs avec de longues périodes, qui sont présentés au chapitre suivant.

La prochaine section donne des notions de base sur les tests statistiques, et explique le choix des tests que nous avons appliqués. Les tests retenus sont décrits plus en détails à la section suivante. Les différentes familles de générateurs que nous avons décidé de tester sont ensuite présentées, et les résultats obtenus sont donnés dans les dernières sections.

6.1 Notions de base et choix des tests statistiques

Les tests statistiques appliqués aux séquences de nombres produites par un générateur sont tous basés sur certains principes communs. D'abord, l'hypothèse à tester, souvent appelée hypothèse nulle et notée $\mathcal{H}_0$, est la suivante : les sorties u_n produites par le générateur sont des variables aléatoires i.i.d. de loi U(0,1). Dans ce cas, les vecteurs $\mathbf{u}_{n,t} = (u_n, ..., n_{n+t-1})$ devraient être uniformément distribués dans l'hypercube $[0,1)^t$, et indépendants lorsqu'ils sont disjoints, et ce peu importe la valeur de n et t. Nous

savons très bien que cette hypothèse est fausse, puisque les nombres sont produits par un algorithme déterministe. Mais nous sommes intéressés à savoir dans quelle mesure les résultats statistiques risquent d'être affectés par l'utilisation de ces nombres pseudo-aléatoires.

Dans ce contexte, un test statistique peut être défini comme une fonction $\mathcal{T}$ à valeur réelle, qui dépend d'un nombre fini de sorties produites par le générateur à tester. La fonction $\mathcal{T}$ doit être choisie de telle sorte que la distribution F de la variable aléatoire $\mathcal{T}$ soit connue, sous l'hypothèse $\mathcal{H}_0$. La statistique $\mathcal{T}$ peut ensuite être calculée à partir des sorties produites par un générateur, et la valeur t_1 prise par $\mathcal{T}$ peut être analysée. En général, la p-valeur du test est obtenue, et indique la probabilité d'obtenir un résultat aussi extrême que t_1, sous l'hypothèse $\mathcal{H}_0$. Lorsque la distribution F est continue, cette p-valeur est définie comme $p_1 = P[T \geq t_1 \mid \mathcal{H}_0] = 1 - F(t_1)$, et devrait suivre une loi uniforme sur l'intervalle $[0, 1]$. Dans le cas discret, nous avons plutôt suivi la convention employée dans [22, 27], en définissant la p-valeur comme suit :

$$p_1 = \begin{cases} P[T \geq t_1 \mid \mathcal{H}_0] & \text{si } P[T \geq t_1 \mid \mathcal{H}_0] < P[T \leq t_1 \mid \mathcal{H}_0], \\ 1 - P[T \leq t_1 \mid \mathcal{H}_0] & \text{sinon.} \end{cases}$$

Dans les deux cas, une valeur de p_1 trop près de 0 ou de 1 nous oblige à rejeter l'hypothèse $\mathcal{H}_0$, et nous disons alors que le générateur a échoué le test.

Jusqu'à maintenant, un très grand nombre de tests ont été développés, pour essayer de détecter des défauts de structure dans les séquences produites par différentes familles de générateurs [9, 17, 22, 23, 24, 27, 30, 31]. Ainsi, certains tests sont très sensibles à la structure de réseau des MRG, alors que d'autres détectent plutôt la trop grande uniformité des points produits par les LFSR maximalement équidistribués. Parmi le vaste choix de tests statistiques disponibles, nous avons décidé d'en appliquer quelques-uns réputés pour être sensibles aux défauts des MRG et des LFSR. Nous voulons ainsi être en mesure de vérifier si la non-linéarité introduite dans les générateurs combinés permet bien d'améliorer la structure trop régulière des MRG ou des LFSR. Nous avons aussi pris soin de choisir les quelques tests pour lesquels certains générateurs non linéaires performent moins bien, pour vérifier si les défauts de structure détectés par ces tests

demeurent présents avec les générateurs combinés.

Le premier test que nous avons retenu est celui des points les plus rapprochés (connu sous le nom de «close-point spatial test» en anglais) [22, 24, 27]. Ce test, basé sur une statistique qui tient compte de la distance entre les M paires de points générés les plus près, est très sensible à la structure de réseau des MRG. Lorsque les paramètres de ce test sont choisis avec soin, les MRG ayant une bonne valeur au test spectral commencent à échouer avant même que la racine carrée de la période ne soit épuisée [24, 27].

Les tests suivants que nous avons appliqués sont souvent désignés par «tests sériels» (ou «serial tests» en anglais) [9, 22, 27, 31]. Nous avons retenu le test de collisions et le test du khi-deux. Les MRG ayant une bonne valeur au test spectral et les LFSR maximalement équidistribués échouent ces tests assez rapidement (lorsqu'environ 16 fois la racine carrée de leur période est épuisée [27, 31]), parce qu'une trop grande uniformité est détectée.

Le test de l'espacement des anniversaires (ou «birthday spacings» en anglais) a aussi été retenu [9, 22, 30]. Ce test est particulièrement discriminant pour les MRG, qui peuvent commencer à échouer lorsque seulement quatre fois la racine cubique de la période est prise [30]. Ce test est aussi très costaud pour les LFSR, et les générateurs cubiques ont éprouvé quelques difficultés avec certains choix de paramètres.

Finalement, le test de l'espacement des apparitions (ou «appearance spacings» en anglais) a été appliqué [22, 36]. Ce test, basé sur une mesure d'entropie d'une suite de bits aléatoires, est qualifié d'«universel» dans [36], dans le sens où il peut détecter, avec une taille échantillonnale suffisamment grande, toute déviation significative des sorties d'un générateur, par rapport à une vraie séquence aléatoire. Très peu de résultats empiriques ont cependant été publiés, et il est difficile de savoir s'il s'agit d'un test discriminant pour les MRG, les LFSR ou les générateurs non linéaires. Nous avons décidé de retenir ce test, pour étudier la question.

6.2 Description des tests retenus

6.2.1 Test des points les plus rapprochés

Considérons des vecteurs de dimension t disjoints, $\mathbf{u}_{ti,t} = (u_{ti}, ..., u_{ti+t-1})$, produits par un générateur, pour $i = 0, ..., n-1$. Notons par $Y_n(s)$ le nombre de paires $(\mathbf{u}_{ti,t}, \mathbf{u}_{tj,t})$, avec $i < j$, dont la distance est bornée comme suit :

$$\|\mathbf{u}_{tj,t} - \mathbf{u}_{ti,t}\|_p \leq \left(\frac{2s}{n(n-1)V_t}\right)^{1/t}, \tag{6.1}$$

où $\| \ \|_p$ est la norme L_p sur $[0,1)^t$ considéré comme un hypertore (c'est-à dire que des points situés sur des faces opposées de l'hypercube peuvent être près les uns des autres), et V_t est le volume de la sphère unitaire $\{x \in \mathbb{R}^t \mid \|x\|_p \leq 1\}$. Il a été montré dans [24] que sous $\mathcal{H}_0$, le processus stochastique $\{Y_n(s), 0 \leq s \leq s_1\}$ converge faiblement vers un processus de Poisson de taux unitaire, lorsque $n \to \infty$. Ainsi, si $T_{n,i} = \min\{s \geq 0 \mid Y_n(s) = i\}$ est le point du $i$ième saut du processus $Y_n(s)$, les distances $\Delta_{n,i} = T_{n,i} - T_{n,i-1}$ sont des variables aléatoires i.i.d. suivant approximativement une loi exponentielle de moyenne unitaire, pour $1 \leq i \leq M$, sous $\mathcal{H}_0$. Cette observation a donné lieu au test des points les plus rapprochés, décrit dans [22, 24, 27].

Lorsque $M = 1$, la valeur δ_1 de $\Delta_{n,1}$ est calculée, et la p-valeur du test est donnée par $p_1 = P[\Delta_{n,1} > \delta_1 \mid \mathcal{H}_0] = 1 - \exp(-\delta_1)$. Il s'agit du test NP (pour «nearest-pair» en anglais). Lorsque $M > 1$, les $\Delta_{n,i}$, $1 \leq i \leq M$, sont transformés en $W^*_{n,i} = 1 - \exp(-\Delta_{n,i})$, des variables aléatoires approximativement i.i.d. U(0,1), sous $\mathcal{H}_0$. Un test statistique de Anderson-Darling [24, 27] est ensuite généralement utilisé, pour comparer la distribution des $W^*_{n,i}$ obtenus à celle d'une loi U(0,1). Il s'agit dans ce cas du test M-NP (pour «M-nearest-pairs»).

Plusieurs valeurs de M, n, t et p ont été testées dans [24]. Parmi les conclusions à retenir, mentionnons que l'approximation exponentielle est acceptable si $n \geq 4M^2$. Aussi, la limite principale dans l'application de ce test est la grande quantité de mémoire requise pour emmagasiner les points $\mathbf{u}_{ti,t}$ produits. Une façon d'augmenter la puissance du test, tout en gardant une valeur de n raisonnable, est de le répéter N fois, et

d'appliquer encore une fois un test d'Anderson-Darling aux N p-valeurs obtenues, pour comparer leur distribution à celle d'une loi U(0,1). Dans ce cas, l'approximation utilisée est valide si $n \geq 4M^2\sqrt{N}$. Il est cependant suggéré dans [24] de d'abord augmenter la taille de n au maximum avant d'augmenter la valeur de N. Mentionnons finalement que la puissance du test augmente avec n et M, et qu'une norme L_∞ semble adéquate, puisqu'elle permet d'obtenir un test aussi sensible qu'avec les autres normes, tout en réduisant les temps de calcul.

Ce test des points les plus rapprochés est très sensible à la structure de réseau des MRG. Lorsque $M = 1$, les MRG ayant une bonne valeur au test spectral commencent à échouer le test parce que la distance séparant les deux points générés les plus près est trop grande. En fait, les points ont une structure de réseau très régulière, ce qui impose une limite inférieure sur la distance entre les points. Dans le cas où $M > 1$, il est fréquent que plusieurs paires de points séparés par la même distance soient retenues, forçant plusieurs $\Delta_{n,i}$ à prendre la valeur 0. Quelques $W^*_{n,i}$ près de 0 suffisent alors pour faire dévier la distribution obtenue d'une uniforme [0,1], près de l'origine. La statistique d'Anderson-Darling est particulièrement sensible à ce genre de déviation, près des extrémités de l'intervalle [0,1].

6.2.2 Tests sériels

Divisons l'intervalle $[0, 1]$ en d segments égaux, pour former une partition de l'hypercube $[0, 1]^t$ en $h = d^t$ parties ou «boîtes» égales. Considérons encore une fois les vecteurs de dimension t disjoints $\mathbf{u}_{ti,t} = (u_{ti}, ..., u_{ti+t-1})$, $i = 0, ..., n-1$, produits par un générateur, et notons par X_j le nombre de ces vecteurs tombant dans la boîte j. Sous l'hypothèse $\mathcal{H}_0$, chaque vecteur $\mathbf{u}_{ti,t}$ produit a une probabilité $1/h$ de se retrouver dans une des h boîtes, et le vecteur $X = (X_0, X_1, ..., X_{h-1})$ suit une loi multinomiale de paramètres $(n, 1/h, ..., 1/h)$. Plusieurs tests ont été développés à partir de cette observation, pour essayer de détecter une différence entre la distribution du vecteur X observé et celle de la loi multinomiale. Nous allons discuter de deux d'entre eux, soient le test du khi-deux (X^2) et le test de collisions (C), décrits dans [9, 22, 27, 31].

Les statistiques pour ces deux tests sont les suivantes :

$$X^2 = \sum_{j=0}^{h-1} \frac{(X_j - n/h)^2}{n/h},$$
$$C = \sum_{j=0}^{h-1} (X_j - 1) I[X_j > 1],$$

où I est la fonction indicatrice. La première statistique est une mesure de l'écart entre le nombre de vecteurs $\mathbf{u}_{ti,t}$ observés par boîte et le nombre de vecteurs espérés, alors que la deuxième statistique compte le nombre de collisions obtenues, c'est-à-dire le nombre de vecteurs $\mathbf{u}_{ti,t}$ tombant dans des boîtes déjà occupées. Dans les deux cas, une valeur plus élévée indique que les points sont distribués moins uniformément dans l'hypercube $[0,1]^t$.

Si h est fixé et $n \to \infty$, la statistisque X^2 converge en distribution vers une χ^2 à $h-1$ degrés de liberté [9, 31]. Pour une valeur de n fixée, l'approximation par une χ^2 est généralement utilisée lorsque $n/h > 1$, ce qui correspond au cas «dense». Notons que le test de collisions n'est pas applicable dans ce cas, puisque la statistique C tend vers l'infini lorsque $n \to \infty$.

Si $h \to \infty$, $n \to \infty$ et $n/h \to \delta$ pour $0 < \delta < \infty$, alors les deux statistiques décrites, corrigées en fonction de leur espérance et de leur variance, convergent vers une distribution normale standardisée $N(0,1)$ [31, 37]. Il s'agit du cas dit «clairsemé», utilisé en pratique lorsque le rapport n/h est suffisamment grand. Pour le test de collisions, l'approximation normale est cependant moins bonne lorsque l'espérance du nombre de collisions est faible, principalement parce que la statistique utilisée ne prend que des valeurs entières. Il est suggéré dans ce cas d'utiliser la loi de Poisson comme approximation [27].

Dépendant de la statistique et de l'approximation utilisée, une p-valeur peut être calculée. Un résultat trop près de 0 indique un manque d'uniformité dans la séquence observée, alors qu'un résultat trop près de 1 témoigne d'une trop grande uniformité. La procédure est parfois répétée N fois, et la distribution des p-valeurs obtenues est comparée à celle d'une loi uniforme par un test d'Anderson-Darling.

Plusieurs valeurs de N, n, d et t ont été essayées dans [27, 31]. Tout comme pour le test des points les plus rapprochés, pour une valeur de Nn fixée, il est recommandé de choisir $N = 1$. Il semble aussi que le cas clairsemé soit plus efficace pour détecter une différence entre les nombres produits et une vraie séquence aléatoire, du moins en ce qui concerne les générateurs linéaires. Un nombre de boîtes h approximativement égal à la période du générateur testé semble particulièrement efficace. Il faut cependant choisir la dimension t judicieusement, pour s'assurer que le $\log_2(d)$ ne dépasse pas le nombre de bits de précision du générateur utilisé. Si t est choisi trop petit, la division de l'hypercube $[0,1]^t$ est trop fine par rapport à la précision des nombres produits, et certaines boîtes restent toujours vides, ce qui fausse les statistiques calculées.

En général, les MRG ayant une bonne valeur au test spectral et les LFSR maximalement équidistribués ou sans-collisions échouent les tests présentés parce qu'une trop grande uniformité est observée. Entre autres, dans le cas des LFSR choisis sans-collisions, une très grande uniformité est mesurée lorsque le nombre de boîtes h est une puissance de 2 près de la période. En effet, le nombre de collisions est alors presque réduit à 0, et est toujours nul si la puissance de 2 dépasse la période. Ce comportement particulier nous incite à distinguer deux cas dans l'application des tests avec une valeur de h près de la période, soit celui où d est choisi impair, et celui où il est pris comme une puissance de 2 dépassant la période.

6.2.3 Test de l'espacement des anniversaires

Le test de l'espacement des anniversaires est une variante du test de collisions que nous venons de décrire. Supposons que l'hypercube $[0,1]^t$ est divisé en $h = d^t$ parties égales, comme précédemment, que les boîtes sont numérotées de 0 à $h-1$, et que $I_0 \leq ... \leq I_{n-1}$ représentent les numéros des boîtes dans lesquelles les vecteurs $\mathbf{u}_{ti,t}$ se retrouvent, classés par ordre croissant. Nous pouvons former les différences $S_j = I_{j+1} - I_j$, et compter le nombre de collisions Y entre ces différences. Ceci est équivalent à observer les dates d'anniversaires de n personnes, en supposant qu'une année compte h jours, et à calculer le nombre de collisions survenant entre l'espacement

des fêtes, d'où le nom du test.

Il a été montré dans [9] que sous $\mathcal{H}_0$, si n est grand et que $\lambda = n^3/(4h)$ n'est pas trop grand, Y suit approximativement une loi de Poisson de moyenne λ. Des expériences empiriques ont montré que cette approximation est valide si $h \geq (4\lambda)^4$, ou de façon équivalente, en remplaçant λ par $n^3/(4h)$ dans l'inégalité, si $n^3 \leq h^{5/4}$ [30]. La puissance du test est aussi discutée dans [30], et une valeur de $\lambda \approx 1$ est suggérée. Dans ce cas, une analyse théorique montre que si h est près de la période du générateur testé, les MRG devraient commencer à échouer le test lorque n est dans l'ordre de la racine cubique de la période. Les résultats empiriques de [30] ont confirmé cette analyse. Les LFSR ont aussi eu quelques difficultés avec ce test, de même que les générateurs cubiques en dimension t dépassant 5.

6.2.4 Test de l'espacement des apparitions

Partitionnons la séquence de nombres produite par un générateur en blocs de L bits, en ne conservant que les s premiers bits de chaque sortie. Générons $Q + K$ blocs de L bits ainsi formés, les Q premiers blocs servant à initialiser le test. Pour chacun des blocs $n = Q + 1, ..., Q + K$ suivant l'initialisation, posons $A_n = i$ si la plus récente apparition du $n$ième bloc est le bloc $n - i$, et posons $A_n = n$ s'il s'agit de la première apparition du bloc. La moyenne du logarithme en base 2 des A_n peut être calculée, pour donner la statistique du test de l'espacement des apparitions [22, 36] :

$$A = \sum_{n=Q+1}^{Q+K} \log_2(A_n)/K.$$

Sous $\mathcal{H}_0$, A suit approximativement une loi normale, dont la moyenne exacte et une variance approximée sont données dans [36], pour des valeurs de $1 \leq L \leq 16$.

Très peu de résultats empiriques, et aucun résultat systématique pour les MRG, les LFSR et les générateurs non linéaires sont disponibles pour ce test. Il est donc difficile de déterminer quelles valeurs de s, L, Q et K utiliser pour augmenter la sensibilité du test. Il est toutefois suggéré dans [36] de choisir des valeurs de L entre 6 et 16, et de prendre $Q \geq 10 \times 2^L$.

6.3 Familles de générateurs testées

Les tests décrits ont été appliqués à plusieurs familles de générateurs. Dans un premier temps, les générateurs linéaires de type MRG et LFSR ont été étudiés, de même que les générateurs non linéaires inversifs et cubiques. Les résultats obtenus ont pu servir de point de comparaison pour les générateurs combinés formés par la suite. Différentes combinaisons entre les familles de générateurs ont été essayées, correspondant toutes à une des deux formes analysées au chapitre précédent. Les différentes familles testées sont présentées dans le reste de cette section, et les paramètres des générateurs décrits sont donnés à l'annexe A.

6.3.1 MRG

Nous avons choisi des MRG d'ordre $k = 1$ (souvent appelé LCG) de modules m premiers, et de période maximale $\rho = m - 1$. Les modules m ont été pris comme étant les plus grands nombres premiers inférieurs à 2^e, pour e allant de 10 à 60. Quant aux paramètres des récurrences, nous avons pris ceux de [22]. Ces LCG ont été sélectionnés parmi ceux de [20] qui donnaient les meilleurs valeurs de M_8 (voir l'équation 2.7) au test spectral. Ils ont donc une bonne structure de réseau jusqu'en dimension 8. Ils ont aussi été utilisés à plusieurs reprises dans des articles appliquant certains tests statistiques [27, 30, 31], où ils ont été désignés par «GoodLCG».

Nous aurions aimé pouvoir également tester des MRG d'ordre $k = 2$ ou 3. Cependant, pour des périodes équivalentes à celles des LCG, l'erreur de discrétisation de ces MRG est très grande. Par exemple, les points produits par un LCG de période $2^{31} - 1$ sont des multiples de $1/2^{31}$, alors que les points produits par un MRG d'ordre 3 de même période ne sont que des multiples de $1/2^{31/3} \approx 1/2^{10}$. Dans ce cas, plusieurs points générés sont identiques, et plusieurs tests statistiques détectent très rapidement une différence entre ces points et ceux d'une vraie séquence aléatoire. Les tests effectués sur les LCG devraient cependant donner une bonne idée du comportement des MRG d'ordre k plus élevé, lorsque la période de ces générateurs devient plus grande, puisque

la même structure de réseau est présente.

6.3.2 LFSR

Pour chaque entier e entre 10 et 36, un LFSR combiné de période près de 2^e a été choisi. Les LFSR ont chacun deux composantes, dont les récurrences suivent la forme simplifiée (2.2). La combinaison est obtenue en faisant un ou-exclusif bit à bit entre les sorties, comme expliqué à la section 4.2. Les paramètres sélectionnés sont ceux des «LFSR2» de [22], et correspondent aux meilleurs valeurs trouvées en terme d'équidistribution. Pour la plupart des générateurs, une équidistribution maximale (et même la propriété «sans-collision») a pu être obtenue.

6.3.3 Générateurs inversifs

Nous avons décidé de considérer la forme la plus simple de générateurs inversifs, soit les générateurs inversifs explicites. Nous avons retenu ceux décrits dans [22]. Les modules m sont premiers, et identiques aux modules des LCG testés. Quant aux paramètres a et c, ils ont été fixés à 123 et 0 pour tous les générateurs. Rappelons que la période de ces générateurs est toujours de m, peu importe le choix des paramètres. Ces générateurs, souvent désignés par «InvExpl», ont été testés à plusieurs reprises [27, 30, 31], et ont semblé se comporter de façon similaire aux autres types de générateurs inversifs, lorsqu'ils ont été comparés.

6.3.4 Générateurs à congruence cubique

Les générateurs cubiques que nous avons utilisés ont deux composantes de la forme (3.3), avec des modules premiers. La combinaison a été faite en additionnant les sorties modulo 1, comme expliqué à la section 4.3. Les paramètres utilisés sont ceux de [22], et permettent d'obtenir des périodes près de 2^e, pour des entiers e allant de 12 à 36. Ces générateurs ont aussi été testés à plusieurs reprises [27, 30], et sont parfois désignés par «CombCubic2».

Tableau 6.1: Tailles approximatives des générateurs non linéaires et des LCG utilisés dans les combinaisons, pour donner un générateur de période $\rho \approx 2^e$

Famille	Taille du LCG	Taille du générateur non linéaire
1	$2^{3e/4}$	$2^{e/4}$
2	$2^{e/2}$	$2^{e/2}$
3	2^{e-6}	2^6
4	2^{e-7}	2^7
5	2^{e-8}	2^8
6	2^{e-9}	2^9
7	2^{e-10}	2^{10}

6.3.5 Combinaison de MRG avec d'autres générateurs

Des LCG ont été combinés avec des générateurs non linéaires de type inversifs explicites et des générateurs cubiques. La combinaison a été faite en additionnant les sorties modulo 1, comme suggéré à la section 5.2.

Plusieurs tailles de générateurs non linéaires ont été essayées, pour former les familles illustrées au tableau 6.1 (une famille étant définie par la période de chacune de ses composantes en fonction de e, où 2^e est la période du générateur combiné). D'abord, des petits générateurs inversifs ou cubiques de période autour de $2^{e/4}$ ont été combinés avec des LCG de période près de $2^{3e/4}$, pour former des générateurs de période approchant 2^e. Des générateurs combinés de période près de 2^e ont aussi été obtenus à partir de générateurs inversifs ou cubiques et de LCG ayant chacun une période approximative de $2^{e/2}$. Ces familles ont permis de voir le comportement de générateurs combinés dont la partie non linéaire croît avec la valeur de e.

Nous avons aussi fixé la période du générateur non linéaire, et fait varier uniquement la taille des LCG pour obtenir un générateur combiné de période 2^e. Nous avons choisi de petits générateurs non linéaires de période allant de 2^6 à 2^{10}, pour vérifier à

partir de quelle taille ils commencent à influencer les résultats. Nous avons aussi voulu comparer les performances des combinaisons, en fonction des périodes des générateurs non linéaires utilisés.

Quelques combinaisons avec des LFSR ont aussi été essayées, pour voir si la non-linéarité du deuxième générateur est nécessaire pour l'obtention de bons résultats, ou si un deuxième générateur linéaire (autre qu'un MRG) peut aussi faire l'affaire. Cependant, seules les familles 2 et 7 du tableau 6.1 ont été formées (les LFSR remplaçant les générateurs non linéaires), puisque l'étude de ces générateurs n'est pas le but premier de ce mémoire.

Il reste maintenant un point à préciser : le choix des paramètres des générateurs LCG, LFSR, inversifs et cubiques utilisés. Une fois les paramètres des LCG fixés, il aurait été intéressant de sélectionner les autres générateurs selon le critère 5.2.1, développé au chapitre précédent. Cependant, comme nous l'avons mentionné précédement, nous n'avons pas trouvé de moyen simple d'appliquer ce critère, en pratique. Nous avons donc décidé de choisir de «bons» paramètres pour chaque générateur pris individuellement, tout en s'assurant d'obtenir une période maximale égale au produit des périodes de chaque composante, lors des différentes combinaisons. Comme les générateurs LFSR ou non linéaires qui sont combinés sont généralement petits par rapport aux LCG, une assez bonne uniformité devrait être conservée suite aux combinaisons.

Les LCG utilisés pour former les générateurs combinés ayant une composante non linéaire sont ceux décrits à la section 6.3.1. Des LCG différents ont dû être utilisés avec les LFSR, pour permettre d'atteindre les périodes désirées. En fait, certaines des périodes des LCG de la section 6.3.1 avaient des multiples communs avec les périodes des LFSR décrits à la section 6.3.2, réduisant de façon significative la période de plusieurs générateurs combinés. Nous avons donc décidé de conserver les LFSR décrits, et de faire une recherche de nouveaux LCG. Les modules m ont été choisis comme étant les plus grands nombres premiers inférieurs à 2^e pour lesquels $(m-1)/2$ est aussi premier, pour e allant de 10 à 31. Ce choix nous assure que la période maximale $m-1$ des LCG ne possède que 2 et un autre nombre premier comme facteurs, réduisant les risques de

communs multiples avec les périodes des LFSR. Les paramètres a_1 retenus sont ceux ayant donnés la meilleure valeur de M_8 au test spectral, pour les modules m donnés, les conditions de périodes maximales vérifiées et la restriction $a_1 m < 2^{53}$ nécessaire pour l'implantation en virgule flottante. Les paramètres sélectionnés ont permis d'obtenir des périodes maximales avec toutes les combinaisons impliquant un LFSR.

Les générateurs cubiques utilisés dans les combinaisons sont non combinés, et leurs paramètres ont été choisis parmi les composantes décrites à la section 6.3.4. Quant aux générateurs inversifs, leurs modules sont les mêmes que ceux des générateurs cubiques utilisés, et les paramètres a et c ont encore été fixés à 123 et 0. Notons que nous avons dû changer les modules des générateurs inversifs, par rapport à ceux décrits à la section 6.3.3. En prenant les mêmes modules que les LCG (comme à la section 6.3.3), de bonnes périodes auraient pu être atteintes, mais de grandes erreurs de discrétisation auraient pu fausser les résultats dans certains cas. Par exemple, dans le cas de la famille 2 du tableau 6.1, lorsque e est pair, des LCG et des générateurs inversifs de même taille sont combinés. Si ces générateurs ont le même module m, les sorties produites ne sont que des multiples de $1/m$, alors que si leurs modules diffèrent légèrement, la précision des sorties peut augmenter jusqu'à environ $1/m^2$.

6.3.6 Combinaison de LFSR avec d'autres générateurs

Les LFSR de la section 6.3.2 ont été combinés avec des LCG, des générateurs inversifs et des générateurs cubiques, au moyen d'un ou-exclusif bit à bit entre les sorties, comme suggéré à la section 5.3. Les tailles des générateurs non linéaires considérées sont les mêmes que dans le cas des LCG combinés de la section précédente (voir le tableau 6.1), et les paramètres sélectionnés sont semblables. En ce qui concerne les combinaisons avec des LCG, les familles 2 et 7 du tableau 6.1 ont été formées, et les paramètres des LCG sont ceux de la section 6.3.5. Les combinaisons ont été faites avec les 31 bits les plus significatifs des sorties de chaque générateur.

6.4 Résultats

Les résultats obtenus aux différents tests sont présentés dans ce qui suit. Nous avons soumis plusieurs générateurs d'une même famille à chaque test, et nous avons cherché à déterminer une relation entre la période des générateurs et le nombre minimal de vecteurs $\mathbf{u}_{ti,t}$ devant être générés avant que le test n'échoue. Pour ce faire, nous avons suivi la méthode utilisée dans [27, 30, 31]. Nous avons cherché à déterminer une règle de la forme $\tilde{n} \approx 2^{\tilde{\nu}+\tilde{\gamma}e}$, où $\tilde{n}$ est le nombre de vecteurs $\mathbf{u}_{ti,t}$ devant être produits avant que les générateurs de période 2^e d'une même famille ne commencent à échouer un test. Une fois spécifiée, cette règle indique qu'un nombre de vecteurs générés proportionnel à la racine $\tilde{\gamma}$ième de la période commence à poser des difficultés, et la constante de proportionnalité est déterminée par $\tilde{\nu}$. Il est à noter qu'une grande valeur de $\tilde{\gamma}$ est particulièrement souhaitée, puisque l'impact de cette constante devient de plus en plus important à mesure que la période des générateurs croît. La constante de proportionnalité $\tilde{\nu}$ joue un rôle plus secondaire, mais peut devenir utile pour comparer des générateurs ayant une valeur de $\tilde{\gamma}$ semblable.

Pour trouver les valeurs de $\tilde{\gamma}$ et $\tilde{\nu}$, nous avons appliqué les tests sur des échantillons de taille $n = 2^{\nu+\gamma e}$, pour différents choix de γ et ν. Nous avons cherché à identifier la valeur de γ pour laquelle la plupart des générateurs commencent à échouer le test pour une même valeur de ν. Il s'agit, bien sûr, d'une procédure heuristique.

Cette méthode a été utilisée à plusieurs reprises [27, 30, 31], et des constantes $\tilde{\gamma}$ et $\tilde{\nu}$ ont déjà été obtenues avec quelques familles de générateurs, pour certains des tests retenus. Entre autres, plusieurs résultats sont connus pour les LCG, les LFSR et quelques générateurs non linéaires. Les résultats présentés ici permettent de compléter les données disponibles, en ajoutant des informations sur plusieurs familles de générateurs combinés jamais testées jusqu'à maintenant.

Les différents tests statistiques ont été appliqués à l'aide du logiciel «TestU01» [22], développé par une équipe du laboratoire de simulation de l'Université de Montréal. Certains modules ont dû être ajoutés au logiciel déjà existant, pour compléter les familles

de générateurs déjà disponibles. Le guide d'utilisation de ces modules est donné à l'annexe B.

6.4.1 Test des points les plus rapprochés

Le test des points les plus rapprochés a été appliqué en dimensions $t = 2$ et $t = 8$, avec une norme L_∞. Dans un premier temps, les deux points les plus près ont été considérés ($M = 1$), puis les 32 paires de points les plus proches ont été retenues ($M = 32$). Dans tous les cas, le test n'a été appliqué qu'une seule fois ($N = 1$).

Exemple de détermination d'une règle

Voici un exemple illustrant les étapes suivies pour déterminer une règle de la forme $\tilde{n} \approx 2^{\tilde{\nu}+\tilde{\gamma}e}$. La famille considérée est celle des LCG, et le test appliqué correspond au cas où $M = 1$ (test NP), en dimension $t = 2$. Les résultats de ce test sont déjà connus [27], et nous avons tenté de les reproduire dans ce qui suit.

Pour une valeur de γ fixée, nous avons produit des tableaux donnant les p-valeurs du test en fonction de la période des générateurs, pour différents ν entiers consécutifs. Par exemple, le tableau 6.2 donne les p-valeurs obtenues dans le cas où $\gamma = 1/3$ et $\nu = 3, ..., 8$. Seules les p-valeurs jugées suspectes sont indiquées, soient celles inférieures à 0.01 ou dépassant 0.99. Les p-valeurs sont imprimées en notation scientifique, $x\mathrm{E}{-}y$ signifiant $x \times 10^{-y}$. Le symbole ϵ, quant à lui, désigne une p-valeur inférieure à 10^{-15}. Nous pouvons voir qu'avec ce choix de γ, les p-valeurs suspectes commencent à apparaître pour différents ν. Il n'est donc pas possible de trouver un $\tilde{\nu}$ tel que la taille minimale de l'échantillon à partir de laquelle les LCG échouent le test puisse être approximée par $\tilde{n} \approx 2^{\tilde{\nu}+e/3}$. Nous pouvons cependant constater que plus la période des générateurs augmente, plus les p-valeurs suspectes surviennent tard (c'est-à-dire pour des plus grandes valeurs de ν). Ceci nous indique que la valeur de $\gamma = 1/3$ utilisée est trop petite.

Les résultats de ce même test, pour une valeur de $\gamma = 2/3$, sont donnés au ta-

Tableau 6.2: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 1/3$, pour les LCG

e	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$	$\nu = 8$
14		1E-4	ϵ	ϵ	ϵ	ϵ
15		2E-5	ϵ	ϵ	ϵ	ϵ
16		4E-4	3E-14	ϵ	ϵ	ϵ
17		5E-3	6E-10	ϵ	ϵ	ϵ
18			7E-6	ϵ	ϵ	ϵ
19		1E-2	9E-9	ϵ	ϵ	ϵ
20			1E-7	ϵ	ϵ	ϵ
21			1E-4	ϵ	ϵ	ϵ
22			6E-4	1E-13	ϵ	ϵ
23			3E-4	1E-14	ϵ	ϵ
24			7E-4	2E-13	ϵ	ϵ
25				6E-7	ϵ	ϵ
26				3E-7	ϵ	ϵ
27				8E-5	ϵ	ϵ
28				2E-3	8E-12	ϵ
29				1E-4	ϵ	ϵ
30					2E-7	ϵ
31				5E-3	5E-10	ϵ
32	1E-3				3E-5	ϵ
33					1E-3	4E-12
34					1E-4	ϵ
35					2E-3	3E-11
36					7E-3	3E-9

bleau 6.3. Le symbole – indique que le test n'a pas pu être appliqué, parce qu'une trop grande quantité de mémoire aurait été nécessaire. Encore une fois, les p-valeurs suspectes commencent à apparaître pour différents ν. Mais cette fois, plus la période des générateurs augmente, plus les p-valeurs suspectes surviennent tôt, ce qui suggère que le γ utilisé est maintenant trop grand.

Le tableau 6.4 donne les résultats obtenus avec une valeur de $\gamma = 1/2$ intermédiaire. Cette fois, les générateurs se comportent de façon similaire, peu importe leur période 2^e. Nous venons donc de trouver la valeur de $\tilde{\gamma}$. En ce qui concerne la valeur de $\tilde{\nu}$, nous avons choisi de prendre la même convention que dans [27, 30, 31], soit le plus petit ν pour lequel la majorité des générateurs ont des p-valeurs suspectes. Nous avons donc $\tilde{\nu} = 1$ dans ce cas-ci. Les LCG sélectionnés commencent donc à échouer le test lorsque

Tableau 6.3: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 2/3$, pour les LCG

e	$\nu = -5$	$\nu = -4$	$\nu = -3$	$\nu = -2$	$\nu = -1$	$\nu = 0$
14					3E-3	1E-10
15					2E-5	ϵ
16					5E-6	ϵ
17					2E-6	ϵ
18					7E-6	ϵ
19				6E-4	2E-13	ϵ
20				4E-5	ϵ	ϵ
21				1E-4	ϵ	ϵ
22				8E-6	ϵ	ϵ
23			6E-3	2E-9	ϵ	ϵ
24			7E-4	2E-13	ϵ	ϵ
25			3E-3	1E-10	ϵ	ϵ
26	5E-3		8E-5	ϵ	ϵ	ϵ
27			8E-5	ϵ	ϵ	ϵ
28			4E-5	ϵ	ϵ	ϵ
29		3E-3	8E-11	ϵ	ϵ	ϵ
30			2E-7	ϵ	ϵ	ϵ
31		2E-4	2E-15	ϵ	ϵ	ϵ
32		1E-3	4E-12	ϵ	ϵ	—
33		1E-3	4E-12	ϵ	ϵ	—
34		4E-7	ϵ	ϵ	—	—
35		2E-7	ϵ	—	—	—
36	7E-3	3E-9	ϵ	—	—	—

la taille échantillonnale est donnée par $n \approx 2^{1+e/2}$. Ceci suggère que pour des LCG comparables à ceux que nous avons choisis, il n'est pas prudent de générer plus de deux fois la racine carrée de la période.

Nous avons aussi décidé, tout comme dans [27, 30, 31], de retenir le plus petit ν pour lequel une majorité de ϵ sont observés, que nous noterons par ν^*. Cette valeur nous permet de connaître la taille échantillonnale à partir de laquelle une famille échoue incontestablement un test, avec des p-valeurs inférieures à 10^{-15}. Dans cet exemple-ci, nous avons $\nu^* = 3$, et les LCG sélectionnés échouent sans le moindre doute le test lorsque la taille échantillonnale est de $n \approx 2^{3+e/2}$.

Tableau 6.4: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 1/2$, pour les LCG

e	$\nu = -1$	$\nu = 0$	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$
14				5E-7	ϵ	ϵ
15	2E-4		4E-3	4E-10	ϵ	ϵ
16			8E-3	3E-9	ϵ	ϵ
17				5E-8	ϵ	ϵ
18				7E-6	ϵ	ϵ
19			3E-3	7E-11	ϵ	ϵ
20			2E-3	1E-11	ϵ	ϵ
21				2E-8	ϵ	ϵ
22			9E-3	8E-9	ϵ	ϵ
23			2E-3	9E-12	ϵ	ϵ
24			7E-4	2E-13	ϵ	ϵ
25				1E-8	ϵ	ϵ
26			3E-3	5E-11	ϵ	ϵ
27			9E-3	6E-9	ϵ	ϵ
28				1E-7	ϵ	ϵ
29			7E-4	2E-13	ϵ	ϵ
30				2E-7	ϵ	ϵ
31			1E-3	2E-12	ϵ	ϵ
32				7E-8	ϵ	ϵ
33				2E-6	ϵ	ϵ
34			3E-3	8E-11	ϵ	ϵ
35			8E-3	4E-9	ϵ	ϵ
36			7E-3	3E-9	ϵ	ϵ

Nous avons répété ce processus pour chaque famille de générateurs décrite, et pour les différents paramètres du test mentionnés.

Résultats attendus

Le test des points les plus rapprochés a déjà été effectué avec des LCG, des générateurs inversifs et des générateurs cubiques [27].

Les LCG commencent à échouer le test très tôt, c'est-à-dire pour une très petite valeur de $\tilde{\gamma}$. Comme mentionné à la section 6.2.1, lorsque $M = 1$, les MRG ayant bien réussi au test spectral commencent à échouer le test parce que la distance séparant les deux points les plus rapprochés est trop grande, alors que dans le cas où $M = 32$, des

distances trop régulières sont mesurées entre les paires de points les plus rapprochés, nous forçant à rejeter l'hypothèse $\mathcal{H}_0$. Une règle peut alors être déduite, comme nous l'avons illustré à la section précédente, parce que la même déviation par rapport à une vraie séquence de nombres aléatoires est systématiquement détectée, pour chaque générateur de la famille.

Quant aux générateurs non linéaires, ils se comportent très bien par rapport à ce test. Les points de ces générateurs ne forment plus une structure de réseau, et peuvent être séparés par des distances très variables les uns des autres. Aussi, la période doit souvent être dépassée avant que des problèmes d'uniformité ne soient détectés. Dans ce cas, des points identiques commencent à être générés, et la distance minimale entre les paires de points les plus proches devient nulle. La p-valeur obtenue est alors presque 1, et l'hypothèse $\mathcal{H}_0$ doit être rejettée.

Regardons maintenant ce à quoi nous pouvons nous attendre, pour les nouvelles familles de générateurs testées.

D'abord, les LCG combinés à un autre générateur par une addition modulo 1 produisent des points tirés d'une superposition de réseaux translatés. Une forte structure semblable à celle des LCG est encore présente, nous laissant croire qu'une règle systématique a de fortes chances de pouvoir être trouvée dans ce cas. Aussi, comme les distances séparant les points produits sont plus variables avec ces LCG combinés, dû à la superposition de plusieurs réseaux translatés, nous pensons devoir générer plus de points avant que les tests ne détectent la structure des vecteurs produits. Selon ce même raisonnement, nous nous attendons à ce qu'un deuxième générateur à combiner de taille plus importante (impliquant une plus grande quantité de translations du réseau de base) permette d'améliorer la performance des LCG combinés à ces tests.

En ce qui concerne les autres familles de générateurs (LFSR et LFSR combinés à un autre générateur à l'aide d'un ou-exclusif), aucune structure de réseau n'est présente. Les points risquent d'être séparés par des distances plus variables que celles observées pour les LCG, mais moins qu'avec les générateurs non linéaires, puisqu'une forte structure due aux bonnes propriétés d'équidistribution recherchées est présente. Les tests

Tableau 6.5: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 1/2$, pour les LCG combinés avec un inversif de période 2^7

e	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
17			7E-3	3E-9	ϵ	ϵ
18				4E-4	2E-14	ϵ
19				3E-4	9E-15	ϵ
20					5E-7	ϵ
21					3E-8	ϵ
22				2E-3	6E-12	ϵ
23				9E-6	ϵ	ϵ
24			2E-3	2E-11	ϵ	ϵ
25			5E-4	5E-14	ϵ	ϵ
26				1E-3	8E-13	ϵ
27				5E-4	8E-14	ϵ
28				7E-4	3E-13	ϵ
29				2E-8	ϵ	ϵ
30				5E-8	ϵ	ϵ
31				2E-4	ϵ	ϵ
32			5E-3	1E-7	ϵ	ϵ
33				4E-6	ϵ	—
34				7E-6	ϵ	—
35			1E-4	ϵ	—	—
36				8E-5	—	—

devraient donc échouer avant que la période ne soit générée en entier, mais avec une taille échantillonnale plus grande que celle trouvée pour les LCG. Aussi, comme la structure des points et les distances mesurées entre les paires de points risquent de varier beaucoup d'un générateur à l'autre d'une même famille, il est peu probable qu'une règle systématique ne puisse être observée.

Test NP

Voici maintenant les résultats obtenus au test NP. Les tableaux 6.5 à 6.8 donnent des exemples de p-valeurs produites en dimension $t = 2$. Les tableaux 6.5 et 6.6 montrent qu'une règle a pu être trouvée pour les LCG combinés à un générateur inversif ou cubique de période 2^7. En fait, comme prévu, des règles ont été trouvées pour toutes

Tableau 6.6: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 1/2$, pour les LCG combinés avec un cubique de période 2^7

e	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
17					1E-6	ϵ
18				1E-5	ϵ	ϵ
19			4E-3	2E-10	ϵ	ϵ
20					5E-7	ϵ
21					3E-8	ϵ
22				1E-3	4E-12	ϵ
23			8E-3	4E-9	ϵ	ϵ
24				4E-6	ϵ	ϵ
25					8E-7	ϵ
26				1E-3	8E-13	ϵ
27				5E-4	8E-14	ϵ
28				7E-4	3E-13	ϵ
29			4E-3	3E-10	ϵ	ϵ
30				5E-4	8E-14	ϵ
31				2E-4	ϵ	ϵ
32				1E-7	ϵ	ϵ
33				8E-4	5E-13	—
34				4E-3	2E-10	—
35			5E-4	1E-4	—	—
36				2E-4	—	—

les familles de LCG combinés à un générateur non linéaire par une addition modulo 1. Dans les deux cas illustrés, le nombre de vecteurs devant être produits avant que le test n'échoue est de $n \approx 2^{4+e/2}$. L'ajout d'une petite composante non linéaire aussi petite que 2^7, qu'elle soit cubique ou inversive, permet donc de générer environ huit fois plus de nombres qu'avec la famille des LCG, avant que des p-valeurs suspectes ne commencent à se produire.

Le tableau 6.7 donne les p-valeurs obtenues pour la famille des LCG combinés à un LFSR de période 2^{10}. Une p-valeur négative ($-\epsilon$) peut être observée dans ce tableau. En fait, tout comme dans [27, 30, 31], nous avons choisi de représenter par $-x$ une p-valeur de $1 - x$ trop près de 1. Le symbole $-\epsilon$ est donc utilisé pour indiquer qu'une p-valeur supérieure à $1 - 10^{-15}$ a été obtenue. Une règle se dégage encore de ce tableau, mais de façon un peu moins précise que dans les deux cas précédents. En fait, le générateur

Tableau 6.7: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 1/2$, pour les LCG combinés avec un LFSR de période 2^{10}

e	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$	$\nu = 8$	$\nu = 9$
20	6E-3	1E-6	ϵ	ϵ	ϵ	$-\epsilon$
21			7E-3	2E-9	ϵ	ϵ
22			1E-3	5E-12	ϵ	ϵ
23				5E-4	6E-14	ϵ
24			5E-3	8E-10	ϵ	ϵ
25			5E-5	ϵ	ϵ	ϵ
26		5E-3	2E-3	2E-11	ϵ	ϵ
27		7E-4	3E-13	ϵ	ϵ	—
28			8E-5	ϵ	ϵ	—
29			7E-4	3E-13	—	—
30			3E-6	ϵ	—	—
31				—	—	—
32				—	—	—
33			—	—	—	—
34		8E-3	—	—	—	—

de période 2^{20} commence à échouer le test plus rapidement que les autres générateurs de sa famille. Un comportement similaire a été observé pour les LCG combinés aux LFSR de période $2^{e/2}$. La règle obtenue a tout de même été notée, mais il est bon de se souvenir que quelques générateurs peuvent dévier légèrement de cette règle.

Le tableau 6.8 illustre le comportement des LFSR. Comme prévu, ces générateurs échouent le test avec une plus grande taille échantillonnale que les LCG, mais avant que leur période ne soit épuisée. Aussi, aucune règle claire ne peut être déduite. Avec le γ illustré, les valeurs de ν à partir desquelles les générateurs commencent à échouer le test, ou encore à obtenir des p-valeurs inférieures à 10^{-15} ou supérieures à $1 - 10^{-15}$ varient beaucoup d'un générateur à l'autre. Des variations encore plus prononcées ont été observées avec les autres valeurs de γ essayées. Un comportement similaire a été obtenu avec toutes les familles de LFSR combinés avec un autre générateur, et aucune relation claire du type $\tilde{n} \approx 2^{\tilde{\nu}+\tilde{\gamma}e}$ n'a pu être déduite.

Le tableau 6.9 résume les différents résultats obtenus à ce test NP en dimension $t = 2$. Seules les familles pour lesquelles une règle a pu être établie sont présentées.

Tableau 6.8: p-valeurs du test NP en dimension $t = 2$ avec $\gamma = 1/2$, pour les LFSR

e	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$
12			1E-3	5E-4	1E-13	$-\epsilon$	$-\epsilon$
13		2E-3			1E-7	ϵ	$-\epsilon$
14				3E-6	9E-7	ϵ	$-\epsilon$
15			9E-7	ϵ	ϵ	ϵ	ϵ
16		3E-5	6E-5	2E-12	ϵ	ϵ	ϵ
17		3E-4	9E-8	3E-15	ϵ	ϵ	ϵ
18			4E-3	2E-10	ϵ	ϵ	ϵ
19	6E-3						2E-7
20		4E-6	2E-11	1E-3	5E-4	5E-14	ϵ
21		2E-4	ϵ	ϵ	ϵ	ϵ	ϵ
22		9E-3	5E-8	ϵ	ϵ	ϵ	ϵ
23		2E-7	4E-3	4E-5	ϵ	ϵ	ϵ
24			2E-3	3E-4	ϵ	ϵ	ϵ
25		1E-7	1E-13	ϵ	ϵ	ϵ	ϵ
26			2E-3	6E-11	ϵ	ϵ	ϵ
27					6E-7	ϵ	ϵ
28			1E-4	4E-7	7E-7	ϵ	ϵ
29		5E-3	2E-4	2E-4	2E-15	ϵ	—
30			4E-3		3E-6	ϵ	—
31				5E-4	4E-7	—	—
32				3E-3	1E-11	—	—
33				4E-6	—	—	—
34		3E-4	1E-14	ϵ	—	—	—

Le symbole + entre deux types de générateurs différents indique que leurs sorties sont combinées avec une addition modulo 1, et la valeur de e_2 indiquée entre parenthèses correspond au $\log_2$ de la période du deuxième générateur. À titre d'exemple, les LCG combinés avec un deuxième générateur cubique de période 2^7 dont nous avons discuté précédemment sont désignés par «LCG + Cub ($e_2 = 7$)». Un symbole * a aussi été ajouté à la suite du $\tilde{\gamma}$ des LCG combinés aux LFSR, pour rappeler qu'une règle un peu moins précise a été obtenue.

Nous pouvons constater que les générateurs non linéaires se comportent de façon très similaire, même lorsque combinés avec les LCG. Aussi, l'ajout d'un petit générateur non linéaire constant aux LCG ne permet pas d'améliorer la constante $\tilde{\gamma}$, mais augmente toutefois la constante multiplicative $2^{\tilde{\nu}}$. Comme prévu, cette constante multiplicative

Tableau 6.9: Valeurs de $\tilde{\nu}$, ν^* et $\tilde{\gamma}$ pour le test NP en dimension $t = 2$

Famille	$\tilde{\gamma}$	$\tilde{\nu}$	ν^*
LCG	1/2	1	3
Inversifs explicites	1	0	0
Cubiques	1	0	0
LCG + Inv ($e_2 = 6$)	1/2	4	5
LCG + Inv ($e_2 = 7$)	1/2	4	5
LCG + Inv ($e_2 = 8$)	1/2	5	6
LCG + Inv ($e_2 = 9$)	1/2	6	7
LCG + Inv ($e_2 = 10$)	1/2	6	8
LCG + Inv ($e_2 = e/4$)	2/3	-1	1
LCG + Inv ($e_2 = e/2$)	2/3	3	5
LCG + Cub ($e_2 = 6$)	1/2	4	5
LCG + Cub ($e_2 = 7$)	1/2	4	6
LCG + Cub ($e_2 = 8$)	1/2	5	7
LCG + Cub ($e_2 = 9$)	1/2	6	7
LCG + Cub ($e_2 = 10$)	1/2	6	8
LCG + Cub ($e_2 = e/4$)	2/3	-1	1
LCG + Cub ($e_2 = e/2$)	2/3	3	5
LCG + LFSR ($e_2 = 10$)	1/2*	6	8
LCG + LFSR ($e_2 = e/2$)	2/3*	3	4

augmente avec la taille du générateur non linéaire. Ainsi, l'ajout d'un petit générateur cubique ou inversif de période 2^6 permet de générer $2^3 = 8$ fois plus de nombres que pour les LCG seuls, alors qu'un générateur non linéaire de période 2^{10} permet d'améliorer la performance de $2^5 = 32$ fois.

Si nous regardons plutôt les générateurs combinés dont la deuxième composante est de taille $2^{e/4}$ ou $2^{e/2}$, nous pouvons constater que la constante $\tilde{\gamma}$ peut être améliorée. Ainsi, le nombre de vecteurs pouvant être produits sans faire échouer le test devient proportionnel à la racine 2/3 de la période, plutôt que la racine carrée. Pour des générateurs de plus longues périodes, ce gain peut devenir très intéressant.

Soulignons aussi le fait que les combinaisons avec des LFSR ont donné des résultats

très semblables aux combinaisons équivalentes avec les générateurs non linéaires. Seuls quelques générateurs ont dévié légèrement des règles indiquées. En fait, il semble que ce soit le fait de superposer plusieurs réseaux translatés de façon plus ou moins aléatoire qui influence les résultats du test NP, plutôt que la manière de translater les réseaux. Les translations aléatoires effectuées permettent de donner une structure moins régulière aux points, et d'augmenter les chances d'obtenir une petite distance entre les deux points les plus rapprochés.

Le test NP a aussi été appliqué en dimension $t = 8$. Cependant, des tailles échantillonnales beaucoup plus grandes ont dû être utilisées avant que le test ne commence à échouer. Comme la quantité de mémoire requise par ce test est très importante (particulièrement en dimension $t = 8$), seuls les plus petits générateurs ont pu être observés. Face à la quantité limitée de résultats que nous avons pu obtenir, nous avons préféré ne pas tirer de conclusions pour ce test.

Test 32-NP

Nous avons aussi effectué le test en considérant les 32 paires de points les plus rapprochés. Les résultats obtenus sont résumés au tableau 6.10. Un symbole * a parfois été ajouté à la suite de la dimension du test, dans les cas où la règle trouvée est un peu moins précise.

Nous pouvons constater que ce test 32-NP est plus discriminant que le test avec $M = 1$. Par exemple, en dimension $t = 2$, les LCG commencent à échouer quatre fois plus tôt, les générateurs cubiques 32 fois plus tôt, et les LCG combinés à un générateur cubique de période 2^6 échouent ce test huit fois plus rapidement. Il nous a donc été possible d'obtenir des résultats plus concluants en dimension $t = 8$. Cependant, le temps d'exécution du test étant assez élevé, nous avons restreint notre étude en haute dimension à quelques familles seulement. En général, lorsqu'une règle claire a été trouvée en dimension $t = 2$, une règle similaire a été obtenue en dimension $t = 8$. Les générateurs non linéaires font toutefois exceptions, puisqu'ils ont échoué le test beaucoup plus rapidement en dimension $t = 2$.

Tableau 6.10: Valeurs de $\tilde{\nu}$, ν^* et $\tilde{\gamma}$ pour le test 32-NP en dimensions $t = 2$ et $t = 8$

Famille	t	$\tilde{\gamma}$	$\tilde{\nu}$	ν^*
LCG	2	1/2	-1	0
	8	1/2	-1	0
Inversifs explicites	2*	1	-6	-4
	8*	1	-1	0
Cubiques	2*	1	-6	-4
	8	1	0	0
LCG + Inv ($e_2 = 6$)	2	1/2	1	2
	8	1/2	2	2
LCG + Inv ($e_2 = 7$)	2	1/2	1	2
LCG + Inv ($e_2 = 8$)	2	1/2	2	3
LCG + Inv ($e_2 = 9$)	2	1/2	3	4
LCG + Inv ($e_2 = 10$)	2	1/2	4	5
	8	1/2	4	5
LCG + Inv ($e_2 = e/4$)	2	2/3	-3	-2
	8	2/3	-3	-2
LCG + Inv ($e_2 = e/2$)	2	3/4	-1	-1
	8	2/3	1	2
LCG + Cub ($e_2 = 6$)	2	1/2	1	2
	8	1/2	1	2
LCG + Cub ($e_2 = 7$)	2	1/2	1	2
LCG + Cub ($e_2 = 8$)	2	1/2	2	3
LCG + Cub ($e_2 = 9$)	2	1/2	3	4
LCG + Cub ($e_2 = 10$)	2	1/2	4	4
	8	1/2	4	5
LCG + Cub ($e_2 = e/4$)	2	2/3	-3	-2
	8	2/3	-3	-2
LCG + Cub ($e_2 = e/2$)	2	3/4	-2	-1
	8	3/4	-2	0
LCG + LFSR ($e_2 = 10$)	2	1/2	4	5
	8	1/2	4	5
LCG + LFSR ($e_2 = e/2$)	2	3/4	-1	0
	8	3/4	-1	0

Plusieurs conclusions semblables à celles discutées pour le test NP s'appliquent encore pour ce test 32-NP. D'abord, les LCG commencent à échouer le test lorsqu'un nombre de vecteurs proportionnel à la racine carrée de la période est généré, alors que

Tableau 6.11: p-valeurs du test 32-NP en dimension $t = 2$ avec $\gamma = 1$, pour les générateurs cubiques

e	$\nu = -7$	$\nu = -6$	$\nu = -5$	$\nu = -4$	$\nu = -3$
18	—	3E-4	5E-3	ϵ	ϵ
19			5E-4	ϵ	ϵ
20	4E-3		ϵ	ϵ	ϵ
21	7E-4	2E-4	6E-6	ϵ	ϵ
22		2E-4	2E-7	ϵ	ϵ
23		ϵ	ϵ	4E-13	ϵ
24		8E-3	ϵ	ϵ	ϵ
25				ϵ	—
26			1E-3	—	—
27		2E-3	—	—	—
28		—	—	—	—

la presque totalité de la période des générateurs non linéaires peut être générée sans problème. Ensuite, les générateurs cubiques et inversifs se comportent de façon similaire, même lorsque combinés avec un LCG. Aussi, l'ajout d'un petit générateur non linéaire constant aux LCG permet de générer entre quatre et 32 fois plus de nombres avant que le test n'échoue, pour des tailles de générateurs non linéaires variant entre 2^6 et 2^{10}. Lorsque la période des générateurs non linéaires augmente avec celle du générateur combiné, la constante $\tilde{\gamma}$ augmente en plus. Il devient alors possible de générer une quantité de vecteurs proportionnelle à la racine 2/3 ou 3/4 de la période.

Une petite différence par rapport au test NP peut toutefois être notée. Dans ce cas-ci, la combinaison avec les LFSR a donné des règles aussi claires que les combinaisons avec des générateurs non linéaires. Ce sont plutôt les générateurs non linéaires qui ont produit des résultats moins systématiques. À titre d'exemple, le tableau 6.11 donne les p-valeurs obtenues pour les générateurs cubiques, en dimension $t = 2$.

Notons finalement que, comme prévu, les LFSR et les LFSR combinés à un autre générateur avec un ou-exclusif n'ont pas produit de résultats systématiques. Aucune règle n'a donc pu être trouvée pour ces familles.

6.4.2 Tests sériels

Les tests du khi-deux (X^2) et de collisions (C) ont été effectués en dimensions $t = 2$ et $t = 6$. Le test le plus costaud a été considéré, soit le cas clairsemé avec un nombre de boîtes $h = d^t$ approximativement égal à la période des générateurs. Comme suggéré à la section 6.2.2, deux cas ont été distingués pour les LFSR ou les LFSR combinés à un autre générateur par un ou-exclusif : celui où d est choisi impair, et celui où il est pris comme la plus petite puissance de 2 dépassant la période des générateurs testés. Aussi, les tests n'ont été appliqués qu'une seule fois ($N = 1$).

Résultats attendus

Des tests sériels ont déjà été effectués avec des LCG, des LFSR et des générateurs non linéaires [27, 31]. Pour toutes ces familles, des règles de la forme $\tilde{n} \approx 2^{\tilde{\nu}+\tilde{\gamma}e}$ ont été trouvées. Comme mentionné à la section 6.2.2, les LCG ayant une bonne valeur au test spectral et les LFSR maximalement équidistribués échouent les tests assez rapidement à cause d'une trop grande uniformité des points produits. Les générateurs non linéaires échouent beaucoup plus tard, souvent après avoir épuisé leur période entière. Dans ce cas, un manque d'uniformité finit par être détecté, parce que des points identiques à d'autres déjà produits commencent à être générés.

En ce qui concerne les générateurs combinant des familles différentes, la théorie développée au chapitre précédent nous assure une relativement bonne uniformité des vecteurs produits, lorsque les points des générateurs linéaires de base (LCG ou LFSR) sont eux-mêmes bien distribués. Cependant, le degré d'uniformité obtenu peut varier d'un générateur à l'autre. La répartition des points produits par un LCG combiné à un autre générateur par une addition modulo 1 dépend des translations effectuées sur le réseau de base, alors que le niveau d'équidistribution obtenu avec les LFSR combinés à un autre générateur avec un ou-exclusif dépend d'une interaction complexe entre les bits des nombres produits par les deux générateurs impliqués. La variation possible du niveau d'uniformité présent au sein des générateurs d'une même famille risque de

rendre plus difficile l'analyse des résultats, nous empêchant peut-être de découvrir une règle s'appliquant à toute la famille. Nous nous attendons toutefois à ce que pris individuellement, les générateurs combinés performent aussi bien ou mieux que les LCG ou les LFSR seuls de période équivalente, à cause de l'uniformité moins excessive de leurs points.

Test du khi-deux

Comme prévu, très peu de règles ont pu être établies avec le test du khi-deux. En fait, tout comme dans [27, 31], des règles claires ont été trouvées pour les LCG, les LFSR et les générateurs non linéaires, mais aucune règle n'a pu être observée pour les générateurs combinant différentes familles.

Un résumé des règles retenues est donné au tableau 6.12. Les résultats présentés en dimension $t = 6$ sont nouveaux, alors que plusieurs des résultats en dimension 2 ont déjà été publiés [27, 31]. Notons que le test en dimension 6 avec d impair n'a pas donné de résultats clairs pour les LFSR, d'où l'absence de cette règle. Quant au cas où d est une puissance de 2, il n'a été appliqué qu'en dimension $t = 6$, pour réduire la quantité de tests à effectuer.

Nous pouvons constater que les LCG et les LFSR commencent à échouer les tests assez rapidement, avec une taille échantillonnale $n \approx 2^{3+e/2}$. Dans tous les cas, des p-valeurs trop élevées finissent par apparaître, à cause de la trop grande uniformité des points produits. Les générateurs non linéaires, quant à eux, ont commencé à éprouver des difficultés beaucoup plus tard, lorsque la période a été épuisée entièrement.

Les règles manquantes pour les autres familles de générateurs indiquent que des résultats difficiles à interpréter ont été obtenus. Les tableaux 6.13 et 6.14 donnent des exemples de p-valeurs produites en dimension $t = 6$, avec $h \approx 2^e$ choisi comme une puissance de 2, pour les LFSR combinés avec des générateurs inversifs de période 2^6 et 2^{10}. Rappelons que les symboles – apparaissant en bas de tableau indiquent que le test n'a pas pu être effectué, parce qu'une trop grande quantité de mémoire

Tableau 6.12: Valeurs de $\tilde{\nu}$, ν^* et γ, pour le test de X^2 avec $h \approx 2^e$

Famille	t	$\tilde{\gamma}$	$\tilde{\nu}$	ν^*
LCG	2	1/2	3	4
	6	1/2	4	5
LFSR, d impair	2	1/2	4	5
LFSR, d puissance de 2	6	1/2	3	5
Inversifs explicites	2	1	1	1
	6	1	1	1
Cubiques	2	1	0	0
	6	1	0	0

aurait été nécessaire. Alors que certains générateurs échouent le test à cause d'une trop grande uniformité des points générés (p-valeurs trop près de 1), d'autres échouent par manque d'uniformité (p-valeurs trop près de 0). Aussi, le nombre de vecteurs pouvant être produits avant que des p-valeurs suspectes n'apparaissent semble dépendre des générateurs choisis, plutôt que de la famille. Il est donc difficile de dégager une règle à partir de ces résultats.

Il est toutefois intéressant de noter que les générateurs ayant une petite composante non linéaire de période 2^6 échouent surtout le test à cause d'une trop grande uniformité des points produits, alors que la majorité des générateurs combinés avec un générateur inversif de période 2^{10} ont plutôt éprouvé des difficultés à cause d'un manque d'uniformité. Cette tendance a été observée pour toutes les familles de générateurs combinant des familles différentes, et s'explique bien par la théorie développée au chapitre précédent. Dans le cas des LCG combinés par une addition modulo 1, un deuxième générateur petit implique un nombre restreint de translations du réseau de base. Dans ce cas, la très grande uniformité du LCG a de bonnes chances d'être conservée. À l'inverse, un deuxième générateur à combiner plus important diminue la taille du réseau de base, et augmente le nombre de translations effectuées, ce qui peut avoir pour effet de diminuer l'uniformité des points produits. De même, dans le cas des LFSR combinés, l'uniformité du générateur est assurée par les propriétés d'équidistribution du LFSR. Ainsi, un LFSR de taille plus importante permet d'assurer une meilleure

Tableau 6.13: p-valeurs du test X^2 avec $h \approx 2^e$ choisi comme une puissance de 2, $t = 6$ et $\gamma = 1/2$, pour les LFSR combinés avec un inversif de période 2^6

e	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$	$\nu = 8$	$\nu = 9$
20				3E-3	5E-9	ϵ
21	- 1E-12	$-\epsilon$	$-\epsilon$	$-\epsilon$	$-\epsilon$	$-\epsilon$
22			- 2E-3	- 2E-9	$-\epsilon$	$-\epsilon$
23			- 3E-11	$-\epsilon$	$-\epsilon$	$-\epsilon$
24	- 1E-3	$-\epsilon$	$-\epsilon$	$-\epsilon$	$-\epsilon$	$-\epsilon$
25		- 4E-3	- 4E-9	$-\epsilon$	$-\epsilon$	$-\epsilon$
26					- 9E-6	$-\epsilon$
27	- 1E-3	- 8E-8	$-\epsilon$	$-\epsilon$	$-\epsilon$	$-\epsilon$
28				5E-3	1E-3	1E-7
29	- 1E-3	- 3E-12	$-\epsilon$	$-\epsilon$	$-\epsilon$	—
30			4E-5	9E-10	ϵ	—
31		- 3E-3	- 5E-7	- 9E-6	—	—
32			$-\epsilon$	$-\epsilon$	—	—
33			- 7E-7	—	—	—

équidistribution du générateur combiné, et donc une meilleure uniformité.

Nous pouvons aussi remarquer que les LFSR combinés à un inversif commencent généralement à échouer le test pour des valeurs de ν plus grandes que dans le cas des LFSR de période équivalente. Cette caractéristique a pu être remarquée pour plusieurs générateurs combinant des familles différentes. Aussi, nous avons voulu quantifier cette amélioration, pour permettre de comparer les différentes familles, même si aucune règle n'a pu être trouvée. Pour ce faire, nous avons divisé les générateurs combinés en deux catégories : ceux dont une des composantes est un LFSR et qui sont combinés à l'aide d'un ou-exclusif, et ceux dont une des composantes est un LCG et qui sont combinés en faisant une addition modulo 1. Les générateurs du premier type ont été comparés avec les LFSR, alors que les générateurs de la deuxième catégorie ont été comparés aux LCG. Des tableaux de p-valeurs similaires à ceux présentés ont été produits, avec une valeur de $\gamma = 1/2$. Pour chaque générateur d'une même famille, la valeur de ν à partir de laquelle des p-valeurs suspectes commencent à apparaître a été notée (appelons-la $\tilde{\nu}$), de même que la valeur de ν pour laquelle des ϵ ont commencé à être observés (notons-la par ν^*). Nous avons ensuite soustrait ces valeurs à celles obtenues pour les

Tableau 6.14: p-valeurs du test X^2 avec $h \approx 2^e$ choisi comme une puissance de 2, $t = 6$ et $\gamma = 1/2$, pour les LFSR combinés avec un inversif de période 2^{10}

e	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$	$\nu = 8$	$\nu = 9$	$\nu = 10$
20		4E-3	1E-3	ϵ	ϵ	ϵ	ϵ
21					2E-8	ϵ	ϵ
22			- 1E-4	- 7E-7	$-\epsilon$	$-\epsilon$	$-\epsilon$
23			9E-3	7E-7	3E-14	ϵ	ϵ
24				- 1E-4	- 2E-12	$-\epsilon$	$-\epsilon$
25					- 2E-4	- 3E-13	$-\epsilon$
26							- 2E-6
27		6E-6	3E-6	3E-14	ϵ	ϵ	—
28				4E-3	9E-4	3E-13	—
29				7E-5	ϵ	—	—
30			- 1E-4	$-\epsilon$	$-\epsilon$	—	—
31					—	—	—
32	9E-6	2E-3	2E-3	ϵ	—	—	—
33				—	—	—	—

LCG ou les LFSR de période équivalente. Une moyenne a finalement été calculée. Les valeurs obtenues, que nous noterons par $\bar{\tilde{\nu}}$ et $\bar{\nu}^*$, représentent donc une moyenne du $\log_2$ de l'augmentation du nombre de vecteurs $\mathbf{u}_{ti,t}$ devant être produits par les générateurs d'une même famille pour faire échouer le test, par rapport aux LCG ou aux LFSR de périodes équivalentes.

Les tableaux 6.15 et 6.16 présentent les résultats obtenus pour les générateurs combinés à l'aide d'un ou-exclusif, alors que le tableau 6.17 donne les résultats pour les générateurs combinés à l'aide d'une addition modulo 1. Le symbole $\oplus$ entre deux familles de générateurs indique qu'une combinaison de type ou-exclusif a été effectuée. Aussi, le nombre de générateurs inclus dans la moyenne calculée est indiqué. Ce nombre varie entre 7 et 17, et dépend de la quantité de générateurs pour lesquels une p-valeur suspecte a pu être observée. Il est à noter que certains générateurs combinés ont parfois été inclus dans les moyennes, même si des p-valeurs suspectes ou valant ϵ n'ont pas pu être produites. Dans ces cas, un symbole $\geq$ a été ajouté devant la moyenne, indiquant qu'un résultat possiblement plus élevé aurait pu être obtenu.

Tableau 6.15: Log de l'amélioration moyenne des générateurs, par rapport à la famille LFSR, pour le test X^2 en dimension $t = 2$ avec $k \approx \rho$ et d impair

Famille	Nombre	$\bar{\bar{\nu}}$	$\bar{\nu}^*$
LFSR	-	0.0	0.0
LFSR $\oplus$ INV ($e_2 = 6$)	14	2.6	≥ 3.0
LFSR $\oplus$ INV ($e_2 = 7$)	16	2.2	≥ 2.4
LFSR $\oplus$ INV ($e_2 = 8$)	12	4.2	≥ 3.7
LFSR $\oplus$ INV ($e_2 = 9$)	10	≥ 4.0	≥ 3.8
LFSR $\oplus$ INV ($e_2 = 10$)	10	≥ 5.0	≥ 4.4
LFSR $\oplus$ INV ($e_2 = e/4$)	9	2.3	≥ 2.9
LFSR $\oplus$ INV ($e_2 = e/2$)	7	≥ 3.8	≥ 3.6
LFSR $\oplus$ CUB ($e_2 = 6$)	12	≥ 3.0	≥ 2.7
LFSR $\oplus$ CUB ($e_2 = 7$)	18	≥ 3.4	≥ 3.1
LFSR $\oplus$ CUB ($e_2 = 8$)	17	≥ 3.2	≥ 2.6
LFSR $\oplus$ CUB ($e_2 = 9$)	12	3.9	≥ 3.8
LFSR $\oplus$ CUB ($e_2 = 10$)	13	≥ 4.3	≥ 4.0
LFSR $\oplus$ CUB ($e_2 = e/4$)	11	2.3	≥ 2.4
LFSR $\oplus$ CUB ($e_2 = e/2$)	10	≥ 5.3	≥ 4.4
LFSR $\oplus$ LCG ($e_2 = 10$)	11	≥ 2.4	2.4
LFSR $\oplus$ LCG ($e_2 = e/2$)	9	≥ 3.8	≥ 3.8

Nous pouvons constater que les générateurs combinés dont une des composantes est non linéaire ont très bien réagi par rapport à ces tests du khi-deux. Ainsi, les LFSR ou les LCG combinés avec un générateur non linéaire de période 2^6 ont performé entre quatre et huit fois mieux que les générateurs linéaires seuls de période équivalente. Aussi, plus la taille du générateur non linéaire augmente, meilleure est la performance des générateurs combinés. Ainsi, les générateurs combinés ayant une composante non linéaire de période 2^{10} ont réussi à faire jusqu'à $2^5 = 32$ fois mieux que les LFSR ou les LCG seuls.

Nous pouvons aussi constater que les générateurs inversifs et cubiques ont réagi de façon semblable aux tests, lorsque combinés avec des LCG ou des LFSR. Les combinaisons impliquant des LCG et des LFSR ont aussi produits de très bons résultats. Une amélioration légèrement moins prononcée a cependant été obtenue en dimension 2,

Tableau 6.16: Log de l'amélioration moyenne des générateurs, par rapport à la famille LFSR, pour le test X^2 en dimension $t = 6$ avec $k \approx \rho$

Famille	Nombre	d	$\bar{\tilde{\nu}}$	$\bar{\nu}^*$
LFSR	-	Impair	0.0	0.0
	-	Puiss. 2	0.0	0.0
LFSR $\oplus$ INV ($e_2 = 6$)	14	Impair	≥ 1.7	≥ 1.6
	17	Puiss. 2	2.4	≥ 2.5
LFSR $\oplus$ INV ($e_2 = 10$)	9	Impair	≥ 3.9	≥ 3.0
	11	Puiss. 2	≥ 4.0	≥ 4.2
LFSR $\oplus$ INV ($e_2 = e/4$)	7	Impair	≥ 2.7	≥ 2.1
	11	Puiss. 2	2.2	≥ 2.2
LFSR $\oplus$ INV ($e_2 = e/2$)	7	Impair	≥ 4.4	≥ 3.1
	9	Puiss. 2	≥ 5.3	≥ 4.9
LFSR $\oplus$ CUB ($e_2 = 6$)	13	Impair	2.3	≥ 1.7
	17	Puiss. 2	2.7	≥ 2.6
LFSR $\oplus$ CUB ($e_2 = 10$)	7	Impair	≥ 3.5	≥ 2.9
	11	Puiss. 2	≥ 3.7	≥ 3.9
LFSR $\oplus$ CUB ($e_2 = e/4$)	7	Impair	2.0	≥ 1.4
	9	Puiss. 2	≥ 3.6	≥ 3.4
LFSR $\oplus$ CUB ($e_2 = e/2$)	7	Impair	3.7	≥ 3.1
	8	Puiss. 2	4.0	≥ 4.5
LFSR $\oplus$ LCG ($e_2 = 10$)	9	Impair	≥ 2.2	≥ 1.9
	11	Puiss. 2	3.0	≥ 2.9
LFSR $\oplus$ LCG ($e_2 = e/2$)	7	Impair	2.7	≥ 2.3
	12	Puiss. 2	≥ 4.0	≥ 4.0

lorsque les sorties ont été additionnés modulo 1, et en dimension 6 avec h choisi impair, lorsque des ou-exclusifs ont été utilisés.

Notons finalement que tous les LCG combinés ont performé aussi bien ou mieux que les LCG de période équivalente, et ce peu importe le deuxième générateur utilisé. Quant aux LFSR combinés, ils ont presque tous fait aussi bien ou mieux que les LFSR de période équivalente. Les quelques exceptions sont généralement dues à des LFSR échouant beaucoup plus tard que les autres générateurs de leur famille. Dans ces cas, les générateurs combinés sont comparés à des LFSR anormalement performants, d'où le faible écart négatif obtenu pour les valeurs de $\tilde{\nu}$ ou ν^*.

Tableau 6.17: Log de l'amélioration moyenne des générateurs, par rapport à la famille des LCG, pour le test X^2 avec $k \approx \rho$

Famille	Nombre	t	$\bar{\bar{\nu}}$	$\bar{\nu}^*$
LCG	-	2	0.0	0.0
	-	6	0.0	0.0
LCG + INV ($e_2 = 6$)	13	2	2.8	≥ 2.5
	17	6	2.2	≥ 2.2
LCG + INV ($e_2 = 7$)	16	2	≥ 3.5	≥ 3.2
LCG + INV ($e_2 = 8$)	15	2	3.9	≥ 3.5
LCG + INV ($e_2 = 9$)	12	2	≥ 5.2	≥ 4.7
LCG + INV ($e_2 = 10$)	11	2	≥ 5.6	≥ 5.6
	9	6	≥ 5.0	≥ 4.3
LCG + INV ($e_2 = e/4$)	11	2	≥ 4.0	≥ 3.1
	8	6	≥ 2.9	≥ 3.1
LCG + INV ($e_2 = e/2$)	10	2	≥ 6.5	≥ 5.2
	9	6	≥ 5.0	≥ 4.2
LCG + CUB ($e_2 = 6$)	15	2	≥ 3.1	≥ 3.2
	17	6	≥ 3.3	≥ 2.7
LCG + CUB ($e_2 = 7$)	14	2	3.6	≥ 3.6
LCG + CUB ($e_2 = 8$)	13	2	5.0	≥ 4.3
LCG + CUB ($e_2 = 9$)	13	2	5.0	≥ 4.8
LCG + CUB ($e_2 = 10$)	11	2	≥ 5.3	≥ 4.7
	9	6	≥ 5.2	≥ 4.5
LCG + CUB ($e_2 = e/4$)	13	2	≥ 3.7	≥ 3.0
	9	6	≥ 4.0	≥ 3.4
LCG + CUB ($e_2 = e/2$)	12	2	≥ 6.4	≥ 5.3
	12	6	≥ 4.7	≥ 4.2
LCG + LFSR ($e_2 = 10$)	13	2	≥ 4.2	≥ 4.0
	9	6	≥ 4.4	≥ 4.3
LCG + LFSR ($e_2 = e/2$)	12	2	4.4	≥ 4.3
	9	6	4.3	≥ 4.1

Les résultats obtenus suggèrent que l'uniformité des points produits par les différents générateurs combinés essayés se rapproche davantage de l'uniformité d'une vraie séquence aléatoire que celle des sorties générées par des LCG ou des LFSR. Il faut toutefois se rappeler que des règles claires n'ont pas pu être trouvées, et que seules des moyennes des améliorations individuelles ont été présentées. Il est donc difficile de prédire les gains

précis pouvant être faits avec un générateur particulier. Il est aussi difficile de déterminer si la moyenne calculée s'applique encore, pour des générateurs de plus grandes périodes. Mais le fait qu'une amélioration ait pu être notée pour presque tous les générateurs combinés testés est très encourageant, et devrait certainement augmenter notre confiance envers ces générateurs.

Test de collisions

Le test de collisions a été appliqué avec les mêmes paramètres que ceux choisis pour le test du khi-deux. Un exemple de p-valeurs obtenues est donné au tableau 6.18. Il s'agit du test en dimension 6 avec h choisi comme une puissance de 2, pour les LFSR combinés avec un générateur inversif de période 2^{10}. Nous pouvons constater que les résultats présentés sont très semblables à ceux obtenus dans les mêmes conditions, pour le test du khi-deux (voir le tableau 6.14 à titre de comparaison). Cette remarque a pu être faite pour toutes les familles testées.

En fait, comme mentionné dans [27, 31], dans le cas très clairsemé considéré, il est très rare de voir plus de deux points se retrouver dans une même boîte. Dans ce contexte, la statistique C indique le nombre de boîtes ayant deux points, et il existe une bijection entre les valeurs de C et X^2. Pour bien voir cette bijection, notons par $q = n/h$ le rapport entre le nombre de points et le nombre de boîtes, et par r le nombre de boîtes contenant 2 points. Nous avons alors

$$X^2 = \frac{r(2-q)^2 + (n-2r)(1-q)^2 + (h-n+r)q^2}{q}.$$

Les quantités n, h et q étant fixées, la valeur de r (et donc celle de la statistique C) peut facilement être déduite à partir du X^2, et vice-versa. Ces deux statistiques sont donc fortement liées, ce qui explique les p-valeurs semblables obtenues.

Comme les conclusions tirées pour le test de collisions sont en tous points similaires à celles présentées pour le test du khi-deux, elles ne seront pas discutées davantage ici.

Tableau 6.18: p-valeurs du test C avec $h \approx 2^e$ choisi comme une puissance de 2, $t = 6$ et $\gamma = 1/2$, pour les LFSR combinés avec un inversif de période 2^{10}

e	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$	$\nu = 8$	$\nu = 9$	$\nu = 10$
20			9E-4	ϵ	ϵ	ϵ	ϵ
21				3E-3	1E-13	ϵ	ϵ
22			- 1E-4	- 9E-7	$-\epsilon$	$-\epsilon$	$-\epsilon$
23			1E-2	3E-6	1E-12	ϵ	ϵ
24				- 7E-5	- 2E-12	$-\epsilon$	$-\epsilon$
25					- 2E-4	- 2E-14	$-\epsilon$
26							
27		3E-5	9E-6	2E-13	ϵ	ϵ	—
28				4E-3	9E-4	1E-13	—
29				8E-5	ϵ	—	—
30			- 9E-5	$-\epsilon$	$-\epsilon$	—	—
31					—	—	—
32	2E-5	2E-3	2E-3	ϵ	—	—	—
33				—	—	—	—

6.4.3 Test de l'espacement des anniversaires

Tel que suggéré dans [30], le test de l'espacement des anniversaires a été appliqué avec une valeur de $\lambda = n^3/(4h) \approx 1$. Pour ce faire, le nombre de divisions de l'invervalle $[0, 1]$ a été choisi comme étant $d = \lfloor (n^3/4)^{1/t} \rfloor$. Les dimensions $t = 2$ et $t = 8$ ont été étudiées.

Résultats attendus

Ce test a déjà été effectué avec des LCG, des LFSR, des générateurs non linéaires et les familles combinant des LCG avec des générateurs cubiques ou des LFSR de même taille [30] . Dans tous les cas, une règle de la forme $\tilde{n} \approx 2^{\tilde{\nu}+\tilde{\gamma}e}$ a pu être établie.

Une explication détaillée du comportement des générateurs linéaires est donné dans [30]. Ces générateurs échouent très rapidement le test, à cause de la structure trop régulière de leurs points. En fait, lorsqu'une partition assez fine de l'espace $[0, 1]^t$ est utilisée, les points produits par les générateurs linéaires ont tendance à se retrouver

dans des boîtes dont les numéros sont espacés par un même multiple. Dans ce cas, un grand nombre de collisions entre les espacements S_j des boîtes survient, nous forçant à rejeter l'hypothèse $\mathcal{H}_0$. Dans le cas des générateurs non linéaires, la période entière doit généralement être épuisée avant que le test n'échoue. Un nombre anormalement élevé de collisions est alors obtenu, puisque des points identiques sont produits.

Une règle a aussi pu être obtenue pour les familles combinant des LCG avec des générateurs cubiques ou des LFSR de même taille. En fait, une structure semblable à celle des points produits par les LCG est encore présente dans ce cas, quoique moins prononcée. Une règle systématique est donc toujours trouvée, mais beaucoup plus de points doivent être générés avant que cette règle n'émerge. Nous pouvons nous attendre à des résultats similaires pour les autres familles de LCG combinés, leurs points étant également tirés d'une superposition de plusieurs réseaux translatés.

Le cas des LFSR combinés est légèrement plus complexe. Nous n'avons aucun résultat préliminaire sur lequel nous baser. Aussi, la théorie nous permet difficilement de prévoir si un défaut systématique pourra être détecté pour tous les générateurs d'une même famille. Nous pouvons cependant nous attendre à ce que ces générateurs combinés performent mieux que les LFSR seuls, à cause de la structure moins régulière des points qu'ils produisent.

Test de l'espacement des anniversaires avec $\lambda \approx 1$

Voici maintenant les résultats obtenus au test de l'espacement des anniversaires, avec $\lambda \approx 1$. Les tableaux 6.19 à 6.21 donnent des exemples de p-valeurs produites en dimension 2, pour les LCG et certains générateurs combinés. Nous pouvons voir sur ces tableaux que les générateurs commencent à échouer le test ou à obtenir d'extrêmement petites p-valeurs (désignés par ϵ) pour une valeur de ν similaire. Cette fois, des règles aussi claires ont pu être établies pour presque toutes les familles de générateurs testées.

Un résumé des règles trouvées pour les générateurs de base (LCG, LFSR, et générateurs non linéaires) est donné au tableau 6.23. Des résultats similaires ont déjà été

Tableau 6.19: p-valeurs du test de l'espacement des anniversaires en dimension $t = 2$, avec $\lambda \approx 1$ et $\gamma = 1/3$, pour les LCG

e	$\nu = 0$	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$
16			4E-3	ϵ	ϵ
17			6E-4	ϵ	ϵ
18			1E-5	ϵ	ϵ
19				ϵ	ϵ
20			1E-8	ϵ	ϵ
21			8E-5	ϵ	ϵ
22			4E-3	ϵ	ϵ
23			6E-4	ϵ	ϵ
24			6E-4	ϵ	ϵ
25				ϵ	ϵ
26			6E-4	ϵ	ϵ
27				ϵ	ϵ
28			8E-5	ϵ	ϵ
29			6E-4	ϵ	ϵ
30			8E-5	ϵ	ϵ
31			1E-5	ϵ	ϵ
32				ϵ	ϵ
33			1E-5	ϵ	ϵ
34			1E-5	ϵ	ϵ
35				ϵ	ϵ
36				ϵ	ϵ
37				ϵ	ϵ
38			1E-5	ϵ	ϵ
39			8E-5	ϵ	ϵ
40			8E-5	ϵ	ϵ

obtenus et publiés dans [30]. Notons que le test en dimension 2 n'a pas pu être appliqué pour les générateurs non linéaires, parce qu'ils échouent le test pour une très grande taille échantillonnale n, et que leur nombre de bits de précision n'atteint pas le $\log_2 d$, pour un nombre de divisions d permettant de conserver une valeur de $\lambda \approx 1$. Nous avons donc remplacé, pour ces générateurs, le test en dimension 2 par un test en dimension 4.

Comme prévu, nous pouvons constater qu'avec la valeur de λ choisie, ce test est très discriminant pour les LCG, qui échouent lorsqu'un nombre de vecteurs proportionnel à la racine cubique de leur période est généré. Ce test est aussi assez costaud pour les LFSR, qui échouent avant qu'une quantité de points proportionnelle à la racine carrée

Tableau 6.20: p-valeurs du test de l'espacement des anniversaires en dimension $t = 2$, avec $\lambda \approx 1$ et $\gamma = 1/3$, pour les LCG combinés à un cubique de période 2^6

e	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$
16				ϵ	ϵ
17				ϵ	ϵ
18			4E-3	ϵ	ϵ
19				ϵ	ϵ
20				ϵ	ϵ
21				3E-13	ϵ
22				1E-8	ϵ
23			4E-3	1E-15	ϵ
24				ϵ	ϵ
25			4E-3	ϵ	ϵ
26				ϵ	ϵ
27				ϵ	ϵ
28				2E-14	ϵ
29				ϵ	ϵ
30			6E-4	ϵ	ϵ
31			1E-5	ϵ	ϵ
32			6E-4	ϵ	ϵ
33				ϵ	ϵ
34				ϵ	ϵ
35				ϵ	ϵ
36			6E-4	ϵ	ϵ

de leur période ne soit produite. Les générateurs non linéaires, quant à eux, performent très bien en dimension 4, puisque leur période entière peut être générée sans problème. Les générateurs cubiques ont toutefois donné de moins bons résultats en dimension 8. Il semble qu'un défaut de structure soit détecté dans ce cas. Il est possible que ce défaut affecte la performance des générateurs combinés ayant une composante cubique, d'où l'importance d'appliquer ce test aux générateurs combinés.

Un résumé des règles observées pour les LCG combinés à un autre générateur par une addition modulo 1 peut être trouvé au tableau 6.24, et les résultats concernant les LFSR combinés à un deuxième générateur par un ou-exclusif sont présentés au tableau 6.25. Les symboles * suivant parfois la dimension du test, dans le cas des LFSR combinés, indiquent qu'une règle un peu moins précise a été observée. En fait, pour

Tableau 6.21: p-valeurs du test de l'espacement des anniversaires en dimension $t = 2$, avec $\lambda \approx 1$ et $\gamma = 2/5$, pour les LFSR combinés à un cubique de période 2^6

e	$\nu = 4$	$\nu = 5$	$\nu = 6$	$\nu = 7$	$\nu = 8$
16			6E-4	ϵ	ϵ
17			1E-8	ϵ	ϵ
18			6E-4	ϵ	ϵ
19			8E-5	ϵ	ϵ
20			1E-7	ϵ	ϵ
21			6E-4	ϵ	ϵ
22			6E-4	ϵ	ϵ
23			1E-7	ϵ	ϵ
24				ϵ	ϵ
25			6E-11	5E-12	ϵ
26				ϵ	ϵ
27			6E-4	2E-14	ϵ
28					2E-5
29				ϵ	ϵ
30			1E-6	2E-13	ϵ
31		4E-3	3E-5	ϵ	ϵ
32			4E-4	ϵ	ϵ
33				2E-4	ϵ
34				2E-7	ϵ

tous les tests en dimension 8, au moins un des LFSR combinés a moins bien performé que les autres générateurs de la même famille, lorsqu'un deuxième générateur de taille constante a été employé. Par exemple, le tableau 6.22 donne les p-valeurs obtenues avec les LFSR combinés à un générateur cubique de période 2^8. Dans ce cas, ce sont les générateurs de période 2^{23} et 2^{32} qui ont légèrement moins bien performé. Les règles rencontrées sont toutefois assez claires pour pouvoir être notées, et les déviations rencontrées sont rares et généralement très peu prononcées.

Nous pouvons voir, sur les tableaux résumés présentés, que les générateurs combinés impliquant des générateurs cubiques se sont comportés de façon identiques à ceux dont une des composantes est de type inversif, et ce même en dimension 8. Il semble donc que le défaut de structure des générateurs cubiques rencontré en dimension 8 n'ait pas eu de répercussions sur les générateurs combinés.

Tableau 6.22: p-valeurs du test de l'espacement des anniversaires en dimension $t = 8$, avec $\lambda \approx 1$ et $\gamma = 2/5$, pour les LFSR combinés à un cubique de période 2^7

e	$\nu = 6$	$\nu = 7$	$\nu = 8$	$\nu = 9$	$\nu = 10$
17			5E-3	ϵ	ϵ
18				1E-12	ϵ
19			5E-3	3E-9	ϵ
20	1E-7			3E-7	ϵ
21			5E-3	ϵ	ϵ
22			7E-4	2E-6	ϵ
23		ϵ	ϵ	ϵ	ϵ
24				2E-13	ϵ
25				5E-12	ϵ
26			8E-3	ϵ	ϵ
27				6E-8	ϵ
28				5E-4	ϵ
29				ϵ	ϵ
30			1E-4	3E-9	—
31			3E-7	ϵ	—
32	2E-3	3E-8	ϵ	—	—
33		8E-3		—	—

Nous pouvons aussi remarquer que, tout comme pour le test des points les plus rapprochés, le fait de combiner les LCG ou les LFSR avec un petit générateur d'une autre famille de période constante ne permet pas d'augmenter sensiblement la constante $\tilde{\gamma}$. Par contre, des valeurs de $\tilde{\nu}$ ou ν^* plus grandes sont obtenues. Par exemple, l'ajout d'un petit générateur non linéaire de période aussi petite que 2^6 permet déjà d'améliorer la performance des LCG ou des LFSR par un facteur variant entre 2 et 4, alors que l'ajout d'un deuxième générateur non linéaire de période 2^{10} permet de générer environ huit fois plus de nombres. Encore une fois, l'augmentation de la taille du deuxième générateur à combiner se traduit par une augmentation de performance. Il est aussi important de noter que même les quelques LFSR combinés ayant déviés des règles inscrites ont mieux performés que les LFSR seuls de période équivalente, et ce pour toutes les familles.

En ce qui concerne les générateurs combinés dont la deuxième composante est de taille $2^{e/4}$ ou $2^{e/2}$, nous pouvons constater qu'ils ont permi d'améliorer la constante $\tilde{\gamma}$.

Tableau 6.23: Résultats du test d'espacement des anniversaires avec $\lambda \approx 1$, pour les familles de base

Famille	t	$\tilde{\gamma}$	$\tilde{\nu}$	ν^*
LCG	2	1/3	2	3
	8	1/3	4	5
LFSR	2	2/5	4	5
	8	2/5	6	7
Inversifs explicites	4	1	1	1
	8	1	1	1
Cubiques	4	1	0	0
	8	1/2	6	7

Dans le cas des LCG, cette constante est passée de 1/3 à des valeurs situées entre 2/5 et 3/5, alors que des LFSR combinés ont réussi à augmenter la fraction de 2/5 à 3/4.

Notons finalement que la combinaison de LCG avec des LFSR au moyen d'une addition modulo 1 a produit d'aussi bons résultats que la combinaison avec des générateurs non linéaires. Les LFSR combinés aux LCG à l'aide d'un ou-exclusif ont aussi donné de bons résultats, mais ont commencé à échouer les tests environ deux fois plus rapidement que les générateurs équivalents ayant une composante non linéaire.

6.4.4 Test de l'espacement des apparitions

Dans ce qui suit, nous avons voulu étudier la sensibilité du test de l'espacement des apparitions, pour différents choix de s et L. Tel que suggéré dans [36], une phase d'initialisation de $Q \geq 10 \times 2^L$ a été utilisée, et les valeurs de L ont été choisies entre 6 et 16.

Les tests ont été appliqués à des LCG et des LFSR particuliers, tirés des familles testées aux sections précédentes. Plus précisément, les générateurs de période près de 2^{10} et 2^{15} ont été retenus. Une phase d'initialisation de $Q = 10 \times 2^L$ a été sélectionnée, et les différents couples (s, L) suivants ont été essayés : $(3, 6)$, $(4, 8)$, $(2, 6)$ et $(3, 9)$.

Tableau 6.24: Résultats du test d'espacement des anniversaires avec $\lambda \approx 1$, pour les LCG combinés à un autre générateur par une addition modulo 1

Famille	t	$\tilde{\gamma}$	$\tilde{\nu}$	ν^*
LCG + Inv ($e_2 = 6$)	2		4	4
	8		5	6
LCG + Inv ($e_2 = 7$)	2		4	5
	8		6	6
LCG + Inv ($e_2 = 8$)	2	1/3	5	5
	8		6	7
LCG + Inv ($e_2 = 9$)	2		5	6
	8		7	8
LCG + Inv ($e_2 = 10$)	2		5	6
	8		7	8
LCG + Inv ($e_2 = e/4$)	2	2/5	2	3
	8	1/2	1	2
LCG + Inv ($e_2 = e/2$)	2	1/2	2	3
	8	3/5	1	2
LCG + Cub ($e_2 = 6$)	2		4	4
	8		6	6
LCG + Cub ($e_2 = 7$)	2		4	5
	8		6	6
LCG + Cub ($e_2 = 8$)	2	1/3	5	5
	8		6	7
LCG + Cub ($e_2 = 9$)	2		5	6
	8		7	8
LCG + Cub ($e_2 = 10$)	2		5	6
	8		7	8
LCG + Cub ($e_2 = e/4$)	2	2/5	2	3
	8	1/2	1	2
LCG + Cub ($e_2 = e/2$)	2	1/2	2	3
	8	3/5	2	2
LCG + LFSR ($e_2 = 10$)	2	1/3	5	6
	8	1/3	7	8
LCG + LFSR ($e_2 = e/2$)	2	3/5	0	0
	8	3/5	1	2

Tableau 6.25: Résultats du test d'espacement des anniversaires avec $\lambda \approx 1$, pour les LFSR combinés à un autre générateur par un ou-exclusif bit à bit

Famille	t	$\tilde{\gamma}$	$\tilde{\nu}$	ν^*
LFSR $\oplus$ Inv ($e_2 = 6$)	2	2/5	6	7
	8*	2/5	8	9
LFSR $\oplus$ Inv ($e_2 = 7$)	2	1/2	4	5
	8*	2/5	8	10
LFSR $\oplus$ Inv ($e_2 = 8$)	2	1/2	5	6
	8*	1/2	7	8
LFSR $\oplus$ Inv ($e_2 = 9$)	2	1/2	5	6
	8*	1/2	7	8
LFSR $\oplus$ Inv ($e_2 = 10$)	2	1/2	6	7
	8*	1/2	8	9
LFSR $\oplus$ Inv ($e_2 = e/4$)	2	3/5	1	2
	8	2/5	8	9
LFSR $\oplus$ Inv ($e_2 = e/2$)	2	3/5	4	5
	8	3/4	3	4
LFSR $\oplus$ Cub ($e_2 = 6$)	2	2/5	6	7
	8*	2/5	8	9
LFSR $\oplus$ Cub ($e_2 = 7$)	2	1/2	4	5
	8*	2/5	8	10
LFSR $\oplus$ Cub ($e_2 = 8$)	2	1/2	5	6
	8*	1/2	7	8
LFSR $\oplus$ Cub ($e_2 = 9$)	2	1/2	5	6
	8*	2/5	10	11
LFSR $\oplus$ Cub ($e_2 = 10$)	2	1/2	6	7
	8*	1/2	8	9
LFSR $\oplus$ Cub ($e_2 = e/4$)	2	3/5	1	2
	8	2/5	8	9
LFSR $\oplus$ Cub ($e_2 = e/2$)	2	3/5	4	5
	8	3/4	3	4
LFSR $\oplus$ LCG ($e_2 = 10$)	2	1/2	5	6
	8	1/2	7	8
LFSR $\oplus$ LCG ($e_2 = e/2$)	2	3/5	3	4
	8	3/4	2	3

Tableau 6.26: Taille échantillonnale requise pour faire échouer le test de l'espacement des apparitions avec certains LCG et LFSR

Test		LCG		LFSR	
s	L	2^{10}	2^{15}	2^{10}	2^{15}
3	6	$\geq 3 \times 10^3$	$\geq 3 \times 10^5$	$\geq 8 \times 10^3$	$\geq 1 \times 10^6$
4	8	—	$\geq 1 \times 10^6$	—	$\geq 4 \times 10^5$
2	6	$\geq 2 \times 10^3$	$\geq 3 \times 10^5$	$\geq 1 \times 10^4$	$\geq 1 \times 10^6$
3	9	—	$\geq 1 \times 10^6$	—	$\geq 2 \times 10^4$

Avec ces choix de s et L, les séquences de L bits sont formés en prenant 2 ou 3 nombres consécutifs générés. Les couples $(4, 8)$ et $(3, 9)$ n'ont cependant pas pu être testés avec les petits générateurs de période 2^{10}, puisque la phase d'initialisation, à elle seule, aurait requis la génération de $Q = 10 \times 2^L \geq 10 \times 2^8 \geq 2^{11}$ nombres, épuisant plus que la période entière du générateur. En général, nous avons éliminé les grandes valeurs de L, pour éviter de devoir générer une trop grande fraction de la période à l'étape d'initialisation. Nous espérons ainsi augmenter nos chances d'obtenir un test discriminant, nécessitant une petite taille échantillonnale $n = Q + K$ avant d'échouer.

Le tableau 6.26 indique le nombre de blocs $Q + K$ de bits qui ont dû être générés, avec les différents couples (s, L) testés, pour obtenir des p-valeurs inférieures à 0.01, ou supérieures à 0.99. Nous pouvons constater que dans presque tous les cas, la période doit être générée plusieurs fois avant que le test ne détecte un défaut de structure (les périodes valant environ $2^{10} = 1024$ et $2^{15} = 32768$). À titre de comparaison, rappelons que les LCG échouent les tests du khi-deux ou de collision de dimension $t = 2$ appliqués à la section 6.4.2 avec des p-valeurs supérieures à 0.99 avec une taille échantillonnale de $n = 2^{3+e/2}$, alors que les LFSR échouent dans les mêmes conditions avec des tailles d'échantillon de $n = 2^{4+e/2}$. Dans le cas des générateurs de période 2^{10} et 2^{15} utilisés, ceci donne des tailles échantillonnales de 256 et 1449 respectivement pour les LCG, et de 512 et 2897 respectivement pour les LFSR. Les tests de l'espacement des apparitions effectués ici requièrent donc la génération de sept à 667 fois plus de nombres que les

Tableau 6.27: p-valeurs du test de l'espacement des apparitions, avec $s = 3$, $L = 9$ et $\gamma = 1$, pour les LFSR

e	$\nu = -2$	$\nu = -1$	$\nu = 0$	$\nu = 1$	$\nu = 2$	$\nu = 3$
13						4E-3
14					2E-3	3E-5
15			3E-4	2E-7	2E-12	ϵ
16						
17						
18						4E-3
19						
20						

tests sériels appliqués à la section 6.4.2 avant de détecter un défaut de structure.

Le cas le plus discriminant est rencontré lorsque $s = 3$ et $L = 9$, avec le LFSR de taille de 2^{15}. Ce test a été appliqué à d'autres LFSR de la même famille, et les p-valeurs obtenues sont données au tableau 6.27. Nous voyons bien sur ce tableau que le LFSR de période 2^{15} échoue le test plus rapidement que les autres générateurs de sa famille. Même dans le cas le plus discriminant, la période doit donc généralement être épuisée plusieurs fois, avant que le test n'échoue.

Comme nous n'avons pas réussi à trouver des valeurs de s et L rendant le test de l'espacement des apparitions discriminant, et ce même avec des générateurs ayant une forte structure linéaire, nous avons décidé de ne pas poursuivre les tests avec les autres familles.

6.5 Familles de générateurs à retenir

Les tests appliqués ont permis de mieux comprendre la structure des familles de générateurs décrites à la section 6.3, et de comparer leur performance empirique, en dimensions 2, 6 et 8. Le choix des dimensions s'est généralement fait de façon arbitraire, mais le comportement des générateurs ne dépend habituellement pas beaucoup de la dimension choisie.

Nous avons pu constater que la structure moins régulière des générateurs combinant des récurrences linéaires et non linéaires a permis de faire des gains non négligeables par rapport aux LCG et aux LFSR, et ce pour tous les tests discriminants appliqués. Ces gains ont pris de l'importance avec la taille du générateur non linéaire utilisé. Ainsi, l'ajout d'un générateur non linéaire de période aussi petite que 2^6 a généralement permis de produire entre quatre et huit fois plus de nombres que les LCG ou les LFSR avant de faire échouer les tests, alors qu'un deuxième générateur non linéaire de période 2^{10} a permis de générer entre huit et 32 fois plus de nombres avant qu'un défaut de structure ne soit détecté. Lorsque la taille des générateurs non linéaires a augmenté avec celle des générateurs combinés, la constante $\tilde{\gamma}$ a pu être augmentée en plus, nous laissant entrevoir un gain considérable pour les générateurs de plus longues périodes.

Il est important de remarquer que les gains observés avec ces combinaisons n'auraient pas pu être obtenus en combinant des LCG entre eux, comme à la section 4.1, ou en combinant plus de LFSR entre eux, selon la méthode de la section 4.2. Les combinaisons de LCG ou de LFSR donnent bien sûr des générateurs de plus grandes périodes, ce qui est souhaitable, mais ne changent pas la nature des générateurs obtenus. Les générateurs formés correspondent encore à des LCG ou des LFSR, qui se comportent de façon similaire à ceux testés. En échange, les générateurs combinant des récurrences linéaires et non linéaires essayés permettent, en plus d'augmenter les périodes obtenues, d'améliorer les constantes $\tilde{\gamma}$ ou $\tilde{\nu}$. À périodes égales, ces générateurs permettent donc de générer plus de points, sans risquer de fausser les résultats d'une application particulière.

Les résultats suggèrent donc que des générateurs combinant des récurrences linéaires et non linéaires de période assez grande pour être utilisés en pratique auraient avantage à être développés, et que la taille de la partie non linéaire devrait être choisie la plus grande possible (mais en tenant compte des contraintes de vitesse et/ou de mémoire). Le chapitre suivant donnera des exemples de tels générateurs ayant de longues périodes.

Quant aux combinaisons impliquant des LCG et des LFSR, elles ont aussi donné d'excellents résultats, permettant dans tous les cas d'améliorer la performance des

LCG ou des LFSR seuls. Ces générateurs combinés ont parfois échoué les tests plus rapidement que les générateurs équivalents ayant une composante non linéaire, mais une différence généralement minime a pu être notée. Aussi, la linéarité du deuxième générateur risque de permettre une implantation relativement efficace, et ce même pour une taille plus imposante de la deuxième composante. Ces générateurs pourraient donc offrir un bon rapport entre performance et vitesse d'exécution, et certaines implantations seront suggérées au chapitre suivant.

Finalement, mentionnons que d'autres types de générateurs pourraient aussi être utilisés, dans les combinaisons avec des LCG ou des LFSR. Nous avons restreint notre étude empirique à quelques familles seulement, ayant une deuxième composante G_2 reconnue pour la bonne uniformité de ses points, en accordant une importance particulière aux générateurs combinant des récurrences linéaires et non linéaires. Des performances très similaires ont été obtenues avec les différents types de générateurs G_2 essayés, et ce même lorsqu'une deuxième composante linéaire a été utilisée. Aussi, il semble que ce soit surtout le fait de superposer plusieurs réseaux d'un LCG translatés de façon relativement aléatoire, ou encore de modifier plus ou moins aléatoirement les bits produits par un LFSR, qui influence les résultats obtenus, plutôt que la manière spécifique de translater les réseaux ou de modifier les bits. D'autres types de générateurs G_2 pourraient donc probablement être utilisés, et donner d'aussi bons résultats que ceux présentés.

Chapitre 7

Générateurs retenus et implantation

Nous venons de voir que la structure moins régulière des LCG ou des LFSR combinés à un autre type de générateur selon les méthodes décrites au chapitre 5 permet d'utiliser une plus grande portion de la période, par rapport aux LCG ou aux LFSR combinés entre eux, sans risquer de fausser les résultats d'une application. Dans un contexte de simulation, où une très grande quantité de nombres doivent être générés à partir d'un seul et même générateur, cette propriété devient très intéressante. Nous avons donc décidé de présenter, dans ce chapitre, quelques bons générateurs combinant des récurrences différentes, de périodes assez longues pour pouvoir être utilisés en pratique. Une attention particulière est portée à leur implantation, la vitesse d'exécution de ces générateurs devant rester assez bonne pour ne pas ralentir de façon excessive les programmes utilisant une grande quantité de variables aléatoires.

Des implantations de générateurs déjà connus sont d'abord données à la prochaine section. Les codes en C d'un MRG, d'un LFSR, d'un générateur cubique et d'un générateur inversif y sont présentés, de même que les temps d'exécution obtenus sur différentes machines, avec différents compilateurs C ou C++. Certaines des implantations sont ensuite reprises dans les nouvelles combinaisons proposées, et les temps obtenus sont discutés.

Notons que pour obtenir de bons générateurs rapides avec de longues périodes, les LCG ont tous été remplacés par des MRG d'ordre 2 ou 3. Ces MRG ont une structure de réseau semblable à celle des LCG, et devraient donc se comporter de façon similaire face aux tests statistiques.

7.1 Générateurs déjà connus

Voici les implantations de quatre générateurs de types différents : un MRG, un LFSR, un générateur cubique et un générateur inversif.

Le MRG présenté a été suggéré dans [19], alors que le LFSR provient de [21]. Ces deux générateurs ont de longues périodes dépassant 2^{100}, et sont considérés comme très rapides. Ils serviront de référence dans les comparaisons avec les nouveaux générateurs combinés proposés.

L'implantation des générateurs non linéaires a plutôt été prise dans [22]. Ces générateurs sont très lents, même lorsque des périodes beaucoup plus petites sont considérées. Nous avons tenu à les présenter quand même, pour donner une idée de leur implantation et de leur vitesse d'exécution. Il n'est toutefois pas conseillé de les utiliser en pratique, à cause de leur trop petite période.

7.1.1 MRG

Le tableau 7.1 donne les paramètres de deux MRG d'ordre 3 tirées de [19], qui lorsque combinés donnent des bonnes valeurs de M_{32} au test spectral (voir la section 2.1.4 pour la définition de M_τ). La période atteinte par le MRG combiné est maximale, et vaut $(m_1^3-1)(m_2^3-1)/2 \approx 2^{191}$. Les récurrences de chacune des composantes ont été choisies avec seulement deux coefficients non nuls, pour permettre une implantation plus efficace. Aussi, des conditions sur les paramètres du type $|a_{j,i}|m_j < 2^{53}$ sont respectées, permettant d'effectuer les calculs $a_{j,i}x_{j,n-i} \bmod m_j$ correctement avec l'arithmétique en virgule flottante (en supposant que la mantisse d'un nombre en virgule flottante a une précision de 53 bits).

La figure 7.1 donne une implantation en virgule flottante du MRG combiné, lorsque la combinaison est faite selon (4.2), avec $\delta_1 = -\delta_2 = 1$. Au départ, les vecteurs $(\texttt{s10}, \texttt{s11}, \texttt{s12})$ et $(\texttt{s20}, \texttt{s21}, \texttt{s22})$ doivent être initialisés pour contenir les valeurs de $(x_{1,0}, x_{1,1}, x_{1,2})$ et $(x_{2,0}, x_{2,1}, x_{2,2})$ respectivement. Ensuite, ces vecteurs contiennent les

Tableau 7.1: Paramètres d'un bon MRG combiné à deux composantes

m_1 m_2	$a_{1,1}$ $a_{2,1}$	$a_{1,2}$ $a_{2,2}$	$a_{1,3}$ $a_{2,3}$
$2^{32}-209$	0	1403580	-810728
$2^{32}-22853$	527612	0	-1370589

valeurs de $(x_{j,n-2}, x_{j,n-1}, x_{j,n})$ nécessaires pour calculer les prochains éléments $x_{j,n}$ des récurrences. La valeur de $x_{1,n} = (a_{1,2}x_{1,n-2} + a_{1,3}x_{1,n-3}) \bmod m_1$ est d'abord calculée, le résultat intermédiaire de $a_{1,2}x_{1,n-2} + a_{1,3}x_{1,n-3}$ étant conservé dans une variable de type `double`. Le même calcul est ensuite effectué pour la deuxième composante. Finalement, la combinaison est faite, et le résultat normalisé entre 0 et 1 est retourné. Il est à noter qu'une petite variante de la combinaison (4.2) est donnée ici : la constante de normalisation utilisée n'est plus $1/m_1$, mais plutôt $1/(m_1+1)$, et la sortie $u_n = 0$ est remplacée par $u_n = m_1/(m_1+1)$. Ce petit changement permet de s'assurer que les valeurs 0 et 1 ne seront jamais produites, ce qui peut être très utile pour certaines applications [19].

7.1.2 LFSR

Le tableau 7.2 donne les paramètres de quatre LFSR tirés de [21], qui lorsque combinés produisent un générateur maximalement équidistribué et «sans-collision». La période du LFSR combiné est maximale, et vaut $(2^{31}-1)(2^{29}-1)(2^{28}-1)(2^{25}-1) \approx 2^{113}$. Chacune des composantes suit la récurrence $x_{j,n} = (x_{j,n-r_j} + x_{j,n-k_j}) \bmod 2$, et produit la sortie $u_{j,n} = \sum_{i=1}^{L} x_{ns_j+i-1}2^{-i}$, avec $L = 32$. Les paramètres choisis permettent d'utiliser l'algorithme «QuickTaus» [15, 21], pour produire des nouveaux états $x_{j,n}$ à partir des états précédents. Comme nous pourrons le constater, cet algorithme est extrêmement efficace.

La figure 7.2 donne une implantation en C du LFSR combiné, basé sur l'algorithme «QuickTaus». Les opérations effectuées consistent principalement en des décalages de bits (>> ou <<), des «et» bit à bit (&) et des ou-exclusifs bit à bit (^). Au

```
#define norm  2.328306549295728e-10 /* 1/(m1+1) */
#define m1   4294967087.0
#define m2   4294944443.0
#define a12     1403580.0
#define a13n     810728.0
#define a21      527612.0
#define a23n    1370589.0

double s10, s11, s12, s20, s21, s22;

double MRG32k3a ()
  {
  register long   k;
  register double p1, p2;
  /* Composante 1 */
  p1 = a12 * s11 - a13n * s10;
  k = (long)(p1 / m1);   p1 -= k * m1;   if (p1 < 0.0) p1 += m1;
  s10 = s11;   s11 = s12;   s12 = p1;
  /* Composante 2 */
  p2 = a21 * s22 - a23n * s20;
  k  = (long)(p2 / m2);  p2 -= k * m2;   if (p2 < 0.0) p2 += m2;
  s20 = s21;   s21 = s22;   s22 = p2;
  /* Combinaison */
  if (p1 <= p2) return ((p1 - p2 + m1) * norm1);
  else return ((p1 - p2) * norm1);
  }
```

Figure 7.1: [19] Implantation en C d'un MRG à deux composantes

Tableau 7.2: Paramètres d'un bon LFSR combiné à quatre composantes

j	k_j	r_j	s_j
1	31	6	18
2	29	27	2
3	28	15	7
4	25	22	13

départ, les variables $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ et $\mathbf{z}_4$ doivent être initialisées pour contenir les valeurs des k_j premiers bits de chacune des récurrences. L'algorithme «QuickTaus» permet ensuite de générer les vecteurs $(x_{j,ns_j}, ..., x_{j,ns_j+L-1})$ à partir des vecteurs précédents $(x_{j,(n-1)s_j}, ..., x_{j,(n-1)s_j+L-1})$. Ces nouvelles valeurs se retrouvent dans les vecteurs $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ et $\mathbf{z}_4$ à la fin du code. Des ou-exclusifs sont finalement effectués pour combiner les quatre vecteurs obtenus, et un résultat normalisé entre 0 et 1 est retourné. Cet algorithme ne sera pas discuté plus en détails ici, puisqu'il est très bien décrit dans [15].

7.1.3 Générateur cubique

Le tableau 7.3 donne les paramètres de deux générateurs cubiques tirés de [22, 27]. Les récurrences ont une forme $x_{j,n} = (a_j x_{j,n-1}^3 + 1) \bmod m_j$ simplifiée, qui permet une implantation plus efficace. En combinant les deux composantes présentées, une période maximale de $m_1 \times m_2 \approx 2^{36}$ est atteinte. Évidemment, il n'est pas conseillé d'utiliser ce générateur combiné en pratique, à cause de sa petite période. Une période plus importante pourrait être obtenue si des composantes individuelles de plus grandes tailles étaient utilisées, ou encore si un plus grand nombre de composantes étaient combinées. Cependant, comme expliqué au chapitre 4, la recherche de bons paramètres permettant d'obtenir une période maximale de m_j devient très difficile et coûteuse, à mesure que la taille de m_j croît. La première solution proposée n'est donc pas réalisable. Aussi, nous allons voir que le générateur combiné présenté est déjà lent. Il n'est donc pas réaliste de vouloir ajouter une autre composante à la combinaison. Néanmoins, l'implantation proposée, même si elle peut difficilement être utilisée telle quelle en pratique, nous sera utile pour produire des nombres avec les nouveaux générateurs

```
#define norm 2.3283064365387e-10 /* 1/(2^32) */
#define s1 18
#define s2 2
#define s3 7
#define s4 13
#define q1 6      /* k1-r1 */
#define q2 2      /* k2-r2 */
#define q3 13     /* k3-r3 */
#define q4 3      /* k4-r4 */
#define p1 13     /* k1-s1 */
#define p2 27     /* k2-s2 */
#define p3 21     /* k3-s3 */
#define p4 12     /* k4-s4 */
#define C1 4294967294 /* masque de k1 1 suivi de L-k1 0 */
#define C2 4294967288 /* masque de k2 1 suivi de L-k2 0 */
#define C3 4294967280 /* masque de k3 1 suivi de L-k3 0 */
#define C4 4294967168 /* masque de k4 1 suivi de L-k4 0 */

unsigned long z1, z2, z3, z4;

double lfsr113 ()
  {
  unsigned long b;
  b  = (((z1 << q1) ^ z1) >> p1);
  z1 = (((z1 & C1) << s1) ^ b);
  b  = (((z2 << q2) ^ z2) >> p2);
  z2 = (((z2 & C2) << s2) ^ b);
  b  = (((z3 << q3) ^ z3) >> p3);
  z3 = (((z3 & C3) << s3) ^ b);
  b  = (((z4 << q4) ^ z4) >> p4);
  z4 = (((z4 & C4) << s4) ^ b);
  return ((z1 ^ z2 ^ z3 ^ z4) * norm);

  }
```

Figure 7.2: [21] Implantation en C d'un LFSR à quatre composantes

Tableau 7.3: Paramètres d'un générateur cubique combiné de deux composantes

m_1	a_1	m_2	a_2
262130	168686	262130	173782

combinés dont nous discuterons.

La figure 7.3 donne une implantation possible du générateur cubique combiné, lorsque la combinaison est faite en additionnant les sorties des deux composantes modulo 1. Les variables `x1` et `x2` contiennent les valeurs de $x_{1,n}$ et $x_{2,n}$ courantes, et doivent être initialisées correctement avant le premier appel au générateur. Les étapes suivantes sont faites pour obtenir un nouvel état $x_{j,n} = (a_j x_{j,n-1}^3 + 1) \bmod m_j$:

1. $p_j \leftarrow a_j x_{j,n-1} \bmod m_j$
2. $p_j \leftarrow p_j x_{j,n-1} \bmod m_j$
3. $p_j \leftarrow (1 + p_j x_{j,n-1}) \bmod m_j$
4. $x_{j,n} \leftarrow p_j$.

Comme les paramètres respectent les conditions $|a_j| m_j < 2^{53}$ et $1 + x_{j,n}^2 < 1 + m_j^2 < 2^{53}$, chacune des 3 étapes décrites ont pu être implantées avec l'arithmétique en virgule flottante. L'opération modulo m_j est effectuée correctement, parce que $\lfloor p_j / m_j \rfloor < (1 + m_j^2)/m_j$ peut toujours être représenté exactement avec une variable de type `long`.

Un résultat normalisé entre 0 et 1 est ensuite calculé pour chaque composante, et la combinaison est finalement effectuée. Il est à noter que les sorties retournées par ce générateur peuvent prendre les valeurs 0 ou 1. Pour éviter de générer ces valeurs, les `norm1` et `norm2` doivent être redéfinies à $1/(m_1 + 1)$ et $1/(m_2 + 1)$ respectivement, et quelques lignes de codes supplémentaires doivent être ajoutées pour retourner une sortie de $(\frac{m_1}{m_1+1} + \frac{m_2}{m_2+1}) \bmod 1$ lorsque `p` vaut 0 ou 1. Ces quelques lignes supplémentaires pourraient réduire légèrement la vitesse de génération des nombres.

```
#define norm1 3.814857343409643e-06 /* 1/m1 */
#define norm2 3.815293987478205e-06 /* 1/m2 */
#define m1   262133.0
#define m2   262103.0
#define a1   168686.0
#define a2   173782.0

double  x1, x2;

double combcub36 ()
  {
  register long k;
  register double p1, p2, p;
  /* Composante 1 */
  p1 = a1 * x1;
  k = (long)(p1 / m1);   p1 -= k * m1;
  p1 = p1 * x1;
  k = (long)(p1 / m1);   p1 -= k * m1;
  p1 = 1 + p1 * x1;
  k = (long)(p1 / m1);   p1 -= k * m1;
  x1 = p1;
  p1 = p1 * norm1;
  /* Composante 2 */
  p2 = a2 * x2;
  k = (long)(p2 / m2);   p2 -= k * m2;
  p2 = p2 * x2;
  k = (long)(p2 / m2);   p2 -= k * m2;
  p2 = 1 + p2 * x2;
  k = (long)(p2 / m2);   p2 -= k * m2;
  x2 = p2;
  p2 = p2 * norm2;
  /* Combinaison */
  p = (p1 + p2);
  if (p >= 1.0) return (p - 1.0);
  else return p;
  }
```

Figure 7.3: Implantation en C d'un générateur cubique à deux composantes

7.1.4 Générateur inversif

Le générateur implanté à la figure 7.4 est de type inversif explicite, et suit la récurrence $x_n = (an)^{-1}$, où l'inverse est calculé modulo m. Les valeurs de a et m, tirées de [22], sont 123 et $2^{31} - 1$ respectivement, et la période maximale $m = 2^{31} - 1$ est obtenue. Cette période est très petite, et il n'est pas conseillé d'utiliser ce générateur en pratique. La période est limitée, dans l'implantation donnée, par le fait que tous les calculs doivent être faits avec des entiers, possiblement négatifs. Les résultats intermédiaires doivent être conservés dans des variables de type `long`, et ne doivent pas dépasser 2^{31}. Ceci est possible seulement si le module m est lui-même inférieur à 2^{31}. Nous avons donc choisi le plus grand module premier possible, soit $m = 2^{31} - 1$. Nous verrons que même avec ce petit module, les calculs sont extrêmement lents.

Le code indiqué à la figure 7.4 est divisé en deux parties. La première fonction permet de calculer l'inverse modulo m d'un entier x, à l'aide de l'algorithme d'Euclide. Cet algorithme est expliqué en détails dans [9], et constitue la partie la plus lente de la génération de nombres à l'aide des méthodes inversives. La deuxième fonction correspond au générateur lui-même. Les valeurs de an sont conservées dans la variable `y`, qui n'a pas besoin d'être initialisée par l'usager si une valeur de départ de 0 est souhaitée. La valeur de `y` est incrémentée à chaque appel à la fonction, et l'inverse modulo m est calculé. Le résultat normalisé est ensuite retourné. Cette fois, une sortie de 0 peut être obtenue, dans le cas où an est un multiple de m. Pour éviter cette valeur, il suffit de redéfinir la constante `norm` par $1/(m+1)$ plutôt que $1/m$, et de retourner $m/(m+1)$ lorsqu'une valeur de `y` $= 0$ est rencontrée. Ce petit changement ne devrait pas modifier la vitesse de génération des nombres.

7.1.5 Temps d'exécution

Les temps de CPU nécessaires pour générer et additionner 10^7 nombres produits par les quatres générateurs proposés sont indiqués au tableau 7.5. Les états `s10`, `s11`, `s12`, `s20`, `s21` et `s22` du MRG et les états `z1`, `z2`, `z3`, `z4` du LFSR ont été initialisés

```
#define a 123
#define m 2147483647
#define mma 2147483524 /* m - a */
#define norm 4.656612875245797e-10  /* 1/m */

long InvEuclid(long x)
  {
  long u1, u3, v1, v3, t1, t3, qq;

  u1 = 0;  u3 = m;
  v1 = 1;  v3 = x;
  while (v3 != 0){
    qq = u3/v3;
    t1 =  u1 - v1*qq;
    t3 = u3 - v3*qq;
    u1 = v1;
    u3 = v3;
    v1 = t1;
    v3 = t3;
    }
  if (u1 < 0)  u1 = u1 + m;
  return  u1;
  }

long y = 0;

double invexp()
  {
  if (y > mma) y = (y - m) + a;
  else y = y + a;
  if (y == 0) return 0.0;
  else return (InvEuclid(y) * norm);
  }
```

Figure 7.4: Implantation en C d'un générateur inversif explicite

Tableau 7.4: Compilateurs et options de compilation utilisés pour les tests de vitesse

Machine	Compilateur	options
AMD Athlon	`gcc`	`-O3 -ffast-math -fexpensive-optimizations -malign-double -s -fomit-frame-pointer -funroll-loops -fschedule-insns2 -mwide-multiply`
Pentium III	`g++`	`-O3 -ffast-math -fexpensive-optimizations -finline-functions`
SUN Ultra-2	`cc`	`-fast -xtarget=ultra -xarch=v8plusa`

à 12345, les états `x1`, `x2` du générateur cubique ont été initialement mis à 123, et la valeur initiale de défaut `y` = 0 du générateur inversif a été conservée. Les sommes des 10^7 premiers nombres obtenus avec ces initialisations ont été ajoutées au tableau.

Les temps donnés ont été calculés sur 4 machines différentes : un AMD Athlon 750 MHz, un Pentium III 500 MHz, un SUN Ultra-2 ayant deux processeurs 200 MHz, et un Pentium II 266 MHz. Les systèmes d'exploitation des machines sont respectivement Linux, Linux, Unix (Solaris) et Windows. Les compilateurs et les options de compilations les plus rapides pour les 3 premières machines ont été sélectionnés, et sont indiqués au tableau 7.4. Dans le cas du Pentium II roulant sous Windows, le code a été compilé avec Visual C++, avec les options «optimisation pour Pentium Pro» et «optimisation en temps» sélectionnées.

Nous pouvons constater que l'algorithme «QuickTaus» utilisé pour générer des nouveaux états avec le LFSR combiné est extrêmement efficace, et ce sur toutes les machines. Ainsi, seulement 0.6 à 1.5 secondes sont requises pour générer et additionner 10^7 nombres avec le générateur `lfsr113`. Le MRG est environ 4 fois plus lent que le LFSR, mais demeure très efficace. Quant au générateur cubique, il est généralement entre 2.2 et 2.4 fois plus lent que le MRG, même si une beaucoup plus petite période a été considérée. Notons une exception cependant : le compilateur `cc` de SUN a très bien optimisé le code du générateur `combcub36`, permettant de l'exécuter à une vitesse

Tableau 7.5: Temps nécessaire pour générer et additionner 10^7 nombres (en secondes) avec des générateurs connus

Générateurs	AMD Athlon 750 MHz	Pentium III 500 MHz	SUN Ultra-2 2 × 200 MHz	Pentium II 266 MHz	Somme
MRG32k3a	2.3	5.6	3.2	8.5	5001090.95
lfsr113	0.6	1.3	0.8	1.5	5001546.72
combcub36	5.1	13.2	3.7	20.2	4999034.98
invexp	14.1	24.4	65.0	33.3	4996711.91

presque comparable à celle du MRG. Le générateur inversif est finalement le plus lent sur toutes les machines, pouvant prendre jusqu'à 65 secondes pour produire et additionner ses 10^7 nombres.

7.2 Générateurs combinant des récurrences différentes

Voici maintenant quelques suggestions de générateurs combinants des récurrences différentes. Dans un premier temps, nous combinons le MRG et le LFSR présentés aux sections 7.1.1 et 7.1.2 avec un deuxième générateur non linéaire selon les méthodes décrites au chapitre 5. Afin d'éviter d'utiliser directement des algorithmes plutôt lents comme `combcub36` ou `invexp`, les implantations proposées se servent de tableaux déjà initialisés contenant une suite de nombres produite par le générateur non linéaire désiré. Une implantation très efficace peut ainsi être obtenue, mais une quantité de mémoire proportionnelle à la taille du deuxième générateur utilisé est requise. Une solution alternative proposée est de combiner des MRG et des LFSR, deux types de générateurs très rapides. Il devient alors possible de choisir deux composantes de tailles équivalentes, et de profiter au maximum des gains empiriques observés au chapitre précédent.

```
#define longtab 2039   /* periode du generateur 2 */

int iter;
double gen2[longtab];

double MRG32k3a_gen2()
  {
  double p;
  p = MRG32k3a() + gen2[iter++];
  if (iter == longtab) iter = 0;
  if (p >= 1.0) return (p - 1.0);
  else return p;
  }
```

Figure 7.5: Implantation en C d'un MRG combiné avec un deuxième générateur

7.2.1 Combinaisons de récurrences linéaires et non linéaires

Les figures 7.5 et 7.6 suggèrent une implantation rapide permettant de combiner respectivement le bon MRG de la section 7.1.1 et le bon LFSR de la section 7.1.2 avec un deuxième générateur quelconque. Les implantations données évitent de faire appel au deuxième générateur directement, à chaque fois qu'un nouveau nombre est demandé. Un tableau `gen2` est plutôt utilisé pour conserver en mémoire la suite de nombres consécutifs produite par le deuxième générateur sur sa période entière, à partir d'un germe initial donné. Une fois une case de départ sélectionnée dans le tableau, il suffit d'aller chercher successivement en mémoire les valeurs déjà calculées pour retrouver une séquence pseudo-aléatoire produite par le deuxième générateur.

Le premier algorithme présenté additionne modulo 1 une sortie produite par le `MRG32k3a` déjà décrit avec l'élément approprié du tableau `gen2`. Avant le premier appel au générateur, le tableau `gen2` doit être correctement initialisé, pour contenir la séquence de nombres entre 0 et 1 désirée. Le générateur `MRG32k3a` doit aussi être initialisé, et une valeur entière entre 0 et `longtab-1` doit être assignée à la variable `iter`, qui indique la prochaine case du tableau `gen2` à considérer.

Le deuxième algorithme combine les sorties produites par le `lfsr113` déjà décrit avec les sorties du tableau `gen2` à l'aide d'un ou-exclusif bit à bit. Pour pouvoir effectuer correctement le ou-exclusif sur les 32 premiers bits de chacun des nombres, les

```
#define longtab 2039
#define norm 2.3283064365387e-10 /* 1/(2^32) */

int iter;
unsigned long gen2[longtab];

double lfsr113_gen2()
  {
  unsigned long z;
  z = lfsr113ul() ^ gen2[iter++];
  if (iter == longtab) iter = 0;
  return (z * norm);
  }
```

Figure 7.6: Implantation en C d'un LFSR combiné avec un deuxième générateur

sorties ont été préalablement multipliées par 2^{32}. Ainsi, la fonction `lfsr113ul` utilisée est identique à `lfsr113`, mais la dernière ligne `return ((z1^z2^z3^z4)*norm)` a été remplacée par `return (z1^z2^z3^z4)`, et un nombre de type `unsigned long` est retourné. Le tableau `gen2` contient aussi des nombres de type `unsigned long`, obtenus en multipliant par 2^{32} les valeurs produites par le deuxième générateur. Le ou-exclusif peut être fait directement sur les sorties obtenues, et une division par 2^{32} permet de ramener le résultat entre 0 et 1. En plus du tableau `gen2`, le générateur `lfsr113ul` et la variable `iter` doivent être initialisés correctement avant le premier appel au générateur.

Il est à noter que la valeur 1 ne sera jamais retournée par les deux générateurs présentés, mais qu'une sortie de 0 peut parfois être obtenue. Un léger ajustement similaire à celui présenté pour le générateur cubique combiné de la section 7.1.3 permettrait d'éviter cette valeur 0, si désiré.

Les deux algorithmes donnés ont l'avantage de demander très peu d'opérations supplémentaires, par rapport à un appel à `MRG32k3a` ou à `lfsr113`. Ils risquent donc d'être très rapides. Cependant, une quantité de mémoire proportionnelle à la taille du deuxième générateur est nécessaire, ce qui peut être contraignant lorsque la deuxième composante devient importante. Néanmoins, des petits générateurs non linéaires de périodes entre 2^{10} et 2^{18} devraient pouvoir être utilisés, et améliorer de façon non négligeable la performance du générateur combiné face aux tests statistiques.

```
double MRG32k3a_lfsr113()
  {
  double p;
  p = MRG32k3a() + lfsr113();
  if (p >= 1.0) return (p - 1.0);
  else return p;
}
```

Figure 7.7: Implantation en C d'un MRG additionné à un LFSR modulo 1

Des exemples de bons petits générateurs cubiques et inversifs sont donnés à l'annexe A. Nous suggérons d'utiliser les générateurs cubiques de paramètres (m_1, a_1) donnés au tableau 9.5, ou des générateurs inversifs de même module m_1 et de paramètre $a = 123$. Avec ces choix de paramètres, les combinaisons avec le `MRG32k3a` ou le `lfsr113` donnent une période maximale égale au produit des périodes de chacune des composantes. De plus, des implantations similaires à celles de `combcub36` et `invexp` peuvent être utilisées pour générer les nombres devant servir à l'initialisation du tableau `gen2`.

7.2.2 Combinaisons de MRG et de LFSR

Les combinaisons de MRG et de LFSR ont aussi donné de bons résultats aux tests empiriques du chapitre précédent. Comme ces deux types de générateurs sont rapides, des générateurs combinés relativement efficaces devraient pouvoir être obtenus, sans avoir besoin de conserver certaines valeurs dans un tableau. La taille de la deuxième composante ne devrait donc plus être aussi limitée, et des gains considérables pourront être obtenus face aux tests statistiques.

Les premières combinaisons suggérées sont présentées aux figures 7.7 et 7.8. Les sorties des générateurs déjà connus `MRG32k3a` et `lfsr113` sont combinées respectivement par une addition modulo 1, puis par un ou-exclusif bit à bit. Les implantations proposées sont très similaires à celles de la section précédente, la seule différence étant que le tableau `gen2` a été remplacé par les fonctions `lfsr113` et `DeuxExp32 * MRG32k3a`, respectivement. Les générateurs combinés ainsi formés ont une très longue période, égale au produit des périodes du MRG et du LFSR, soit $\rho \approx 2^{304}$. Les temps d'exécution

```
#define DeuxExp32   4294967296.0        /* 2^32 */
#define norm        2.3283064365387e-10 /* 1/(2^32) */

double lfsr113_MRG32k3a()
  {
  unsigned long z;
  z = (unsigned long) (DeuxExp32 * MRG32k3a()) ^ lfsr113ul();
  return (z * norm);
}
```

Figure 7.8: Implantation en C d'un LFSR combiné à un MRG avec un ou-exclusif

de ces générateurs devraient être très près de la somme des temps requis pour générer les nombres avec le MRG et le LFSR, puisque très peu d'opérations supplémentaires ne sont nécessaires.

Pour obtenir des meilleurs temps d'exécution, des composantes de plus petites périodes peuvent aussi être combinées. À titre d'exemple, le tableau 7.6 présente un MRG d'ordre 2 de période $m^2-1 \approx 2^{64}$ ayant une bonne valeur de M_{32} au test spectral. Ce MRG peut être implanté avec l'arithmétique en virgule flottante, de façon similaire au `MRG32k3a` présenté. Environ la moitié des opérations peut toutefois être éliminée, puisque seulement une composante est présente.

Tableau 7.6: Paramètres d'un bon MRG d'ordre 2 à une seule composante

m	a_1	a_2
$2^{32}-209$	1268383	-1645506

Un LFSR maximalement équidistribué et sans-collision, tiré de [15] et souvent appelé «taus88», est aussi présenté au tableau 7.7. Ce LFSR, de période maximale $(2^{31}-1)(2^{29}-1)(2^{28}-1) \approx 2^{88}$, peut être implanté exactement comme `lfsr113`. Une composante de moins doit toutefois être calculée, ce qui représente une économie du quart des opérations.

Beaucoup de temps risque d'être sauvé en remplaçant les `MRG32k3a` et `lfsr113` par ces deux générateurs, et la combinaison permet d'obtenir une période d'environ $2^{64+88}/3 \approx 2^{150}$ assez longue pour être utilisée en pratique. Dans les tests de vitesse qui

Tableau 7.7: Paramètres d'un bon LFSR combiné à trois composantes

j	k_j	r_j	s_j
1	31	18	12
2	29	27	4
3	28	25	17

suivent, nous désignerons par «MRG32k2_taus88» le générateur combinant les sorties avec une addition modulo 1, et par «taus88_MRG32k2» celui utilisant un ou-exclusif bit à bit.

7.2.3 Temps d'exécution

Voici maintenant les temps de CPU nécessaires pour générer et additionner 10^7 nombres, avec les différents générateurs combinés proposés. Les mêmes machines, compilateurs et options de compilation qu'à la section 7.1.5 ont été choisis.

Le tableau 7.8 donne les temps pour les générateurs combinés à l'aide d'une addition modulo 1. La vitesse du `MRG32k3a` est indiquée, à titre de comparaison. Trois tailles de tableau `gen2` ont été essayées avec l'algorithmes `MRG32k3a_gen2`, soient $1019 \approx 2^{10}$, $16361 \approx 2^{14}$ et $262133 \approx 2^{18}$. Les tailles choisies correspondent aux périodes des

Tableau 7.8: Temps nécessaire pour générer et additionner 10^7 nombres (en secondes) avec les générateurs combinés à l'aide d'une addition modulo 1

Générateurs	Taille gen2	AMD Athlon 750 MHz	Pentium III 500 MHz	SUN Ultra-2 2 × 200 MHz	Pentium II 266 MHz
MRG32k3a	-	2.3	5.6	3.3	8.5
MRG32k3a_gen2	2^{10}	2.7	6.1	3.8	9.4
	2^{14}	2.7	6.1	3.8	9.5
	2^{18}	2.8	6.5	3.8	10.0
MRG32k2_taus88	-	2.0	4.5	3.0	5.8
MRG32k3a_lfsr113	-	3.5	7.9	5.1	10.5

Tableau 7.9: Temps nécessaire pour générer et additionner 10^7 nombres (en secondes) avec les générateurs combinés à l'aide d'un ou-exclusif

Générateurs	Taille gen2	AMD Athlon 750 MHz	Pentium III 500 MHz	SUN Ultra-2 2 × 200 MHz	Pentium II 266 MHz
lfsr113	-	0.6	1.3	0.8	1.5
lfsr113_gen2	2^{10}	0.6	1.7	1.9	1.5
	2^{14}	0.7	1.8	1.9	1.5
	2^{18}	0.7	1.9	2.0	1.9
taus88_MRG32k2	-	2.2	5.4	3.5	7.5
lfsr113_MRG32k3a	-	3.5	8.3	5.6	12.3

générateurs cubiques ou inversifs suggérés, décrits au tableau 9.5 de l'annexe A. Les temps obtenus avec les deux générateurs combinant un MRG avec un LFSR sont aussi donnés.

Nous pouvons constater que les combinaisons faites à l'aide du tableau `gen2` sont très efficaces, ne demandant pas beaucoup plus de temps que le `MRG32k3a` pour produire 10^7 nombres. Aussi, la taille du tableau utilisé affecte peu la vitesse de génération des nombres. Il est donc suggéré de prendre le plus grand générateur non linéaire possible, en fonction de la mémoire disponible. Les combinaisons de MRG et de LFSR offrent un rapport vitesse d'exécution/période encore plus intéressant, le `MRG32k2_taus88` permettant même de générer des nombres plus rapidement que le `MRG32k3a`. La qualité des points qu'ils produisent, en plus de leur vitesse de génération, en font des générateurs de premier choix.

Le tableau 7.9 permet finalement de comparer les temps obtenus avec les générateurs combinés à l'aide d'un ou-exclusif. Les combinaisons utilisant un tableau `gen2` sont encore une fois très efficaces, permettant même de générer des nombres aussi rapidement qu'avec le `lfsr113` de référence, sur les machines AMD Athlon et Pentium II. Si une quantité suffisante de mémoire est disponible, il est donc préférable d'utiliser ces générateurs, à cause des bonnes propriétés de leurs points. Les combinaisons de LFSR avec des MRG donnent maintenant des temps nettement plus lents que le `lfsr113`,

puisque les temps de générations des MRG sont plus élevés. Ces générateurs semblent donc moins attrayants, mais leur utilisation pourrait être justifiée dans certaines situations où une plus grande robustesse est désirée.

Chapitre 8

Conclusion

Dans ce mémoire, nous nous sommes intéressés aux générateurs combinant des récurrences de différents types, en portant une attention particulière aux combinaisons de récurrences linéaires et non linéaires. Des résultats généraux concernant deux façons de combiner des générateurs, soient par l'addition des sorties modulo 1 et par un ou-exclusif bit à bit des sorties, ont été donnés. La structure de l'ensemble des vecteurs formés de t valeurs successives produites par deux types de générateurs combinés particuliers a été présentée. Nous nous sommes d'abord intéressés à l'addition modulo 1 des sorties d'un MRG et d'un deuxième générateur quelconque, puis nous avons examiné la combinaison d'un LFSR avec un autre générateur au moyen d'un ou-exclusif bit à bit. Dans les deux cas, nous avons montré que les points produits forment une structure moins régulière que la structure connue des générateurs linéaires utilisés. Nous avons aussi montré qu'un bon niveau d'uniformité pouvait être conservé, à condition que les points du générateur linéaire utilisé soient eux-mêmes bien distribués.

Nous avons ensuite formé plusieurs familles de générateurs combinés, la plupart ayant une composante linéaire (un MRG ou un LFSR) et une petite composante non linéaire. Nous avons aussi essayé quelques combinaisons de MRG et de LFSR. Plusieurs tests statistiques considérés comme «costauds» ont été appliqués, et nous avons cherché à identifier à partir de quelle fraction de la période les générateurs d'une même famille commencent à éprouver des difficultés. Les combinaisons proposées ont permis de faire des gains non négligeables, par rapport aux MRG ou aux LFSR combinés entre eux, et ce pour tous les tests appliqués.

Nous avons finalement suggéré quelques bons générateurs combinés, et discuté de

leur implantation. Des temps d'exécution comparables à ceux de MRG ou de LFSR combinés considérés comme rapides ont pu être obtenus dans certains cas.

Les combinaisons de familles différentes étudiées dans ce mémoire ont permis d'obtenir de bons résultats, et risquent d'être d'un grand intérêt pour plusieurs applications en simulation, puisqu'elles offrent une alternative plus «robuste» aux MRG ou LFSR traditionnellement utilisés.

Des travaux supplémentaires auraient avantage à être faits, pour compléter notre compréhension de ces générateurs. Entre autres, il serait intéressant de développer des critères de sélection plus faciles à appliquer que ceux mentionnés au chapitre 5. Ces critères pourraient permettre de sélectionner les paramètres du deuxième générateur à combiner, de façon à garantir le meilleur niveau d'uniformité possible.

Une plus grande variété de combinaisons pourrait aussi être étudiée. Par exemple, les LFSR pourraient être remplacés par des GFSR [13, 16, 18] ou des «twisted GFSR» [13, 16, 18, 34]. Des générateurs G_2 encore plus complexes pourraient aussi être utilisés, si l'implantation utilisant un tableau pour conserver les valeurs du deuxième générateur est envisagée. Plusieurs générateurs G_2 pourraient même être combinés entre eux, par exemple en conservant plusieurs tableaux en mémoire. Cette implantation pourrait permettre d'obtenir un deuxième générateur de période importante, même si chacune des composantes (et donc chacun des tableaux) demeurent relativement petites.

Ensuite, une plus grande variété de tests statistiques pourraient être appliqués aux différentes familles de générateurs. Nous avons dû restreindre le nombre de tests effectués, à cause de la quantité importante de familles testées, et du temps et du travail nécessaire pour appliquer chacun des tests. Il serait toutefois utile d'en appliquer d'autres, pour s'assurer que les bonnes performances obtenues peuvent être généralisées.

Finalement, des choix de paramètres spécifiques devraient être proposés. Les générateurs retenus devraient être retestés à fond, pour s'assurer du bon comportement de leurs points.

Bibliographie

[1] BLUM, L., M. BLUM et M. SCHUB, «A simple unpredictable pseudo-random number generator», *SIAM Journal on Computing*, vol. 15, no. 2, 1986, pp. 364–383.

[2] BOYAR, J., «Inferring sequences produced by a linear congruential generator missing low-order bits», *Journal of Cryptology*, vol. 1, no. 3, 1989, pp. 177–184.

[3] BOYAR, J., «Inferring sequences produced by pseudo-random number generators», *Journal of the ACM*, vol. 36, no. 1, 1989, pp. 129–141.

[4] BRATLEY, P., B. L. FOX et L. E. SCHRAGE, *A Guide to Simulation*, deuxième ed. Springer-Verlag, New York, 1987.

[5] BROWN, M. et H. SOLOMON, «On combining pseudorandom number generators», *Annals of Statistics*, vol. 1, 1979, pp. 691–695.

[6] DENG, L.Y., D.K.J. LIN, J. WANG et Y. YUAN, «Statistical justification of combination generators», *Statistica Sinica*, vol. 7, 1997, pp. 993–1003.

[7] EICHENAUER-HERRMANN, J. et E. HERRMANN, «Compound cubic congruential pseudorandom numbers», *Computing*, vol. 59, 1997, pp. 85–90.

[8] EICHENAUER-HERRMANN, J., E. HERRMANN et S. WEGENKITTL. «A survey of quadratic and inversive congruential pseudorandom numbers». Dans *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing* (New York, 1997), P. Hellekalek, G. Larcher, H. Niederreiter, et P. Zinterhof, Eds., vol. 127 de *Lecture Notes in Statistics*, Springer, pp. 66–97.

[9] KNUTH, D. E., *The Art of Computer Programming, Volume 2 : Seminumerical Algorithms*, troisième ed. Addison-Wesley, Reading, Mass., 1998.

[10] LAW, A. M. et W. D. KELTON, *Simulation Modeling and Analysis*, troisième ed. McGraw-Hill, New York, 2000.

[11] L'ECUYER, P., «Random numbers for simulation», *Communications of the ACM*, vol. 33, no. 10, 1990, pp. 85–97.

[12] L'ECUYER, P. «Testing random number generators». Dans *Proceedings of the 1992 Winter Simulation Conference* (Dec 1992), IEEE Press, pp. 305–313.

[13] L'ECUYER, P., «Uniform random number generation», *Annals of Operations Research*, vol. 53, 1994, pp. 77–120.

[14] L'ECUYER, P., «Combined multiple recursive random number generators», *Operations Research*, vol. 44, no. 5, 1996, pp. 816–822.

[15] L'ECUYER, P., «Maximally equidistributed combined Tausworthe generators», *Mathematics of Computation*, vol. 65, no. 213, 1996, pp. 203–213.

[16] L'ECUYER, P. «Random number generation». Dans *Handbook of Simulation*, J. Banks, Ed. Wiley, 1998, pp. 93–137.

[17] L'ECUYER, P. «Random number generators and empirical tests». Dans *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, P. Hellekalek, G. Larcher, H. Niederreiter, et P. Zinterhof, Eds., vol. 127 de *Lecture Notes in Statistics*. Springer, New York, 1998, pp. 124–138.

[18] L'ECUYER, P. «Uniform random number generators». Dans *Proceedings of the 1998 Winter Simulation Conference* (Dec 1998), IEEE Press, pp. 97–104.

[19] L'ECUYER, P., «Good parameters and implementations for combined multiple recursive random number generators», *Operations Research*, vol. 47, no. 1, 1999, pp. 159–164.

[20] L'ECUYER, P., «Tables of linear congruential generators of different sizes and good lattice structure», *Mathematics of Computation*, vol. 68, no. 225, 1999, pp. 249–260.

[21] L'ECUYER, P., «Tables of maximally equidistributed combined LFSR generators», *Mathematics of Computation*, vol. 68, no. 225, 1999, pp. 261–269.

[22] L'ECUYER, P. «TestU01 : Un logiciel pour appliquer des tests statistiques à des générateurs de valeurs aléatoires». En préparation, Circa 2000.

[23] L'ECUYER, P., A. COMPAGNER et J.-F. CORDEAU. «Entropy tests for random number generators». Manuscrit, 1996.

[24] L'ECUYER, P., J.-F. CORDEAU et R. SIMARD, «Close-point spatial tests and their application to random number generators», *Operations Research*, vol. 48, no. 2, 2000, pp. 308–317.

[25] L'ECUYER, P. et S. CÔTÉ, «Implementing a random number package with splitting facilities», *ACM Transactions on Mathematical Software*, vol. 17, no. 1, 1991, pp. 98–111.

[26] L'ECUYER, P. et R. COUTURE, «An implementation of the lattice and spectral tests for multiple recursive linear random number generators», *INFORMS Journal on Computing*, vol. 9, no. 2, 1997, pp. 206–217.

[27] L'ECUYER, P. et P. HELLEKALEK. «Random number generators : Selection criteria and testing». Dans *Random and Quasi-Random Point Sets*, P. Hellekalek et G. Larcher, Eds., vol. 138 de *Lecture Notes in Statistics*. Springer, New York, 1998, pp. 223–265.

[28] L'ECUYER, P. et F. PANNETON. «A new class of linear feedback shift register generators». Dans *Proceedings of the 2000 Winter Simulation Conference* (Pistacaway, NJ, Dec 2000), J. A. Joines, R. R. Barton, K. Kang, et P. A. Fishwick, Eds., IEEE Press, pp. 690–696.

[29] L'ECUYER, P. et R. SIMARD, «Beware of linear congruential generators with multipliers of the form $a = \pm 2^q \pm 2^r$», *ACM Transactions on Mathematical Software*, vol. 25, no. 3, 1999, pp. 367–374.

[30] L'ECUYER, P. et R. SIMARD, «On the performance of birthday spacings tests for certain families of random number generators», *Mathematics and Computers in Simulation*, vol. 55, no. 1–3, 2001, pp. 131–137.

[31] L'ECUYER, P., R. SIMARD et S. WEGENKITTL. «Sparse serial tests of uniformity for random number generators». Rapport du GERAD G-98-65, 1998.

[32] L'ECUYER, P. et R. TOUZIN. «Fast combined multiple recursive generators with multipliers of the form $a = \pm 2^q \pm 2^r$». Dans *Proceedings of the 2000 Winter Simulation Conference* (Pistacaway, NJ, Dec 2000), J. A. Joines, R. R. Barton, K. Kang, et P. A. Fishwick, Eds., IEEE Press, pp. 683–689.

[33] MARSAGLIA, G. «A current view of random number generators». Dans *Computer Science and Statistics, Sixteenth Symposium on the Interface* (North-Holland, Amsterdam, 1985), Elsevier Science Publishers, pp. 3–10.

[34] MATSUMOTO, M. et Y. KURITA, «Twisted GFSR generators», *ACM Transactions on Modeling and Computer Simulation*, vol. 2, no. 3, 1992, pp. 179–194.

[35] MATSUMOTO, M. et Y. KURITA, «Twisted GFSR generators II», *ACM Transactions on Modeling and Computer Simulation*, vol. 4, no. 3, 1994, pp. 254–266.

[36] MAURER, U. M., «A universal statistical test for random bit generators», *Journal of Cryptology*, vol. 5, no. 2, 1992, pp. 89–105.

[37] READ, T. R. C. et N. A. C. CRESSIE, *Goodness-of-Fit Statistics for Discrete Multivariate Data.* Springer Series in Statistics. Springer-Verlag, New York, 1988.

[38] TEZUKA, S. et P. L'ECUYER, «Efficient and portable combined Tausworthe random number generators», *ACM Transactions on Modeling and Computer Simulation*, vol. 1, no. 2, 1991, pp. 99–112.

[39] WU, P.-C., «Multiplicative, congruential random number generators with multiplier $\pm 2^{k_1} \pm 2^{k_2}$ and modulus $2^p - 1$», *ACM Transactions on Mathematical Software*, vol. 23, no. 2, June 1997, pp. 255–265.

Annexe A : Paramètres des générateurs testés au chapitre 6

Voici une liste exhaustive des paramètres des générateurs testés au chapitre 6.

Les générateurs de type LCG, LFSR, inversif et cubique sont ceux implantés dans le logiciel ≪TestU01≫ [22]. Les autres familles de générateurs combinés ont été obtenus, pour la plupart, en réutilisant certains de ces paramètres du logiciel ≪TestU01≫. Chaque composante a été choisie avec soin, afin de permettre aux générateurs combinés d'atteindre leur période maximale, correspondant au produit des périodes de chacune des composantes. Notons que nous avons dû chercher de nouveaux LCG avec des périodes différentes, pour permettre aux générateurs combinant LCG et LFSR d'atteindre les périodes désirées.

LCG

Les paramètres des LCG testés sont indiqués au tableau 9.1. Ils suivent tous une récurrence de la forme $x_n = (a_1 x_{n-1}) \bmod m$, et leur période s'approche de 2^e, pour des valeurs de e variant entre 10 et 60.

LFSR

Les LFSR utilisés ont chacun deux composantes, de récurrences $x_{j,n} = (x_{j,n-r_j} + x_{j,n-k_j}) \bmod 2$ et de sorties $u_{j,n} = \sum_{i=1}^{L} x_{ns_j+i-1} 2^{-i}$, où $j = 1$ ou 2 désigne la composante considérée. La sortie du LFSR combiné est obtenue en faisant un ou-exclusif bit à bit entre les sorties $u_{1,n}$ et $u_{2,n}$ de chacune des composantes. La valeur de L a été fixée à 32, et les paramètres k_1, r_1, s_1 et k_2, r_2, s_2 choisis sont indiqués au tableau 9.2.

Les périodes des LFSR combinés sont près de 2^e, pour des entiers e variant entre 10 et 36.

Générateurs inversifs

Les sorties produites par les générateurs inversifs qui ont été utilisés sont données par $u_n = (an + c)^{-1}/m$, où l'inverse est calculé modulo m. Les valeurs de m choisies sont identiques aux modules des LCG présentés, et les paramètres a et c ont été fixés à 123 et 0 respectivement. La période des générateurs ainsi formés est près de 2^e, pour les mêmes valeurs de e données au tableau 9.1.

Générateurs à congruence cubique

Les générateurs cubiques utilisés ont deux composantes, de récurrences $x_{j,n} = (a_j x_{j,n-1}^3 + 1) \bmod m_j$ et de sorties $u_{j,n} = x_{j,n}/m_j$, où $j = 1$ ou 2 indique la composante considérée. La sortie du générateur combiné est obtenue en additionnant modulo 1 les sorties $u_{1,n}$ et $u_{2,n}$ de chacune des composantes. Les paramètres m_1, a_1 et m_2, a_2 choisis sont donnés au tableau 9.3. Les générateurs combinés ont des périodes près de 2^e, pour des entiers e variant entre 12 et 36.

Combinaison de différentes familles de générateurs

Les LCG utilisés dans les combinaisons impliquant un générateur non linéaire et les LFSR utilisés dans toutes les combinaisons sont ceux déjà présentés aux tableaux 9.1 et 9.2. Les nouveaux LCG utilisés dans les combinaisons avec des LFSR sont décrits au tableau 9.4. Ces LCG ont la meilleure valeur de M_8 possible au test spectral, pour les modules m donnés et la restriction $am < 2^{53}$ nécessaire pour l'implantation en virgule flottante. Ils ont des périodes s'approchant de 2^e, pour des entiers e variant entre 10 et 31.

Les générateurs cubiques utilisés dans les combinaisons sont non combinés, et leurs

paramètres ont été choisis parmi les composantes du tableau 9.3. Les paramètres sélectionnés sont différents, selon que la combinaison est faite avec un LCG ou un LFSR. Le tableau 9.5 résume les valeurs retenues. Il est à noter que dans la combinaison avec un LCG de période près de 2^{38}, le générateur cubique de période près de 2^8 a été remplacé par celui utilisé avec les LFSR. Ce petit changement a été nécessaire pour obtenir la période près de 2^{46} désirée pour le générateur combiné.

Quant aux générateurs inversifs, ils ont les mêmes modules que ceux des générateurs cubiques utilisés, et les valeurs de a et c ont été fixées à 123 et 0.

Ces différents paramètres assurent que les générateurs combinés des familles 1 à 7 du tableau 6.1 ont bien des périodes de 2^e, tel que désiré.

Tableau 9.1: Paramètres des LCG testés au chapitre 6

e	m	a_1	e	m	a_1
10	1021	65	36	68719476731	49865143810
11	2039	995	37	137438953447	76886758244
12	4093	209	38	274877906899	17838542566
13	8191	884	39	549755813881	61992693052
14	16381	572	40	1099511627689	1038914804222
15	32749	219	41	2199023255531	1013262675629
16	65521	17364	42	4398046511093	2214813540776
17	131071	43165	43	8796093022151	4114249742626
18	262139	92717	44	17592186044399	6307617245999
19	524287	283741	45	35184372088777	25933916233908
20	1048573	380985	46	70368744177643	63975993200055
21	2097143	360889	47	140737488355213	102306498730560
22	4194301	914334	48	281474976710597	49235258628958
23	8388593	653276	49	562949953421231	265609885904224
24	16777213	6423135	50	1125899906842597	1087141320185010
25	33554393	25907312	51	2251799813685119	349044191547257
26	67108859	26590841	52	4503599627370449	4359287924442956
27	134217689	45576512	53	9007199254740881	2333175048965096
28	268435399	31792125	54	18014398509481951	17554612001638734
29	536870909	16538103	55	36028797018963913	33266544676670489
30	1073741789	5122456	56	72057594037927931	39159994680362565
31	2147483647	1389796	57	144115188075855859	75953708294752990
32	4294967291	1588635695	58	288230376151711717	252847049180516155
33	8589934583	7425194315	59	576460752303423433	346764851511064641
34	17179869143	5295517759	60	1152921504606846883	561860773102413563
35	34359738337	3124199165	-	-	-

Tableau 9.2: Paramètres des LFSR testés au chapitre 6

e	k_1	r_1	s_1	k_2	r_2	s_2
10	7	6	2	3	2	1
11	6	5	1	5	3	2
12	7	6	5	5	3	1
13	7	6	3	6	5	4
14	9	5	3	5	3	2
15	11	9	6	4	3	1
16	11	9	3	5	3	2
17	10	7	5	7	4	3
18	11	9	5	7	6	2
19	10	7	4	9	5	2
20	11	9	8	9	5	4
21	11	9	3	10	7	4
22	15	11	8	7	6	6
23	17	14	10	6	5	1
24	17	14	7	7	6	5
25	18	11	5	7	6	4
26	15	8	4	11	9	7
27	17	12	10	10	7	2
28	17	14	11	11	9	8
29	18	11	10	11	9	6
30	21	19	12	9	5	5
31	20	17	8	11	9	6
32	17	14	12	15	11	6
33	23	18	10	10	7	4
34	23	18	8	11	9	7
35	18	11	11	17	12	8
36	25	18	14	11	9	5

Tableau 9.3: Paramètres des générateurs cubiques testés au chapitre 6

e	m_1	a_1	m_2	a_2
12	59	15	47	7
13	101	61	59	15
14	101	61	83	9
15	251	135	101	61
16	251	135	233	61
17	503	445	251	135
18	503	445	491	277
19	1019	437	503	445
20	1019	437	1013	31
21	2039	243	1019	437
22	2039	243	2027	349
23	4091	494	2039	243
24	4091	494	4079	3757
25	8147	2410	4091	494
26	8147	2410	8111	2257
27	16361	5595	8147	2410
28	16361	5595	16319	3013
29	32693	1190	16361	5595
30	32693	1190	32687	4535
31	65519	512	32693	1190
32	65519	512	65447	27076
33	131063	110230	65519	512
34	131063	110230	130859	48249
35	262133	168686	131063	110230
36	262133	168686	262103	173782

Tableau 9.4: Paramètres des LCG utilisés dans les combinaisons avec des LFSR, dans les tests du chapitre 6

e	m	a_1
10	1019	440
11	2039	995
12	4079	212
13	8147	619
14	16223	10730
15	32603	7513
16	65267	16086
17	130787	57113
18	262127	22725
19	524243	882
20	1048343	377158
21	2097143	360889
22	4194287	341585
23	8388287	1869896
24	16776899	213590
25	33553799	14704570
26	67108187	35190319
27	134217323	912557
28	268435019	12222884
29	536870723	10896430
30	1073740439	848890
31	2147483579	3396173

Tableau 9.5: Paramètres des générateurs cubiques utilisés dans les combinaisons avec des LFSR (m_1, a_1) et des LCG (m_2, a_2), dans les tests du chapitre 6

e	m_1	a_1	m_2	a_2
6	59	15	47	7
7	101	61	83	9
8	251	135	233	61
9	503	445	491	277
10	1019	437	1013	31
11	2039	243	2027	349
12	4091	494	4079	3757
13	8147	2410	8111	2257
14	16361	5595	16319	3013
15	32693	1190	32687	4535
16	65519	512	65447	27076
17	131063	110230	130859	48249
18	262133	168686	262103	173782

Note : Lorsque combiné avec le LCG de période près de 2^{38}, le générateur cubique utilisé est celui de paramètres (m_1, a_1).

Annexe B : Guide d'utilisation des modules programmés

Deux modules ont dû être ajoutés au logiciel «TestU01» [22] utilisé pour appliquer les tests statistiques du chapitre 6. Ces modules ont permis de compléter les familles de générateurs déjà disponibles, pour former toutes les familles dont nous avons eu besoin pour notre étude empirique.

Le premier module, nommé «ugrang», regroupe différentes implantations de générateurs combinés. Les fonctions programmées donnent à l'utilisateur le choix des paramètres à utiliser. Le deuxième module, nommé «tgrang », définit des générateurs non linéaires et combinés spécifiques de période 2^e, pour différents entiers e. Les paramètres sélectionnés correspondent à ceux décrits dans l'annexe précédente. Dans tous les cas, l'appel aux fonctions programmées a pour effet de sélectionner un générateur particulier, qui est ensuite utilisé par les autres modules pour produire des nombres lorsque nécessaire.

Les différentes fonctions ont été implantées en modula-2, le langage majoritairement utilisé dans le reste du logiciel «TestU01». Le guide d'utilisation qui suit présente l'entête de chaque fonction, et y associe une brève description. Des références à d'autres fonctions déjà disponibles sont parfois indiquées, et le lecteur est invité à consulter le guide complet [22] pour obtenir plus de détails sur les modules existants.

ugrang

Ce module implante différents générateurs combinés. Les procédures `Set...` affectent un générateur particulier à la variable globale `Gen`, le générateur courant utilisé pour les tests.

```
DEFINITION MODULE ugrang ;

FROM SUPINT     IMPORT SuperInteger ;

PROCEDURE SetCombLCGInvExpl (m1, a1, c1, s1 : LONGINT ;
                             m2, a2, c2     : LONGINT) ;
```

Combine un générateur LCG de paramètres (m_1, a_1, c_1) et d'état initial s_1 avec un générateur non linéaire inversif de type explicite de paramètres (m_2, a_2, c_2). L'implantation du LCG est celle de `SetLCGFloat` ou de `SetLCG` de `ulcg`, dépendant des paramètres (m_1, a_1, c_1), et l'implantation du générateur inversif est celle de `SetInvExpl` de `uinv`. La combinaison se fait en additionnant modulo 1 les outputs des deux générateurs. Restrictions : les mêmes que pour `SetLCG` et `SetInvExpl`.

```
PROCEDURE SetCombBigLCGInvExpl (m1,  a1, c1, s1 : SuperInteger ;
                                m2, a2, c2      : LONGINT) ;
```

Même chose que `SetCombInvExplLCG`, mais le LCG est implanté en utilisant de grands entiers (`SetBigLCG` de `ulcg`). Restrictions : les mêmes que dans `SetBigLCG` et `SetInvExpl`.

```
PROCEDURE SetCombLCGCub (m1,  a1, c1, s1 : LONGINT ;
                         m2, a2, s2     : LONGINT) ;
```

Combine un générateur LCG de paramètres (m_1, a_1, c_1) et d'état initial s_1 avec un générateur cubique de paramètres (m_2, a_2) et d'état initial s_2. L'implantation du LCG est celle de `SetLCGFloat` ou de `SetLCG` de `ulcg`, dépendant des paramètres (m_1, a_1, c_1), et l'implantation du générateur cubique est celle de `SetCubic1Float` de `ucubic`. La combinaison se fait en additionnant modulo 1 les outputs des deux générateurs. Restrictions : les mêmes que pour `SetLCG` et `SetCubic1Float`.

```
PROCEDURE SetCombBigLCGCub (m1,  a1, c1, s1 : SuperInteger ;
                            m2, a2, s2      : LONGINT) ;
```

Même chose que `SetCombCubLCG`, mais le LCG est implanté en utilisant de grands entiers (`SetBigLCG` de `ulcg`). Restrictions : les mêmes que pour `SetBigLCG` et `SetCubic1Float`.

```
PROCEDURE SetCombTausBigLCG (k1, q1, s1, SS1   : LONGCARD;
                             k2, q2, s2, SS2   : LONGCARD;
                             m, a , c, SS3     : LONGINT);
```

Combine deux générateurs Tausworthe de paramètres (k_1, q_1, s_1) et (k_2, q_2, s_2) et d'états initiaux $SS1$ et $SS2$ avec un LCG de paramètres (m, a, c) et d'état initial $SS3$. La combinaison est obtenue en additionnant les outputs du LCG et du Tausworthe combiné, modulo 1. L'implantation du LCG est celle de `SetBigLCG` de `ulcg` et l'implantation du Tausworthe combiné est celle de `SetCombTaus2` de `utaus`. Restrictions : les mêmes que pour `SetBigLCG` et `SetCombTaus2`.

Même chose que `SetCombCubLCG`, mais le LCG est implanté en utilisant de grands entiers (`SetBigLCG` de `ulcg`). Restrictions : les mêmes que pour `SetBigLCG` et `SetCubic1Float`.

```
PROCEDURE SetCombTausLCG21xor (k1, q1, s1, SS1   : LONGCARD;
                               k2, q2, s2, SS2   : LONGCARD;
                               m, a , c, SS3     : LONGINT);
```

Combine deux générateurs Tausworthe de paramètres (k_1, q_1, s_1) et (k_2, q_2, s_2) et d'états initiaux $SS1$ et $SS2$ avec un LCG de paramètres (m, a, c) et d'état initial $SS3$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs. L'implantation du LCG est celle de `SetLCGFloat` ou de `SetLCG` de `ulcg`, dépendant des paramètres (m, a, c), et l'implantation du Tausworthe combiné est celle de `SetCombTaus2` de `utaus`. Restrictions : les mêmes que pour `SetLCG` et `SetCombTaus2`.

```
PROCEDURE SetCombTausCub21xor (k1, q1, s1, SS1   : LONGCARD;
                               k2, q2, s2, SS2   : LONGCARD;
                               m, a, SS3      : LONGINT);
```

Combine deux générateurs Tausworthe de paramètres (k_1, q_1, s_1) et (k_2, q_2, s_2) et d'états initiaux $SS1$ et $SS2$ avec un générateur cubique de paramètres (m, a) et d'état initial $SS3$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs. L'implantation du Tausworthe combiné est celle de `SetCombTaus2` de `utaus`, et l'implantation du générateur cubique est celle de `SetCubic1Float` de `ucubic`. Restrictions : les mêmes que pour `SetCombTaus2` et `SetCubic1Float`.

```
PROCEDURE SetCombTausInvExpl21xor (k1, q1, s1, SS1   : LONGCARD;
                                   k2, q2, s2, SS2   : LONGCARD;
                                   m, a, c           : LONGINT);
```

Combine deux générateurs Tausworthe de paramètres (k_1, q_1, s_1) et (k_2, q_2, s_2) et d'états initiaux $SS1$ et $SS2$ avec un générateur inversif explicite de paramètres (m, a, c). La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs. L'implantation du Tausworthe combiné est celle de `SetCombTaus2` de `utaus`, et l'implantation

du générateur inversif est celle de `SetInvExpl` de `uinv`. Restrictions : les mêmes que pour `SetCombTaus2` et `SetInvExpl`.

`END ugrang.`

tgrang

Ce module définit des générateurs non linéaires et des générateurs combinés de différentes tailles, de périodes près d'une puissance de 2. Les procédures `Choose...` retournent VRAI si un générateur du type spécifié et de la taille demandée est défini et a été correctement initialisé, et retournent FAUX sinon. Les paramètres utilisés correspondent à ceux décrits à l'annexe A.

```
DEFINITION MODULE tgrang ;
```

```
CONST
  PrecXOR = 31 ;
```

Précision des générateurs combinés en faisant un ou exclusif bit à bit des outputs de chaque générateur individuel. Dans ce cas, la fonction CombGen2xor de unif01 est utilisée pour faire la combinaison, et la sortie a une précision de 31 bits.

```
VAR
  UnSurRatio :  LONGINT ;
```

Certains générateurs sont combinés de telle sorte que le premier générateur a une période près de 2^{e_1} et le deuxième générateur une période près de 2^{e_2}, où $e_1 + e_2 = e$ et $e_2 = \lfloor e/\texttt{UnSurRatio} \rfloor$. Le générateur combiné devrait avoir une période de 2^e. Valeur de défaut : 2.

```
  TailleGen2 :  LONGINT ;
```

Certains générateurs sont combinés de telle sorte que le premier générateur a une période près de 2^{e_1} et le deuxième générateur une période près de 2^{e_2}, où $e_2 =$ `TailleGen2`, et $e_1 + e_2 = e$. Le générateur combiné devrait avoir une période de 2^e. Valeur de défaut : 12.

```
PROCEDURE ChooseCubicOut (e : LONGINT) : BOOLEAN ;
```

Choisit les mêmes LCG que `ChooseGoodLCG`, mais produit une sortie cubique $u_n = (x_n^3 \bmod m)/m$. Restrictions : $10 \leq e \leq 31$.

```
PROCEDURE ChooseCombLCGInvExpl (e : LONGINT) : BOOLEAN ;
```

Choisit un générateur combiné dont la première composante est un LCG et la deuxième est un générateur inversif explicite, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 = \lfloor e/\texttt{UnSurRatio} \rfloor$ et $e_1 + e_2 = e$. La combinaison se fait est additionnant les outputs, modulo 1. Restrictions : $10 \leq e_1 \leq 31$, $6 \leq e_2 \leq 18$.

`PROCEDURE ChooseCombLCGInvExplB (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LCG et la deuxième est un générateur inversif explicite, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où e_2 =`TailleGen2` et $e_1 + e_2 = e$. La combinaison se fait en additionnant les outputs, modulo 1. Restrictions : $10 \leq e_1 \leq 60$, $6 \leq e_2 =$ `TailleGen2` ≤ 18, $e_1 + e_2 \leq 60$.

`PROCEDURE ChooseCombLCGCub (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LCG et la deuxième est un générateur cubique, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 = \lfloor e/$`UnSurRatio`$\rfloor$ et $e_1 + e_2 = e$. La combinaison se fait est additionnant les outputs, modulo 1. Restrictions : $10 \leq e_1 \leq 31$, $6 \leq e_2 \leq 18$.

`PROCEDURE ChooseCombLCGCubB (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LCG et la deuxième est un générateur à congruence cubique, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où e_2 =`TailleGen2` et $e_1 + e_2 = e$. La combinaison se fait en additionnant les outputs, modulo 1. Restrictions : $10 \leq e_1 \leq 60$, $6 \leq e_2 =$ `TailleGen2` ≤ 18, $e_1 + e_2 \leq 60$.

`PROCEDURE ChooseCombLCGTaus2 (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LCG et la deuxième est un LFSR2, dont les modules m_1 et m_2 sont légèrement inférieurs à $2^{e-\lfloor e/2 \rfloor}$ et $2^{\lfloor e/2 \rfloor}$ respectivement. La combinaison se fait en additionnant les outputs, modulo 1. Restrictions : $20 \leq e_1 \leq 60$.

`PROCEDURE ChooseCombLCGTaus2B (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LCG et la deuxième est un LFSR2, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où e_2 =`TailleGen2` et $e_1 + e_2 = e$. La combinaison se fait en additionnant les outputs, modulo 1. Restrictions : $10 \leq e_1 \leq 31$, $10 \leq e_2 =$ `TailleGen2` ≤ 36, $e_1 + e_2 \leq 60$.

`PROCEDURE ChooseTausLCG21xor (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LFSR2 et la deuxième est un LCG, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_1 = \lfloor e/2 \rfloor$ et $e_1 + e_2 = e$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs individuels. Restrictions : $20 \leq e \leq 60$.

`PROCEDURE ChooseTausLCG21xorB (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LFSR2 et la deuxième est un LCG, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 =$ `TailleGen2` et $e_1 + e_2 = e$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs individuels. Restrictions : $10 \leq e_1 \leq 36$, $10 \leq e_2 \leq 31$, $e_1 + e_2 \leq 60$.

`PROCEDURE ChooseTausCub21xor (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LFSR2 et la deuxième est un générateur cubique, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 = \lfloor e/$`UnSurRatio`$\rfloor$ et $e_1 + e_2 = e$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs individuels. Restrictions : $10 \leq e_1 \leq 36$, $6 \leq e_2 \leq 18$.

`PROCEDURE ChooseTausCub21xorB (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LFSR2 et la deuxième est un générateur cubique, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 =$ `TailleGen2` et $e_1 + e_2 = e$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs individuels. Restrictions : $10 \leq e_1 \leq 36$, $6 \leq e_2 \leq 18$.

`PROCEDURE ChooseTausInvExpl21xor (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LFSR2 et la deuxième est un générateur inversif explicite, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 = \lfloor e/$`UnSurRatio`$\rfloor$ et $e_1 + e_2 = e$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs individuels. Restrictions : $10 \leq e_1 \leq 36$, $6 \leq e_2 \leq 18$.

`PROCEDURE ChooseTausInvExpl21xorB (e : LONGINT) : BOOLEAN ;`

Choisit un générateur combiné dont la première composante est un LFSR2 et la deuxième est un générateur inversif explicite, dont les modules m_1 et m_2 sont légèrement inférieurs à 2^{e_1} et 2^{e_2}, où $e_2 =$ `TailleGen2` et $e_1 + e_2 = e$. La combinaison est obtenue en faisant un ou exclusif bit à bit des outputs des générateurs individuels. Restrictions : $10 \leq e_1 \leq 36$, $6 \leq e_2 \leq 18$.

`END tgrang.`