Chapitre 1

Générateurs des nombres au hasard

Many random-number generators in use today are not very good. There is a tendency for people to avoid learning anything about random-number generators; quite often we find that some old method which is comparatively unsatisfactory has blindly been passed from one programmer to another, and todays users have no understanding of its limitations.

Donald E. Knuth : *The art of computer programming*, vol. 2, Addison-Wesley, Reading, Massachusetts (1969).

1.1 Introduction

Toute simulation Monte Carlo fait intervenir, par définition, des nombres au hasard. Les questions pratiques que l'on se pose sont donc :

1. comment générer une suite des nombres qui sont la réalisation d'une suite de variables aléatoires réelles indépendantes $X_1, X_2, \ldots$ et de même loi (donnée) $F(x) = \mathbb{P}(X_i \leq x)$ et, de manière plus fondamentale,
2. si une telle suite des nombres nous est fournie, comment décider si elle est en effet une réalisation fidèle de la loi donnée ?

Les réponses à ces deux questions semblent évidentes : pour simuler une loi il suffit de trouver un phénomène physique bien modélisé par un processus de cette loi et identifier les valeurs successives de la grandeur physique avec la suite des nombres aléatoires cherchée. On peut d'ailleurs limiter la discussion dans le cas de loi uniforme sur l'intervalle $[0, 1]$. On verra dans le paragraphe **??** que des suites selon toute autre loi (raisonnable) peuvent être obtenues à partir de suites des nombres distribués selon la loi uniforme. En ce qui concerne la vérification qu'une suite fournie soit la réalisation de la loi F, il faudrait montrer que les fonctions de répartition empiriques coïncident avec les fonctions de répartition théoriques, par exemple vérifier que

$$\lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}_{[a,b[}(X_n) = F(b^-) - F(a)$$

ou

$$\lim_{N\to\infty} \frac{1}{N(N-1)} \sum_{m,n=1;m\neq n}^{N} \mathbb{1}_{[a_1,b_1[\times[a_2,b_2[}(X_m, X_n) = (F(b_1^-) - F(a_1)) \times (F(b_2^-) - F(a_2))$$

et ainsi de suite pour tous les k-uplets. La réponse à la deuxième question est un peu naïve puisqu'elle oublie que toute suite des nombres au hasard fournie par simulation est finie. Y a-t-il un moyen objectif de décider si une suite finie est réalisation d'une suite de variables aléatoires de loi donnée ? Intuitivement, non !

Exemple 1.1.1 [du « singe dactylographe »] Soit un singe qui sait frapper à la machine à écrire. On peut penser raisonnablement que le singe frappe les touches au hasard. Le texte que le singe tape est donc une réalisation de la loi uniforme sur l'ensemble $\{a, b, \ldots, z\}$. Si le singe frappe indéfiniment, par le lemme de Borel-Cantelli,

$$\mathbb{P}\{\text{l'œuvre complète de V. Hugo apparaît}\} = 1.$$

Or, si sur un feuillet le singe frappe : *« Il neigeait. On était vaincu par sa conquête. Pour la première fois l'aigle baissait la tête... »*, il sera suspecté de ne pas être un singe...

Le but de ce chapitre est de fournir une réponse (partielle) à la première question soulevée plus haut, à savoir comment générer une suite de nombres au hasard. La deuxième question, concernant les propriétés statistiques, sera traitée dans la chapitre **??**.

1.2 Le développement historique

La « pré-histoire » (avant 1940) : les premières suites des variables aléatoires étaient générées par des dispositifs analogiques. Les fluctuations de tension aux bords d'une résistance électrique chauffée alimentée par un courant stabilisé sont bien modélisées par un bruit blanc. Les instants où des coups sont enregistrés par un compteur de rayonnement cosmique suivent une loi exponentielle. Ces phénomènes ont été exploités pour créer des tables des variables aléatoires [**?**]. L'avènement des premiers ordinateurs numériques, a rendu ces méthodes impraticables. D'une part, le stockage sur les premières machines des tables des variables aléatoires occupait beaucoup de mémoire (très chère à l'époque) et, d'autre part, le couplage direct des dispositifs analogiques aux ordinateurs numériques avait l'inconvénient de fournir des suites non reproductibles des nombres au hasard.

La « proto-histoire » (1940–1950) : il est apparu que la génération algorithmique des suites des nombres au hasard était plus efficace. Plusieurs algorithmes ont été proposés pour engendrer de telles suites. Par exemple

1. la suite des digits successifs des nombres transcendants. Si

   $$x - [x] = \langle\, 0, d_1(x)d_2(x)\ldots\rangle_b$$

   est la représentation en base [1] b de la partie fractionnaire de x (on note $[\cdot]$ la partie entière), on sait que pour Lebesgue-presque tout nombre transcendant x, la suite de ses digits est

[1] On note dans la suite $\langle\cdot\rangle_b$ la représentation en base b, la base étant *toujours* représentée en base décimale. On omet la valeur de la base si celle-ci est facilement déduite du contexte.

une suite équi-distribuée ([?], théorème 1.2) sur $\{0, 1, \ldots, b-1\}$. Le problème est que pour un transcendant *donné* on ignore si ceci est vrai. On ignore même si pour les décimaux du nombre π, par exemple, la quantité

$$\frac{1}{n} \text{card}\{i \in \mathbf{N}, i \leq n : d_i(\pi) = k\}$$

tend vers $1/10$ pour tout $k = 0, 1, \ldots, 9$ [?].

2. la méthode de « Procuste »[2] qui consiste, en partant d'un nombre entier à N digits (en base de 10 par exemple) $x = \langle\, d_1 \ldots d_N \,\rangle$, de prendre son carré $x^2 = \langle\, d'_1 \ldots d'_{2N} \,\rangle$ et de former un nouvel entier en ne gardant que les N digits du milieu $x_{\text{nouveau}} = \langle\, d'_{[N/2]} \cdots d'_{[N/2]+N} \,\rangle$. Cette méthode proposée par [?] donne une suite d'entiers avec des propriétés statistiques très mauvaises [?] et a été vite abandonnée.
3. mélange aléatoire des méthodes connues. Il s'agit d'une méthode qui tente de combiner de manière aléatoire les méthodes précédentes pour fournir une « meilleure » suite aléatoire. Il a été démontré qu'en absence d'une théorie, une telle méthode peut être très dangereuse (cf. Algorithme K dans Vol. 2, page 4 de [?]). Dans la quête du « toujours plus », on s'est aperçu que l'arithmétique ne manque pas ...d'humour.

Si les premières méthodes algorithmiques n'ont pas résolu le problème de génération de nombres au hasard, elles ont eu le mérite d'introduire une nouvelle problématique. Il est apparu en effet que l'algorithmique ne fournit pas des suites aléatoires mais des suites déterministes qui se comportent comme des suites aléatoires dans le sens que leurs propriétés statistiques pertinentes sont similaires à celles des suites aléatoires [?]. On parle alors de nombres *pseudo-aléatoires*.

Les « temps modernes » (après 1950) : L'essor des ordinateurs digitaux a imposé des algorithmes simples et rapides pour la génération des nombres pseudo-aléatoires. Les premiers résultats de la théorie ergodique de systèmes dynamiques sont aussi disponibles ce qui incite à utiliser des transformations ergodiques du cercle, ou plus précisement leurs contre-partie discrète, pour engendrer des suites pseudo-aléatoires. Les méthodes les plus utilisées de nos jours sont

1. la méthode de récurrence congruentielle affine de rang 1[?] qui consiste à fixer des constantes positives a, c et m appelées respectivement *multiplicateur*, *accroissement* et *module* ainsi que la valeur initiale x_0, appelée *racine du générateur* et à considérer la suite des entiers x_i, $i = 1, 2, \cdots$ donnés par la récurrence

$$x_i = ax_{i-1} + c \mod m.$$

2. la méthode de récurrence congruentielle affine de rang r [?, ?] consiste à fixer le module m, les r paramètres $a_1, \ldots, a_r$ et le paramètre c ainsi que les r valeurs initiales $x_0, \ldots, x_{r-1}$ du générateur et à considérer la suite, pour $i = r, r+1, \ldots$,

$$x_i = a_1 x_{i-1} + \ldots + a_r x_{i-r} + c \mod m.$$

Le cas particulier $r = 1$ nous ramène au cas précédent. Le cas particulier $r = 2$, $a_1 = a_2 = 1$ et $c = 0$, connu sous le nom de générateur de Fibonacci, se comporte comme la suite congruentielle affine avec $a = (1 + \sqrt{5})/2$ et a des propriétés statistiques très mauvaises. Celles-là s'améliorent au fur et à mesure que r augmente.

[2] Personnage mythologique qui comparait la hauteur de ses victimes avec la longueur de son lit ; si elles étaient plus grandes, il les amputait, si elles étaient plus petites, il les étirait afin d'égaliser leur hauteur avec la longueur de son lit. En fin probabiliste, il ne se préoccupait pas des personnes dont la hauteur coïncidait exactement avec la longueur de son lit, persuadé qu'elles ne forment qu'un échantillon « négligeable ».

3. la méthode de décalage du registre dérive de la méthode précédente (avec $m = 2$) dans le sens qu'une méthode de récurrence congruentielle affine de rang r est appliquée non pas aux nombres eux mêmes mais sur les digits de leur représentation binaire.

4. la méthode de Fibonnaci lacunaire où l'on fixe deux lacunes r et s avec $r > s > 0$ et r racines $x_0, \ldots, x_{r-1}$ et on construit la suite

$$x_i = x_{i-r} \circ x_{i-s} \quad \text{pour} \quad i \geq r,$$

avec $\circ$ représentant une opération binaire (par exemple l'addition modulo m).

1.3 Générateurs uniformes sur $[0, 1]$

On a vu dans la section précédente que les méthodes actuellement utilisées pour engendrer des nombres aléatoires sont toutes algorithmiques. Par conséquent, les suites obtenues sont purement déterministes. Elles doivent se comporter de manière à vérifier soit tous les théorèmes des probabilités soit certains d'entre eux ; on parle alors respectivement de suites pseudo-aléatoires ou quasi-aléatoires. Toutes les simulations se faisant sur ordinateur, les nombres que l'on peut représenter exactement ne sont qu'un sous-ensemble *fini* des rationnels. En multipliant ces rationnels par le plus grand dénominateur qui appraît dans la liste de ses réduits, on obtient un ensemble fini d'entiers. Sans perte de généralité, on peut donc se limiter au cas des nombres *entiers-machine* non-signés représentables sur une architecture à B bits ; on note $\mathsf{EMR}(B)$ cet ensemble. Par exemple, pour l'architecture la plus courante à 32 bits, on a $\mathsf{EMR}(32) = \{0, 1, \ldots, 4294967295\}$ avec $\operatorname{card} \mathsf{EMR}(32) = 2^{32} = 4294967296$.

Plus généralement, un générateur algorithmique de nombres au hasard sur $[0, 1]$ est un système dynamique discret (X, T), où X est un ensemble discret et fini de symboles arbitraires et T une application $T : X \to X$, couplé à une application de normalisation $U : X \to [0, 1]$ qui à chaque symbole $x \in X$ associe un nombre dans $[0, 1]$. Il faut noter cependant que $U(X)$ est un ensemble discret et fini et que l'application U est la partie triviale du générateur ; tout l'aspect aléatoire est codé dans l'application T. Partant d'un symbole initial $x_0 \in X$, l'application répétée de T produit une suite $(x_n)_{n\in\mathbb{N}}$ avec $x_n = T(x_{n-1})$, appelée *orbite émanant de* x_0. Les valeurs $U_n = U(x_n) \in [0, 1]$ vont jouer le rôle de variables aléatoires uniformément distribuées sur $[0, 1]$. Or l'ensemble X étant fini, toute orbite sera inéluctablement périodique et par conséquent les valeurs de la suite (U_n) vont se répéter à partir d'un certain indice appelé *période du générateur*.

On récapitule ci-dessous les qualités requises par un bon générateur algorithmique de suites pseudo-aléatoires telles qu'elles sont définies par Brent [?] :

- *Uniformité.* La suite doit passer avec succès les épreuves d'uniformité de distribution (cf. chapitre ??). La plupart de générateurs utilisés des nos jours, passent ces tests.
- *Indépendance.* L'orbite complète et des sous-orbites particulières doivent être indépendantes. Les générateurs les plus utilisés de nos jours remplissent cette condition pour toutes les sous-orbites du type U_{nd} pour des valeurs raisonnables de d ($d \leq 6$).
- *Longue période.* Les programmes de simulation qui tournent sur des superordinateurs ont besoin de 10^{10} nombres aléatoires par seconde. Sachant que les programmes typiques ont des temps d'exécution allant de quelques heures à quelques mois, ils ont besoin de 10^{14} à 10^{18} nombres au hasard. Ceci impose une borne inférieure à la période des générateurs.

- *Reproducitbilité.* Pour pouvoir vérifier les programmes lors de leur développement, on doit être capable de reproduire exactement la suite de variables aléatoires générées.
- *Portabilité.* Encore pour des raisons de vérification, il est parfois nécessaire de faire exécuter le programme sur des machines différentes, éventuellement avec des longueurs de mots différentes. Si, par exemple, on transporte un programme développé sur une machine avec architecture de 32 bits sur une machine avec une architecture de 64 bits, il faut que les orbites émanant de la même racine soient identiques dans les deux cas.
- *Sous-suites disjointes.* Si une simulation est effectuée sur une machine multi-processeur ou si le calcul est distribué à travers un réseau, il faut que les sous-suites utilisées par chaque sous-tâche du programme soient indépendantes.
- *Efficacité.* Étant donné que l'appel à la routine du générateur s'effectue un très grand nombre de fois, il faut que son programme soit le plus simple possible, avec un minimum d'opérations nécessaires.

La recherche de nouveaux générateurs a pour but de satisfaire le plus grand nombre de critères de qualité mentionnés ci-dessus. Il s'agit d'un domaine de recherche où des compétences en arithmétique et en théorie probabiliste des nombres sont nécessaires et il est en plein essor actuellement pour répondre aux exigences, toujours plus grandes, de précision dans les prédictions issues de simulations.

1.3.1 Récurrences congruentielles de rang 1

Ces générateurs correspondent à des systèmes dynamiques (X,T) avec $X=\{0,1,\ldots,m-1\}$ pour une entier m fixé et $T: X \to X$ une application de la forme $Tx = ax + c \bmod m$. La fonction de normalisation dépend des paramètres du générateur ; elle s'écrit généralement sous la forme $U(x) = x/N(a,c,m)$ où N est une fonction des paramètres.

Générateur $x_i = ax_{i-1} + c \mod m$

Ce générateur est l'un des plus étudiés dans la littérature à cause de la simplicité de sa programmation. Le plus grand entier qui peut être généré est m et de manière évidente la période ne peut pas excéder m. Cette borne supérieure de la période est atteinte pour certaines valeurs des paramètres (par exemple $a=1$; est-ce que ce choix particulier est raisonnable pour générer une suite pseudo-aléatoire ?). Il y cependant des jeux des paramètres pour lesquels la période est strictement inférieure à m (par exemple $a=c=7$ et $m=10$ avec racine $x_0=7$). Ce qui précède nous amène à prendre comme module un entier assez grand très souvent de l'ordre de l'entier-machine le plus grand) et à chercher des conditions sur les paramètres qui nous permettent de maximiser la période. Le théorème suivant donne une solution complète au problème de maximisation de la période.

Théorème 1.3.1 *Le générateur* $x_i = ax_{i-1} + c \mod m$ *a une période égale à* m *si, et seulement si,*

1. c *et* m *sont premiers entre eux,*
2. *pour tout facteur premier* p *de* m, $a-1$ *est un multiple de* p *et*
3. *si* m *est un multiple de 4 alors* $a-1$ *est un multiple de 4.*

La démonstration de ce théorème se trouve, par exemple, dans [?], Vol. 2, pp. 15–17.

Le premier exemple connu [?] de générateur de ce type utilisait les paramètres $m = 2^{35}$, $a = 2^7$ et $c = 1$ et donc ne vérifiait pas les hypothèses du théorème de maximisation de la période. Ultérieurement, la valeur $a = 2^7 + 1$ a été proposée qui vérifie les hypothèses et donne des bonnes propriétés statistiques [?, ?, ?].

Générateur $x_i = ax_{i-1} \bmod m$

Il s'agit du générateur le plus utilisé de nos jours. On sait que sa période est strictement inférieure à m et les théorèmes suivants ([?], pp. 20–22) nous donnent les conditions sous lesquelles on obtient la maximisation de la période.

Théorème 1.3.2 *Le générateur* $x_i = ax_{i-1} \bmod m$ *avec* $m = 2^\beta$ *pour un entier* $\beta \geq 4$ *a une période (maximale)* $m/4$ *si, et seulement si,*

1. x_0 *et* m *sont premiers entre eux et*
2. $a \bmod 8 \in \{3, 5\}$.

En outre, si $a = 5 \bmod 8$ *et* $b = x_0 \bmod 4$ *alors* $(x_i - b)/m = y_i$ *est la suite obtenue par le générateur* $y_i = ay_{i-1} + b(a-1)/4 \bmod (m/4)$.

La dernière partie de ce théorème est due à [?] et répond à une vieille querelle qui opposait les partisans des générateurs avec $c \neq 0$ et de ceux — supposés moins performants — avec $c = 0$.

Théorème 1.3.3 *Si* m *est premier, la période du générateur* $x_i = ax_{i-1} \bmod m$ *est un diviseur de* $m - 1$. *Elle est égale à* $m - 1$ *si, et seulement si,* a *est une racine primitive (* i.e. $a \neq 0$ *et* $a^{(m-1)/p} \neq 1 \bmod m$ *pour tout facteur premier* p *de* $m - 1$*).*

Il peut s'avérer difficile de trouver les racines primitives. Mais si une telle racine est trouvée, toutes les autres sont obtenues en vertu du

Théorème 1.3.4 *Si* a *est une racine primitive de* m*, alors ceci est vrai pour* $a^k \bmod m$ *pourvu que* k *et* $m - 1$ *sont premiers entre eux.*

On voit que la factorisation en nombres premiers de $m - 1$ joue un rôle important. Le tableau ?? donne les facteurs premiers de $2^e - 1$ pour quelques valeurs choisies de e.

Les valeurs des paramètres recommandées par [?] sont $m = 2^{31} - 1$ et $a = 7^5 = 16807$. On vérifie à l'aide du tableau précédent que m est premier et que 7 est une racine primitive de $m - 1 = 2 \times (2^{30} - 1)$. Ce générateur est utilisé par plusieurs bibliothèques des programmes installées sur des ordinateurs divers puisqu'il possède de bonnes propriétés statistiques. Son écriture algorithmique est

e	Factorisation première de 2^e-1
15	$7\times 31\times 151$
16	$3\times 5\times 17\times 257$
30	$3^2\times 7\times 11\times 31\times 151\times 331$
31	2147483647
32	$3\times 5\times 17\times 257\times 65537$
33	$7\times 23\times 89\times 599479$
60	$3^2\times 5^2\times 7\times 11\times 13\times 31\times 41\times 61\times 151\times 331\times 1321$
63	$7^2\times 73\times 127\times 337\times 92737\times 649657$
64	$3\times 5\times 17\times 257\times 65537\times 6700417$

TAB. 1.1 – Décomposition de 2^e-1 en facteurs premiers pour quelques valeurs choisies de e présentant un intérêt particulier pour la simulation numérique. On observe que pour les ordinateurs à 32 bits, on dispose du nombre premier de Marsanne $2^{31}-1$. Il n'y a pas de nombre premier analogue pour les architectures à 64 bits.

Algorithme 1.3.5 GénérateurUnifStandard[0,1]

Requiert: Racine $X\in\{1,2,\ldots,2^{31}-1\}$.

Retourne: $U\in[0,1]$ selon loi uniforme sur $[0,1]$ par méthode congruentielle et nouvelle valeur de X.

$X\leftarrow 16807\times X \mathrm{mod}(2^{31}-1)$

$U\leftarrow \frac{X}{2^{31}-1}$

La programmation de cet algorithme présente certaines difficultés sur les ordinateurs actuels (à architecture de 32 bits). En effet, les entiers X qui apparaissent peuvent être aussi grands que $16807\times 2147483646\simeq 1,03\times 2^{45}$ qui ne sont pas représentables par des entiers-machine de 32 bits. Quelques astuces sont donc nécessaires pour programmer correctement cet algorithme ; deux de ces astuces sont proposées dans le paragraphe d'exercices. Un bon test de l'implémentation informatique de cet algorithme consiste à l'initialiser avec $X_0=1$; la 10000e itérée doit alors être $X_{10000}=1043618065$.

1.3.2 Décalage du registre

Il s'agit d'une récurrence congruentielle affine de rang r qui agit sur les bits. Pour un r fixé, ce générateur est décrit par le système dynamique (X,T) avec $X=\{0,1\}^r$ et : $X\to X$ défini pour tout $x=(b_1,b_2,\ldots,b_r)\in X$ par $Tx=(b_2,\ldots,b_r,b)$ où $b=(a_1b_a+\ldots a_rb_r)\mathrm{mod}2$ et $a_1,\ldots,a_r$ sont des constantes binaires. Partant d'une racine $x_0=(b_1,\ldots,b_r)\in X$ (r bits initiaux) on obtient une suite $(x_n)_{n\in\mathbb{N}}$ (ou de manière équivalente une suite $(b_n)_{n\in\mathbb{N}}$) qui est l'orbite émanant de x_0. L'opération de normalisation consiste à isoler des tronçons de taille k sur l'orbite des bits et à considérer les l premiers bits de chaque tronçon ($l\leq k$) comme la représentation binaire d'un nombre réel dans $[0,1]$. On construit alors la suite $(U_n)_{n\in\mathbb{N}}$ avec

$$U_n=\langle\, 0,b_{nk+1}\ldots b_{nk+l}\,\rangle_2,\quad \text{pour}\quad n=0,1,2,\ldots.$$

On vérifie aisément que l'addition binaire modulo 2 a la même table de vérité que l'opération logique « ou exclusif » notée $\oplus$. Ainsi, supposons que seuls les paramètres $a_{j_1} = a_{j_2} = \cdots = a_{j_m} = 1$, avec $j_1 < j_2 < \cdots < j_m$, les $r - m$ paramètres restants étant nuls. La récurrence s'écrit alors

$$b_i = b_{i-j_1} \oplus [b_{i-j_2} \oplus [b_{i-j_3} \oplus [\cdots \oplus b_{i-j_m}] \cdots] .$$

Cette écriture permet la programmation — et même le câblage — du générateur par décalage du registre avec rétro-action comme le montre l'exemple suivant.

Exemple 1.3.6 [Programmation du générateur $b_i = (b_{i-2} + b_{i-4})\text{mod}2$] Ce générateur est associé au diagramme logique présenté à la figure suivante :

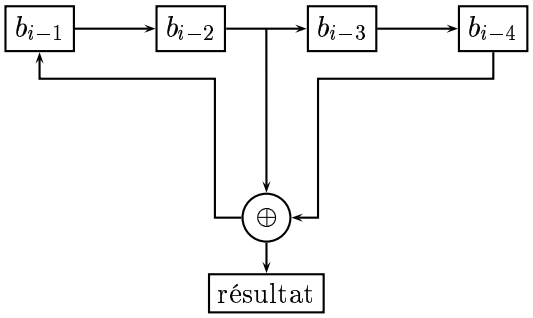

FIG. 1.1 – Schéma logique du générateur $b_i = (b_{i-2} + b_{i-4})\text{mod}2$

Il est évident que les valeurs successives prises par les 4 bits qui servent à programmer ce générateur dépendent des valeurs avec lesquelles ces 4 bits sont initialisés (racine du générateur). Sur 4 bits, on peut coder les entiers de 0 à 15. Si le générateur est initialisé avec des zéros il générera des zéros. Si les bits d'initialisation ne sont pas 0000, le générateur visitera-t-il toutes les autres valeurs de 0001 à 1111 ? En d'autres termes, la période du générateur est-elle $2^4 - 1$? On peut vérifier que le générateur de cet exemple n'a pas une période maximale. Dans le cas général, l'étude de maximisation de la période répose sur une étude du polynôme $h(x)$ associé au générateur.

Définition 1.3.7 À la récurrence $[b_i = b_{i-j_1} \oplus [b_{i-j_2} \oplus [b_{i-j_3} \oplus [\cdots \oplus b_{i-j_m}] \cdots]$ pour un générateur de rang r, on associe le polynôme $h(x)$ de degré r,

$$h(x) = x^r + x^{r-j_1} + \ldots + x^{r-j_m}.$$

Ce polynôme est appelé *primitif* si le générateur correspondant a une période de $2^r - 1$.

Afin d'optimiser le temps de calcul, les polynômes habituellement utilisés sont de la forme $h(x) = x^r + x^q + 1$, avec $q < r$. Ces polynômes sont étudiés et le tableau suivant donne toutes les paires (r, q) avec $r \leq 33$ qui les rendent primitifs (voir par exemple [?, ?, ?]).

Des polynômes (des couples (r, q)) de degré plus élevé ont été proposés dans la littérature comme (98,27) [?], (521,32) [?], (607,273) [?]. Le générateur associé à ce dernier polynôme a une période de $2^{607} - 1 \simeq 10^{182}$.

r	q	r	q
2	1	18	7,11
3	1,2	20	3,17
4	1,3	21	2,19
5	2,3	22	1,21
6	1,5	23	5,9,14,18
7	1,3,4,6	25	3,7,18,22
9	4,5	28	3,9,13,15,19,25
10	3,7	29	2,27
11	2,9	31	3,6,7,13,18,24,25,28
15	1,4,7,8,11,14	33	11,25
17	3,5,6,11,12,14		

TAB. 1.2 – Toutes les paires « primitives » (r, q) avec $r \leq 33$.

La période du générateur n'est évidemment pas la seule caractéristique déterminant sa qualité. Ses propriétés statistiques sont aussi, sinon plus, importantes pour sa validation et celles-ci sont essentiellement déterminées par la constante de décimation k. Cette constante est dite *propre* si $\text{pgcd}(k, 2^r - 1) = 1$. Ainsi, dans [?] les valeurs $r = 33$, $q = 13$ et $k = l = 32$ sont proposées.

Ce générateur à décalage du registre a un intérêt particulier pour la construction de suites de nombres au hasard avec des périodes très longues.

1.4 Générateurs des lois autres que l'uniforme

On a vu au paragraphe précédent comment peut-on générer des nombres pseudo-aléatoires selon la loi uniforme sur $[0, 1]$. Or, dans plusieurs situations des nombres distribués selon d'autres lois sont nécessaires. Dans ce paragraphe on va présenter certaines méthodes qui permettent d'échantillonner selon les lois usuelles.

1.4.1 Méthodes générales

Inversion

La loi d'une variable aléatoire X est déterminée par sa fonction de répartition $F_X(x) = \mathbb{P}(X \leq x)$. F_X est une fonction croissante qui varie entre 0 et 1. Supposons qu'elle soit inversible et que U est une variable aléatoire uniforme sur $[0, 1]$. Alors $X = F_X^{-1}(U)$. En effet,

$$\mathbb{P}(X \leq x) = \mathbb{P}(F_X^{-1}(U) \leq x) = \mathbb{P}(U \leq F_X(x)) = F_X(x).$$

Ceci nous amène à utiliser l'algortihme suivant :

Algorithme 1.4.1 Inversion

Requiert: Fonction de répartition inversible F et LoiUnif[0,1]
Retourne: X selon loi de répartition F par méthode d'Inversion
  Générer U selon LoiUnif[0,1]
  $X \leftarrow F^{-1}(U)$

Exercice 1.4.2 Soient X_1, X_2 deux variables aléatoires indépendantes et de même loi dont la fonction de répartition, F, est inversible. Simuler la variable $Y = \max(X_1, X_2)$.

Solution : La fonction de répartition de la variable aléatoire Y est donnée par

$$G(y) = \mathbb{P}(Y \leq y) = \mathbb{P}(\max(X_1, X_2) \leq y) = \mathbb{P}(\{X_1 \leq y\} \cap \{X_2 \leq y\}) = F(y)^2.$$

Donc,

$$G^{-1}(a) = \{y : G(y) = a\} = \{y : F(y)^2 = a\} = \{y : F(y) = \sqrt{a}\} = F^{-1}(\sqrt{a}).$$

Alors, on obtient $Y = F^{-1}(\sqrt{U})$. □

Si la fonction de répartition n'est pas inversible (*i.e.* F n'est pas *strictement* croissante) on peut généraliser la méthode précédente en définissant

$$F^{-1}(y) = \inf\{x : F(x) \geq y\}.$$

Composition

Soit une loi avec une fonction de répartition F exprimable comme

$$F = \sum_{i=1}^{r} p_i F_i$$

avec $\sum p_i = 1$ et $p_i \geq 0$. Supposons, en outre, que les variables aléatoires distribuées selon les lois F_i soient facilement simulables. Afin d'échantillonner selon F, on utilise l'algorithme suivant :

Algorithme 1.4.3 Composition

Requiert: Vecteur probabilité $\mathbf{p} = (p_1, \ldots, p_r)$, LoiDiscrète($\mathbf{p}$) et LoiRépartition($F_i$), $i = 1, \ldots, r$
Retourne: Z selon loi de répartition $F = \sum_{i=1}^{r} p_i F_i$ par méthode de Composition
  Générer $J \in \{1, \ldots, r\}$ selon LoiDiscrète($\mathbf{p}$)
  Générer X selon LoiRépartition(F_J)
  $Z \leftarrow X$

Démonstration : En effet,

$$\mathbb{P}(X \leq x) = \sum_{k=1}^{r} \mathbb{P}(X \leq x | J = k)\mathbb{P}(J = k) = \sum_{k=1}^{r} F_k(x) p_k = F(x).$$

□

Remarque : Cette méthode se généralise trivialement au cas où $F = \sum_{i=1}^{\infty} p_i F_i$ avec $p_i \geq 0$ et $\sum_{i=1}^{\infty} p_i = 1$. Elle se généralise aussi au cas où F s'écrit comme une intégrale selon une famille des densités des probabilités conditionnelles $g(x|y)$ comme

$$F_X(x) = \int g(x|y) F_Y(dy) dx.$$

Exercice 1.4.4 Générer une variable aléatoire de loi $F(x) = \sum_{i=1}^{\infty} p_i x^i$ pour $0 \leq x \leq 1$, $p_i \geq 0$ et $\sum_{i=1}^{\infty} p_i = 1$.

Solution : On tire deux variables aléatoires U et V, indépendantes, selon la loi uniforme sur $[0,1]$ et on détermine la variable aléatoire J telle que $\sum_{i=1}^{J-1} p_i \leq U \leq \sum_{i=1}^{J} p_i$. Alors, $X = V^{1/J}$ est distribuée selon F. En effet,

$$\mathbb{P}(X \leq x) = \sum_{k=1}^{\infty} \mathbb{P}(J=k)\mathbb{P}(V^{1/J} \leq x | J=k) = \sum_{k=1}^{\infty} p_k \mathbb{P}(V \leq x^k) = \sum_{k=1}^{\infty} p_k x^k.$$

□

Rejet

Il s'agit de générer une variable aléatoire suivant une loi dont la densité est une fonction f qui peut être représentée sous la forme

$$f(x) = Ch(x)g(x)$$

où $C \geq 1$ et h est la densité d'une loi facilement simulable et $0 < g(x) \leq 1$ pour tout x.

Théorème 1.4.5 *Soit X une variable aléatoire de loi ayant comme densité $f_X(x) = Cg(x)h(x)$, où $C \geq 1$ et $0 < g(x) \leq 1$ pour tout x ; la fonction h est la densité d'une loi de probabilité. Soient U une variable aléatoire distribuée selon la loi uniforme sur $[0,1]$ et Y une variable aléatoire distribuée selon la loi de densité h. Alors,*

$$\mathbb{P}(Y \in dx | U \leq g(Y)) = f_X(x)dx.$$

Démonstration : Par la formule de Bayes,

$$\mathbb{P}(Y \in dx | U \leq g(Y)) = \frac{\mathbb{P}(U \leq g(Y) | Y \in dx)\mathbb{P}(Y \in dx)}{\mathbb{P}(U \leq g(Y))}.$$

Or, $\mathbb{P}(U \leq g(Y) | Y \in dx)\mathbb{P}(Y \in dx) = g(x)h(x)dx$ et

$$\mathbb{P}(U \leq g(Y)) = \int \mathbb{P}(U \leq g(Y) | Y \in dx)\mathbb{P}(Y \in dx) = \int g(x)h(x)dx = \frac{1}{C}.$$

□

Ce théorème nous incite à utiliser l'algorithme suivant :

Algorithme 1.4.6 Rejet

Requiert: Fonction $g > 0$, LoiUnif[0,1] et LoiDensité(h)
Retourne: Z selon loi de densité $Cg \cdot h$, où C normalisation, par méthode du Rejet
  répéter
    Générer U selon LoiUnif[0,1]
    Générer Y selon LoiDensité(h)
  jusqu'à $U \leq g(Y)$
  $Z \leftarrow Y$

Remarque : Le nombre d'essais, N, nécessaires pour accepter un nombre Y est une variable aléatoire de loi

$$\mathbb{P}(N = k) = \mathbb{P}(U \leq g(Y))[\mathbb{P}(U > g(Y))]^{k-1} = \frac{1}{C}(1 - \frac{1}{C})^{k-1}$$

d'espérance $\mathbb{E}N = C$. Donc, l'efficacité de l'algorithme est d'autant plus grande que la constante C est petite (proche de 1). Le nombre $1/C$ est appelé *taux d'acceptation.*

Exemple 1.4.7 Pour générer une variable aléatoire selon la loi de densité $f(x) = 3x^2$ si $0 \leq x \leq 1$ et 0 sinon, on peut choisir $C = 3$, $h(x) = 1$ et $g(x) = x^2$.

1.4.2 Génération de variables aléatoires selon les lois usuelles

Loi normale $\mathcal{N}(0,1)$

Plusieurs algorithmes permettent d'échantillonner la loi normale. Le plus simple est une variante de la méthode du rejet connu sous le nom d'*algorithme polaire :*

Algorithme 1.4.8 LoiNormale(0,1)

Requiert: Loi LoiUnif[0,1]
Retourne: Z_1 et Z_2 indépendantes selon LoiNormale(0,1)
  répéter
    Générer U_1 et U_2 indépendantes selon LoiUnif[0,1]
    $V_1 \leftarrow 2U_1 - 1$
    $V_2 \leftarrow 2U_2 - 1$
    $S \leftarrow V_1^2 + V_2^2$
  jusqu'à $S \leq 1$
  $Z_1 \leftarrow V_1\sqrt{-2\ln S/S}$
  $Z_2 \leftarrow V_2\sqrt{-2\ln S/S}$

Démonstration : (V_1, V_2) est un point uniformément distribué dans le carré $[-1,1]^2$. Si $S \leq 1$, ce point est aussi à l'intérieur du disque unité. En passant en coordonnées polaires, $V_1 = R\cos\Theta$

et $V_2 = R\sin\Theta$, la variable S s'exprime comme $S = R^2$ et dans le disque $R \leq 1$, les variables aléatoires S et Θ sont indépendantes avec Θ distribuée selon la loi uniforme sur $[0, 2\pi]$ et S distribuée selon la loi uniforme sur $[0, 1]$. En notant $Y = \sqrt{-2\ln S}$, on a

$$F_Y(r) = \mathbb{P}(Y \leq r) = \mathbb{P}(-2\ln S \leq r^2) = \mathbb{P}(S \geq \exp(-r^2/2)) = 1 - \exp(-r^2/2).$$

Donc, $\mathbb{P}(Y \in dr) = r\exp(-r^2/2)dr$ et $X_1 = (Y/R)R\cos\Theta = Y\cos\Theta$ et de même $X_2 = Y\sin\Theta$. On trouve alors pour la probabilité conjointe :

$$\begin{aligned}\mathbb{P}(X_1 \leq x_1, X_2 \leq x_2) &= \int_{\{(r,\theta): r\cos\theta \leq x_1, r\sin\theta \leq x_2\}} \tfrac{1}{2\pi}\exp(-r^2/2) r dr d\theta \\ &= \tfrac{1}{2\pi}\int_{\{(x,y): x\leq x_1, y\leq x_2\}} \exp(-x^2/2 - y^2/2)dx\,dy \\ &= \left(\tfrac{1}{\sqrt{2\pi}}\int_{-\infty}^{x_1}\exp(-x^2/2)dx\right) \times \left(\tfrac{1}{\sqrt{2\pi}}\int_{-\infty}^{x_2}\exp(-y^2/2)dy\right).\end{aligned}$$

Ceci montre que X_1 et X_2 sont indépendantes et distribuées selon $\mathcal{N}(0,1)$. □

Loi exponentielle de paramètre 1

Cette loi, de densité $f(x) = \exp(-x)$, est d'une grande utilité pour tous les problèmes relatifs aux files d'attente. La simulation est basée sur la méthode d'inversion et l'algorithme est le suivant.

Algorithme 1.4.9 LoiExpon(1)
Requiert: Loi LoiUnif[0,1]
Retourne: Z selon LoiExpon(1)
  Générer U selon LoiUnif[0,1]
  $Z \leftarrow -\ln(U)$

Démonstration : La fonction de répartition de la loi exponentielle est donnée par

$$F(x) = \mathbb{P}(X \leq x) = \int_0^x \exp(-t)dt = 1 - \exp(-x).$$

Donc $F^{-1}(y) = -\ln(1-y)$ pour $y \in [0,1]$. Or, si $1-U$ est uniformément distribuée sur $[0,1]$, U l'est aussi. □

Loi du χ^2 à ν degrés de liberté

Cette loi est définie [?] comme la loi suivie par la variable aléatoire $X = Y_1^2 + \cdots + Y_\nu^2$ où les Y_i sont des variables aléatoires indépendantes distribuées selon $\mathcal{N}(0,1)$.

Lemme 1.4.10 *La fonction de répartition de la loi du χ^2 à ν degrés de liberté est donnée pour $x \geq 0$ par*

$$F_\nu(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}\int_0^x t^{\nu/2-1}\exp(-t/2)dt.$$

Démonstration : Si $X \sim \mathcal{N}(0,1)$ alors $\mathbb{P}(X^2 \in dx) = \frac{\exp(-x/2)}{\sqrt{2\pi x}}dx$. Par conséquent, si $X_1, X_2 \sim \mathcal{N}(0,1)$ et X_1, X_2 sont indépendants,

$$\mathbb{P}(X_1^2 + X_2^2 \in dx) = \frac{\exp(-x/2)}{2}dx \tag{1.1}$$

Finalement, on montre que si $Y \sim F_{\nu_1}$ et $Z \sim F_{\nu_2}$ avec Y et Z indépendantes, alors $Y + Z \sim F_{\nu_1+\nu_2}$. □

La définition de la loi du χ^2 et la relation (**??**) sont mises à profit pour introduire un algorithme très efficace d'échantillonnage selon χ^2 :

Algorithme 1.4.11 LoiChi2(ν)
Requiert: LoiUnif[0,1] et LoiNormale(0,1)
Retourne: Z selon LoiChi2(ν)
  $k = [\nu/2]$
  Générer $U_1, \ldots, U_k$ indépendantes selon LoiUnif[0,1]
  $X \leftarrow -\ln(U_1 \cdots U_k)$
  si $\nu = 2k$ **alors**
    $Z \leftarrow X$
  sinon
    Générer Y selon LoiNormale(0,1)
    $Z \leftarrow X + Y^2$
  fin si

Remarque : L'attention est attirée que pour générer la variable aléatoire $(Y_1 + \cdots + Y_k)$ de l'algorithme précédent, on génère k variables aléatoires $U_1, \ldots, U_k$ distribuées selon la loi uniforme sur $[0,1]$ et on pose $(Y_1 + \cdots + Y_k) = -\ln(U_1 \cdots U_k)$ et non pas $-(\ln U_1 + \cdots + \ln U_k)$. Ceci a l'avantage de faire appel une seule fois à la fonction logarithme.

1.5 Génération des nombres pseudo-aléatoires corrélés

Il est souvent nécessaire d'engendrer des variables aléatoires corrélées entre elles. Le cas des processus sera traité *in extenso* dans le chapitre consacré à l'intégration stochastique. Ici on se limite au cas où il faut générer une suite de paires des variables aléatoires corrélées, avec un coefficient de corrélation fixé à l'avance. L'algorithme de simulation découle immédiatement du lemme suivant.

Lemme 1.5.1 *Soient X et Y deux variables aléatoires indépendantes de même loi. En supposant que $E(X^2)$ existe, la variable aléatoire $Z = \alpha X + (1-\alpha)Y$ avec $-1 \leq \alpha \leq 1$ a un coefficient de corrélation avec X donné par*

$$\rho(X, Z) = \frac{\alpha}{\sqrt{\alpha^2 + (1-\alpha)^2}}.$$

1.6 Suites déterministes, systèmes dynamiques et nombres au hasard

On a ouvert ce chapitre avec quelques réflexions sur la nature logique de la notion d'une suite aléatoire. Après avoir vu les différentes méthodes de génération de telles suites on reste troublé. Il serait donc sain, avant de clore ce chapitre et d'entamer le suivant sur les épreuves statistiques des nombres au hasard, de se poser de nouveau quelques questions fondamentales sur la nature de l'« aléatoire ».

La notion de l'aléatoire inclut intuitivement une notion d'imprévisibilité, d'impossibilité de prédire sa valeur avant d'effectuer l'expérience de mesure. Or on produit de nombres pseudo-aléatoires en se servant de procédures purement déterministes. On connaît plusieurs phénomènes physiques qui quoique régis par des équations déterministes présentent de telles imprédictibilités ; le cas le plus parlant étant celui de la météorologie. L'atmosphère est un système décrit par les équations des fluides qui sont des équations aux dérivées partielles non-linéaires mais il se comporte comme un système que l'on caractérise de *complexe* ou *chaotique*. D'autres systèmes, connus sous le nom de *systèmes dynamiques* présentent de caractéristiques analogues, à savoir

- ils dépendent de manière sensitive aux conditions initiales, leurs trajectoires émanant de deux conditions initiales très « proches » se mettent à diverger de manière exponentielle avec le temps (divergence régie par les *exposants de Lyapunov*).
- ils produisent de l'information en possédant une *entropie* ou *complexité de Kolmogorov* non-nulles.
- les trajectoires se concentrent asymptotiquement sur des sous-ensembles de l'espace de phases, les *attracteurs*, ayant une dimension de Hausdorff non-triviale.

Ces notions vont être précisées par la suite en suivant [**?**] et [**?**].

1.6.1 Systèmes dynamiques, transformations et théorèmes ergodiques

Définition 1.6.1 Un *système dynamique* est la donnée d'un espace mesuré $(X, \mathcal{A}, \mu)$ et d'une transformation $T : X \to X$ qui est mesurable ($\forall A \in \mathcal{A}, T^{-1}(A) \in \mathcal{A}$) et non-singulière ($\forall A \in \mathcal{A}$ tels que $\mu(A) = 0$ alors $\mu(S^{-1}(A)) = 0$).

On s'intéresse aux itérées successive de T en partant d'un point $y_0 \in X$, $y_1 = T(y_0)$, $y_2 = T(y_1)$,.... La donnée $\mathbf{y} = (y_0, y_1, y_2, \ldots)$ est appelée *orbite* du système émanant de y_0.

Exemple 1.6.2 Soit $X = [0, 1]$ muni de sa tribu borélienne et d'une mesure μ arbitraire absolument continue par rapport à la mesure de Lebesgue. On définit $T : X \to X$ par $T(y) = 4y(1-y)$. Pour un y_0 générique — c'est-à-dire à l'extérieur d'un ensemble exceptionnel négligeable — l'orbite $\mathbf{y}$ remplit l'intervalle.

Pour mieux étudier les propriétés stochastiques de la trajectoire, on fixe un entier n, on partitionne $[0, 1[= \cup_{i=1}^{n} [i - 1/n, i/n[$ et on génère l'histogramme de fréquence de visite pour

chaque sous-intervalle lors des N premières itérations de la transformation T

$$\nu_i = \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}_{[i-1/n,i/n[}(T^i y_0).$$

On s'attend à la convergence de $\lim_{N\to\infty} \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}_{[i-1/n,i/n[}(T^i y_0)$ vers $\int_{i-1/n}^{i/n} f(u)du$, où f est une certaine fonction densité. Pour déterminer cette fonction densité il s'avère plus simple de considérer non pas une orbite $\mathbf{y}$ mais un ensemble d'orbites émanant des points différents distribués selon une densité. Soit donc une variable aléatoire Y_0 à valeurs dans X telle que $\mathbb{P}(Y_0 \in dy_0) = f_0(y_0)dy_0$ où f_0 est une certaine fonction densité sur X. On examine l'orbite aléatoire $\mathbf{Y} = (Y_0, Y_1, \ldots)$ où $Y_i = T(Y_{i-1})$, pour $i = 1, 2, \ldots$. On a alors

$$\begin{aligned}\mathbb{P}(Y_1 \in [a,y]) &= \int_a^y f_1(u)du \\ &= \int_{T^{-1}([a,y])} f_0(u)du.\end{aligned}$$

De cette équation on conclut

$$f_1(y) = \frac{d}{dy}\int_{T^{-1}([a,y])} f_0(u)du.$$

Définition 1.6.3 Soit $P : L^1(X) \to L^1(X)$ l'opérateur défini par

$$Pf(y) = \frac{d}{dy}\int_{T^{-1}([a,y])} f_0(u)du$$

ou de manière équivalente

$$\int_A Pf(y)\mu(dy) = \int_{T^{-1}(A)} f(x)\mu(dx),$$

pour tout $A \in \mathcal{A}$. Cet opérateur est appelé *opérateur de Perron et Frobenius.*

On s'intéresse aux densités limites de l'itération ; elles sont des points fixes de l'opérateur de Perron et Frobenius. La convergence vers la densité limite siginifie que quelque que soit la densité initiale, la trajectoire charge asymptotiquement l'espace X selon une densité indépendante de la condition initiale. On peut même imaginer des densités initiales arbitrairement concentrées sur des points génériques (« presque de masses de Dirac ») et obtenir des densités asymptotiques diffuses. On a un oubli de la condition initiale pour ces systèmes déterministes qui les rapproche du comportement stochastique des processus markoviens. En effet, on peut définir

Définition 1.6.4 Soit $(X, \mathcal{A}, \mu)$ un espace mesuré et $P : L^1 \to L^1$ un opérateur tel que, pour toute application $f \geq 0$, $f \in L^1$, on ait $Pf \geq 0$ et $\|Pf\| = \|f\|$. Alors l'opérateur P est dit *markovien.*

Il est alors facile de montrer que l'opérateur de Perron et Frobenius est un opérateur de Markov, cas particulier d'une chaîne de Markov dite déterministe **??**. Par ailleurs, toute application

intégrable $f \in L^1$ est en bijection, par le théorème de Radon et Nikodým, avec une mesure μ_f absolument continue par rapport à la mesure μ ($\mu_f(A) = 0$ pour tout $A \in \mathcal{A}$ tel que $\mu(A) = 0$). L'opérateur de Perron et Frobenius agit par conséquent (à gauche) sur les mesures μ_f par $\mu_f P(A) = \int_A \mu(dy) Pf(y)$.

L'espace des fonctions intégrables $L^1(X, \mathcal{A}, \mu)$ étant dual à l'espace de fonctions bornées $L^\infty(X, \mathcal{A}, \mu)$, on peut définir l'action de la transformation T sur ce dernier espace par dualité.

Définition 1.6.5 Soit $(X, \mathcal{A}, \mu)$ un espace mesuré, $T : X \to X$ une application non-singulière et $g \in L^\infty(X, \mathcal{A}, \mu)$. L'opérateur $U : L^\infty \to L^\infty$, défini par $Uf(x) = f(Tx)$, est appelé *opérateur de Koopman* pour la transformation T.

Proposition 1.6.6 *L'opérateur de Koopman est l'adjoint de l'opérateur de Perron et Frobenius*

Démonstration : Notant $\langle \cdot, \cdot \rangle$ le produit scalaire de la dualité, on calcule pour $f \in L^1$, $g = \mathbb{1}_A \in L^\infty$ et $A \in \mathcal{A}$, les produits scalaires $\langle Pf, g \rangle$ et $\langle f, Ug \rangle$. On a

$$\begin{aligned} \langle Pf, g \rangle &= \int_X Pf(x) \mathbb{1}_A(x) \mu(dx) = \int_A Pf(x) \mu(dx). \\ \langle f, Ug \rangle &= \int_X f(x) U\mathbb{1}_A(x) \mu(dx) = \int_X f(x) \mathbb{1}_A(T(x)) \mu(dx) \\ &= \int_{T^{-1}(A)} f(x) \mu(dx) = \int_A Pf(x) \mu(dx). \end{aligned}$$

□

L'étude des propriétés probabilistes d'un système dynamique se fait aisement à travers ses ensembles invariants.

Définition 1.6.7 Soit $(X, \mathcal{A}, \mu, S)$ un système dynamique. L'ensemble $A \subset X$ est *invariant* si $T^{-1}(A) = A$.

On distingue trois niveaux de comportement de plus en plus « aléatoire ».

Définition 1.6.8 Soit $(X, \mathcal{A}, \mu, T)$ un système dynamique. Le système dynamique (ou de manière équivalente la transformation T) est dit *ergodique* si pour tout ensemble A invariant mesurable, on a soit $\mu(A) = 0$ soit $\mu(A^c) = 0$.

La signification de cette définition est quelque peu clarifiée par le résultat suivant.

Théorème 1.6.9 *Soit $(X, \mathcal{A}, \mu, T)$ un système dynamique. La transformation T est ergodique si, et seulement si, toute application mesurable $g : X \to \mathbb{R}$ qui vérifie $g(Tx) = g(x)$, pour μ-presque tout x, est constante μ-presque partout.*

Ce résultat nous dit qu'une fonction constante sur les trajectoires et constante partout. Autrement dit, les trajectoires sont tellement recroquevillées sur l'espace qu'elles le remplissent. Ce résultat est complété par les deux théorèmes suivants

Théorème 1.6.10 (Birkhoff) *Soient $(X, \mathcal{A}, \mu)$ un espace mesuré, $T : X \to X$ une application mesurable et $f : X \to \mathbb{R}$ une application intégrable. Si la mesure μ est invariante, alors il existe une application $f^* : X \to \mathbb{R}$ intégrable telle que*

$$f^*(x) = \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} f(T^k x), \textit{ pour presque tout } x.$$

Théorème 1.6.11 (Théorème ergodique) *Soient $(X, \mathcal{A}, \mu)$ un espace mesuré avec $\mu(X) < \infty$, $S : X \to X$ une application ergodique et μ une mesure invariante (i.e. $\mu(T^{-1}(A) = \mu(A)$ pour tout $A \in \mathcal{A}$). Alors pour toute application intégrable $f : X \to \mathbb{R}$,*

$$\lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} f(T^k y) = \frac{1}{\mu(X)} \int_X f(x)\mu(dx), \textit{ pour presque tout } y.$$

Ce dernier théorème garantit donc que les espérances peuvent être estimées par des moyennes ergodiques.

Définition 1.6.12 Soit $(X, \mathcal{A}, \mu, T)$ un système dynamique avec μ une probabilité. Une transformation T qui préserve la mesure (*i.e.* $\mu(T^{-1}(A)) = \mu(A)$ pour tout $A \in \mathcal{A}$) est dite *mélangeante* si

$$\lim_{n \to \infty} \mu(A \cap T^{-n}(B)) = \mu(A)\mu(B), \text{ pour tout } A, B \in \mathcal{A}.$$

Définition 1.6.13 Soit $(X, \mathcal{A}, \mu, T)$ un système dynamique avec $\mu(X) < \infty$. Si la transformation T laisse la mesure μ invariante, elle est appelée *exacte* si, pour tout $A \in \mathcal{A}$ tel que $\mu(A) > 0$, on a $\lim_{n \to \infty} \mu(T^n(A)) = 1$.

Théorème 1.6.14 *Une transformation exacte est mélangeante. Une transformation mélangeante est ergodique.*

Finalement, le résultat suivant montre une graduation dans les convergences pour les suites ergodiques, mélangeantes et exactes.

Théorème 1.6.15 *Soient $(X, \mathcal{A}, \mu)$ un espace de probabilité, $T : X \to X$ une application qui preserve μ et P l'opérateur de Perron et Frobenius associé. Alors*

1. *T est ergodique si, et seulement si, pour tout $f \in L^1$ et tout $g \in L^\infty$, la suite $\langle P^n f, g \rangle$ converge en moyenne de Cesàro,*

$$\lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \langle P^k f, g \rangle = \langle f, 1 \rangle \langle 1, g \rangle;$$

2. *T est mélangeante si, et seulement si, pour tout $f \in L^1$, la suite $P^n f$ converge faiblement,*

$$\lim_{n\to\infty} \langle P^n f, g \rangle = \langle f, 1\rangle\langle 1, g\rangle, \text{ pour tout } g \in L^\infty;$$

3. *T est exacte si, et seulement si,pour tout $f \in L^1$, la suite $P^n f$ converge fortement,*

$$\lim_{n\to\infty} \|P^n f - \langle f, 1\rangle\| = 0.$$

1.6.2 Exposants de Lyapunov

On étudiera dans ce paragraphe un deuxième aspect de l'imprévisibilité d'un système dynamique, sa sensibilité aux conditions initiales. Soit $(X, \mathcal{A}, \mu, T)$ un système dynamique et supposons l'espace $X = [0,1]$ muni de sa métrique euclidienne. Prenons deux points y_0 et y_0' de X « proches » comme conditions initiales des suites $T^n(y_0)$ et $T^n(y_0')$. On a alors

$$\begin{aligned} T^n(y_0') - T^n(y_0) &= \frac{d}{dy}T^n(y_0)\cdot(y_0' - y_0) + \mathcal{O}(y_0' - y_0)^2 \\ &= \frac{d}{dy}T(y_{n-1})\frac{d}{dy}T(y_{n-2})\ldots\frac{d}{dy}T(y_0)\cdot(y_0' - y_0) + \mathcal{O}(y_0' - y_0)^2. \end{aligned}$$

Si $X \subset \mathbb{R}^m$, on doit remplacer la dérivée $\frac{d}{dy}T(y_0)$ par $D_yT(y_0)$ où D_yT est la matrice jacobienne $(\frac{\partial T_i}{\partial y_j})_{i,j=1,\ldots,m}$.

On a alors le résultat suivant

Théorème 1.6.16 (Oseledets) *La limite*

$$\lambda = \lim_{n\to\infty} \frac{1}{n}\log\|D_yT^n\delta y\|$$

existe pour μ-presque tout y ; elle dépend de la direction δy. Si μ est ergodique, la plus grande valeur de λ (en fonction des directions possibles) est indépendante de y. Cette valeur est alors appelée plus grand exposant de Lyapunov

Intuitevement, ce théorème nous dit que, pour $\|y_0' - y_0\| = \mathcal{O}(1)$, $T^n(y_0') - T^n(y_0) = \mathcal{O}(\exp(\lambda n))$. Donc si $\lambda > 0$, deux trajectoires émanant de points très voisins se mettent à diverger exponentiellement avec le temps n pour certaines directions. C'est l'effet « papillon » de la météorologie : un papillon qui bât les ailes à Tokyo peut déclencher un cyclone tropical en Guadeloupe.

1.6.3 Entropie et quantité d'information

Entropie de Boltzmann ; une espérance qui nous fait vieillir !

La notion d'entropie est une des notions fondamentales de la thermodynamique. Elle est intimement liée à l'irréversibilité temporelle des évolutions macroscopiques. En effet, toutes les

équations qui régissent les phénomènes microscopiques sont réversibles dans le temps. Par contre les évolutions macroscopiques sont irréversibles. Déjà les thermodynamiciens du 19e siècle ont introduit la quantité phénoménologique d'entropie comme la quantité mesurant le « désordre » du système et énoncé le premier principe de la thermodynamique qui stipule que l'entropie d'un système isolé est une fonction croissante du temps. Toute évolution macroscopique s'effectue donc de l'état le plus ordonné vers le plus désordonné. On doit cependant à Boltzmann une définition microscopique de l'entropie comme l'espérance d'une certaine quantité. Pour ne pas avoir à assimiler toute la théorie des gaz, on illustre dans la suite les idées de Boltzmann dans un modèle très simple.

Soit un « dé polyhédrique » à r faces, chacune d'elles ayant une probabilité d'apparition $p_i, i = 1, \ldots, r$. On s'intéresse au nombre des fois que chaque face apparaît lorsqu'on jette n fois le « dé ». On formalise la situation en introduisant l'ensemble fini $A = \{1, \ldots, r\}$ que l'on probabilise à l'aide du vecteur $p = (p_1, \ldots, p_r)$, avec $p_i \geq 0$ et $\sum_{i=1}^r p_i = 1$. L'espace des épreuves est alors $\Omega = A^n$ qui sera muni de la probabilité produit. Soit $(M_i)_{i=1,\ldots,r}$ la famille de variables aléatoires

$$M_i(\omega) = \sum_{k=1}^{n} \mathbb{1}_{\{i\}}(\omega_k), i = 1, \ldots, r$$

qui représentent le nombre des fois où chaque face apparaît. On s'intéresse à la quantité

$$P = \mathbb{P}(M_1 = m_1, \ldots, M_r = m_r) = \frac{n!}{m_1! \ldots m_r!} p_1^{m_1} \ldots p_r^{m_r}.$$

Évidemment, $\sum_{i=1}^r M_i(\omega) = 1$ pour tout ω de sorte que le vecteur aléatoire avec des composantes $\pi_i(\omega) = \frac{M_i(\omega)}{n}$, pour $i = 1, \ldots, r$ soit une probabilité sur A (probabilité empirique). On obtient une expression approchée de $\log P$ à grand n en se servant de la formule de Stirling[3]

$$\log P = n \sum_{i=1}^{r} \pi_i (\log p_i - \log \pi_i).$$

Lemme 1.6.17 *Pour tout $u > 0$ et $v > 0$, l'inégalité de Gibbs est vraie*

$$u - \log u \leq v - u \log v.$$

Démonstration : Soit $\eta(u) = -u \log u$ définie pour $u \geq 0$ avec la convention $0 \log 0 = 0$. La fonction η est concave sur $\mathbb{R}^+$ puisque $\eta'' = -1/u < 0$. Son graphe reste donc au dessous de la tangente à tout point. On a donc

$$\frac{\eta(u) - \eta(v)}{u - v} \leq \eta'(v) = -\log v - 1,$$

d'où découle l'inégalité annoncée. □

Corollaire 1.6.18 *Si $p = (p_1, \ldots, p_r)$ et $q = (q_1, \ldots, q_r)$ sont deux vecteurs de probabilité arbitraires sur l'espace fini $\{1, \ldots, r\}$, alors*

$$\sum_{i=1}^{r} q_i \log p_i - \sum_{i=1}^{r} q_i \log q_i \leq 0.$$

[3]Le fait que l'on utilise la formule de Stirling pour estimer la quantité $\log P$ n'enlève rien à la qualité des résultats. On peut en effet montrer, en se servant de la théorie des grandes déviations, que les résultats obtenus dans le cadre de l'approximation de Striling sont rigoureusement vrais.

Proposition 1.6.19 *Pour le modèle du « dé polyhédrique », les configurations $\omega \in \Omega$ qui maximisent $\log P$ sont celles pour lesquelles la probabilité empirique coïncide avec le vecteur p.*

Définition 1.6.20 (Entropie de Boltzmann) Soit X un espace fini et p un vecteur de probabilité sur X. On appelle *entropie de Boltzmann* de (X, p), et on note $S(p)$, la quantité

$$S(p) = -\mathbb{E}\log p = -\sum_{x\in X} p(x)\log p(x).$$

Proposition 1.6.21 *Le vecteur de probabilité p_0 qui maximise l'entropie de Botzmann sur l'espace fini X est le vecteur de probabilité uniforme*

$$p_0(x) = \frac{1}{\operatorname{card} X}.$$

Démonstration : On calcule $S(p_0) = -\mathbb{E}\log\frac{1}{\operatorname{card} X} = \log(\operatorname{card} X)$. Soit p un autre vecteur arbitraire de probabilité sur X. Alors $S(p) = -\sum_x p(x)\log p(x) \leq -\sum_x p(x)\log p_0(x)$ d'après l'inégalité de Gibbs. Or le dernier terme de cette expression vaut $\sum_x p(x)\log p_0(x) = \log(\operatorname{card} X)$. □

On peut maintenant introduire un modèle simplifié de gaz. Soit (X, p) un espace de probabilité fini et $U : X \to \mathbb{R}$ une fonction qui à chaque configuration $x \in X$ associe sa valeur d'énergie, On s'intéresse à un système isolé tel que l'énergie moyenne soit fixée à une valeur E, c'est-à-dire $\mathbb{E}U = \sum_x U(x)p(x) = E$. On veut résoudre le problème de maximisation d'entropie sous la contrainte d'energie moyenne fixée. Analytiquement, on doit donc déterminer le vecteur p qui maximise $S(p) = -\sum_x p(x)\log p(x)$ sous les contraintes $p(x) \geq 0$ pour tout x, $\sum_x p(x) = 1$ et $\sum_x p(x)U(x) = E$.

Théorème 1.6.22 *Le vecteur p_0 qui maximise l'entropie de Boltzmann à énergie moyenne fixée, est le vecteur de probabilité canonique de Gibbs,*

$$p_0(x) = \frac{\exp(-\beta U(x))}{Z(\beta)},$$

où $Z(\beta) = \sum_x \exp(-\beta U(x))$ est un facteur de normalisation appelé fonction de partition et β un paramètre positif, solution de l'équation

$$\frac{\sum_x U(x)\exp(-\beta U(x))}{Z(\beta)} \simeq -\frac{d}{d\beta}\log Z(\beta) = E$$

qui est physiquement interprété comme l'inverse de la température absolue.

Démonstration : On commence par calculer $S(p_0) = -\sum_x p_0(x)\log p_0(x) = \log Z(\beta) + \beta E$. Supposons que p soit un autre vecteur de probabilité qui vérifie $\sum_x p(x)U(x) = E$. Alors, par l'inégalité de Gibbs,

$$S(p) = -\sum_x p(x)\log p(x) \leq -\sum_x p(x)\log p_0(x) = \log Z(\beta) + \beta E.$$

□

Exercice 1.6.23 Soient (X_1, p_1) et (X_2, p_2) deux systèmes isolés à énergies moyennes respectives E_1 et E_2, où p_1 et p_2 sont les probabilités de Gibbs correspondantes. Si les deux systèmes sont mis en contact de sorte que leur énergie totale deviennne $E = E_1 + E_2$ et si p désigne la probabilité de Gibbs pour le système composé, on a l'inégalité

$$S(p) \geq S_1(p_1) + S_2(p_2).$$

Cet exercice montre que l'entropie ne peut qu'augmenter.

Exercice 1.6.24 Étendre toutes les définitions d'entropie de Boltzmann, de probabilité canonique de Gibbs, de fonction de partition, données dans un espace X fini au cas d'un espace mesuré arbitraire $(X, \mathcal{A}, \mu)$ où μ est une mesure positive.

La quantité d'information selon Shannon et Khinchin

Les premiers chercheurs qui ont essayé de définir mathématiquement la notion d'information contenue dans un message ont dressé une liste des propriétés que devrait vérifier la quantité d'information à partir d'une définition plausible. Ils ont ainsi procédé sans s'en rendre compte comme les thermodynamiciens du siècle dernier. L'anecdote veut que Claude Shannon, un des pionniers de l'information, en discutant un jour avec John von Neuman sur la définition qu'il avait proposée pour l'information, s'est rendu compte que ce qu'il venait de définir comme information n'etait que l'inverse de l'entropie de Boltzmann. En effet, on a tendance à considérer comme du bruit impertinent un message totalement aléatoire tandis qu'un message structuré est sensé contenir beaucoup d'information. Ainsi pour un espace fini X avec un vecteur de probabilité p, Shannon définit l'information $I(p) = -S(p)$. Aujourd'hui la théorie de l'information est mathématiquement bien établie [**?**] et [**?**]. Comme l'information est l'inverse de l'entropie, on ne répète les arguments donnés au paragraphe précédent mais on donne juste la définition dans un cadre un peu plus général.

Définition 1.6.25 Soit $(X, \mathcal{A}, \mu)$ un espace mesuré et $f : X \to \mathbb{R}^+$ une fonction intégrable. Soit $\zeta : \mathbb{R}^+ \to \mathbb{R}$ définie par $\zeta(u) = u \log u$ pour $u > 0$ et $\zeta(0) = 0$. On appelle *information* de f, la quantité

$$I(f) = \int_X \zeta(f(x))\mu(dx).$$

Sachant que les fonctions positives intégrables sur X ont une interpétation comme densités, la notion d'information est définie pour toute mesure positive sur X. Elle nous renseigne sur la « quantité d'information » véhiculée par une mesure de densité f. Parmi toutes les mesures sur X, avec $\mu(X) < \infty$, la probabilité uniforme minimise la fonction information.

Exercice 1.6.26 Montrer que si $\mu(X) < \infty$, la densité $f(x) = \frac{1}{\mu(X)}$ possède une information minimale $I(f) = \log \frac{1}{\mu(X)}$.

On peut étudier l'information contenue dans le système dynamique dans un stade ultérieure de son évolution, en examinant la quantité $I(P^n f_0)$, où P est l'opérateur de l'evolution markovienne ou l'opérateur de Perron et Frobenius du système. Intuititevment, une suite aléatoire contient une quantité d'information proportionnelle à la longueur de la suite. Voyons pourquoi sur un exemple.

Soit $X = [0,1[$ muni de sa tribu borélienne. Soit $T : X \to X$ définie par $T(x) = 2x \mathrm{mod} 1$. Pour $x = \langle 0, d_1 d_2 d_3 \ldots \rangle_2$, l'application $T(x)$ est le décalage à gauche avec amputation du bit de poids $T(x) = \langle 0, d_2 d_3 \ldots \rangle_2$.

Supposons que pour observer l'évolution nous disposons d'un instrument imparfait qui donne pour tout $y \in X$,

$$\mathsf{Obs}(y) = \begin{cases} 0 & \text{si } y \leq 1/2 \\ 1 & \text{sinon.} \end{cases}$$

On voit que d'après la définition de Obs, on a $\mathsf{Obs}(T^n x) = d_n$. Donc avec l'instrument imparfait de détection que nous disposons, nous pouvons le déterminer x avec une précision arbitraire (un nombre arbitraire de digits) si on l'oberve suffisamment longtemps. Comme l'information moyenne produite par unité de temps est de 1 bit, l'information contenue dans la suite de longueur n est n bits.

On peut calculer facilement l'exposant de Lyapunov de cette suite $\lambda = \log 2 > 0$ qui entraîne une sensibilité aux conditions initiales. Comme en observant l'évolution pour des temps de plus en plus longs, on arrive à déterminer la condition initiale de plus en plus précisemment, des différences qui restent inaperçues à petit temps peuvent se distinguer à des stades ultérieurs de l'évolution.

1.6.4 Mesures de Gibbs pour les systèmes dynamiques

Soit $(X, \mathcal{A}, \mu_0, T)$ système dynamique sur X compact métrique, T homéomorphisme de X et μ_0 mesure normalisée, invariante sous T.

On se sert de μ_0 pour choisir le point initial y_0 selon cette mesure. L'évolution du système dynamique entraîne que μ_0 induit une mesure sur les orbites émanant de y_0 de la même manière que pour une chaîne de Markov, la donnée de la mesure a priori et de la matrice stochastique déterminent de manière unique une mesure sur les trajectoires. On note abusivement par le même symbole μ_0 la mesure induite sur les orbites infinies du système dynamique.

Soit $h : X \to \mathbb{R}$ une fonction bornée, On définit la *fonction de partition à horizon fini* pour le système dynamique, la quantité

$$Z_{m,n}(h; \mu_0) = \int_X \exp\Big(\sum_{k=-m}^{n} h(T^k x)) \mu_0(dx).$$

On introduit une nouvelle mesure sur les orbites, $\mu_{m,n}(h) \ll \mu_0$ définie par

$$\frac{d\mu_{m,n}(h)}{d\mu_0}(x) = \frac{\exp(\sum_{k=-m}^{n} h(T^k x))}{Z_{m,n}(h; \mu_0)}.$$

Définition 1.6.27 Une mesure $\mu(h)$, limite faible de $\mu_{m,n}(h)$, est appelée *mesure de Gibbs* pour la fonction h.

On peut montrer que comme pour le cas fini, les mesures de Gibbs sur les orbites infinies des systèmes dynamiques sont les mesures qui maximisent l'entropie et sont obtenues alors par un principe variationnel. Malgré l'intérêt que présentent ces notions, cette ligne ne sera pas développée dans ce chapitre. Le lecteur intéressé pourra consulter l'article de Sinai [**?**]. On aura l'occasion par la suite d'aborder de nouveau la notion de mesure de Gibbs sous un autre aspect.

1.6.5 Suites pseudo-aléatoires et complexité de Kolmogorov

Pour simplifier la discussion, on va se limiter au cas des suites composées de zéros et de uns. On veut étudier dans quelles conditions ces suites peuvent être considérées comme de réalisations de suites de Bernoulli avec probabilité $1/2$.

On note

$$X = \cup_{k\in\mathbb{N}\cup\{+\infty\}}\{0,1\}^k$$

l'ensemble de toutes les suites de longueur arbitraire et $l(x)$ la longueur de chaque suite x, *i.e.* si $x \in \{0,1\}^k$ alors $l(x) = k$. Pour tout $n \leq l(x)$ on note $x|_n$ la restriction de x à ces n premiers éléments.

Pour qu'une telle suite ait des chances de correspondre à notre intuition de l'aléatoire, elle doit être

- stochastique, dans le sens qu'elle vérifie certaines propriétés de stabilité fréquentielle,
- chaotique, dans le sens qu'elle soit désordonnée avec une entropie de Kolmogorov (mesure de l'information contenue) proportionnelle à sa longueur et
- typique, dans le sens que les suites non-typiques sont dans un ensemble effectivement négligeable

Stochasticité

Les premiers débats et travaux mathématiques sur la nature de l'aléa sont dus à von Mises qui propose la notion de stabilité fréquentielle [**?**] comme caractérisation de l'aléa. On entend par là que si, par exemple, on considère une suite distribuée selon la loi de Bernoulli, la fréquence d'apparition de zéros tend vers $1/2$. A cette « définition » on peut objecter que la suite

$$010101010101\cdots$$

vérifie cette propriété mais n'a rien d'aléatoire. On pallie cet inconvénient en exigeant que non seulement la suite mais aussi des sous-suites vérifient la propriété de stabilité fréquentielle. Cependant, cela ne peut pas être vrai pour toutes les sous-suites. Pour s'en convaincre, supposons qu'une suite infinie $x = (x_1, x_2, x_3, \cdots)$ soit donnée et construisons la suite d'entiers définie par

$$n_1 = \inf\{n \geq 1 : x_n = 0\}$$

et récursivement, tant que n_{m-1} soit fini,

$$n_m = \inf\{n > n_{m-1} : x_n = 0\}.$$

Alors, dans le cas où $\inf\{m : n_m = +\infty\} = +\infty$, la sous-suite $(x_{n_1}, x_{n_2}, x_{n_3}, \cdots)$ ne vérifie pas — par construction — la propriété de stabilité fréquentielle. Il est donc évident que cette propriété doit être vraie uniquement pour certaines sous-suites qui remplissent des conditions particulières.

La bonne notion de stochasticité est la suivante : Soit w un mot binaire arbitraire fini. On considère une suite $x \in X$. Si x est de longueur infinie et aléatoire, le mot w apparaît une infinité de fois (cf. exemple du « singe dactylographe ») dans x. On isole la sous-suite de lettres qui suivent les apparitions de w. Cette sous-suite doit vérifier la propriété de stabilité fréquentielle pour que la suite x soit stochastique.

Corollaire 1.6.28 *Soit w un mot arbitraire tel que $l(w) = k$ et x une suite stochastique. Alors*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{x_i \cdots x_{i+k-1} = w\}}(i) = \frac{1}{2^k}.$$

En d'autres termes, toutes le suites stochastique sont de suites normales (au sens de Borel).

Chaoticité

Ici, on va se concentrer sur la notion d'aléa telle qu'elle a été formalisée par Kolmogorov en introduisant la notion de complexité. L'approche de Kolmogorov [?] consiste à introduire une mesure de complexité (chaoticité) et définir ensuite comme aléatoire toute suite vérifiant un critère de chaoticité. Considérons, par exemple, deux suites finies x et y avec $l(x) = l(y) = 1000$ telles que $x = (000000\cdots)$ tandis que y contient à la fois des 0 et des 1. Intuitivement, la suite y est plus complexe que x. En effet, x est décrite comme la suite des « mille zéros » tandis que la description de y doit nécessairement être plus longue puisqu'il faut indiquer les endroits précis où se trouvent les zéros. On va rendre cette intuition rigoureuse.

Définition 1.6.29 Une application $f : X \to X$ est dite *calculable* si elle vérifie les conditions suivantes :

- Si x est un segment initial d'une suite y alors la suite $f(x)$ est un segment initial de la suite $f(y)$.
- Si $l(x) = +\infty$, la valeur $f(x)$ est la suite minimale qui est continuation de *toutes* les suites $f(x|_n)$ pour $n = 1, 2, \cdots$.

Cette définition implique que toute application calculable peut être décrite à l'aide d'un algorithme *fini*. Intuitivement, f est un algorithme ; si on entre la suite x_0, x_1, ...au clavier d'un ordinateur qui exécute le programme f, on obtient

$$\begin{aligned} x_0 &\mapsto f(x_0) \\ x_0x_1 &\mapsto f(x_0x_1) \\ x_0x_1x_2 &\mapsto f(x_0x_1x_2) \\ &\vdots \end{aligned}$$

Définition 1.6.30 Soit $f : X \to X$ une application calculable fixée. On dit qu'une suite finie y *est décrite* par une suite finie x relativement à l'application f, si y est le segment initial de la suite (finie ou infinie) $f(x)$.

En d'autre termes, décrire la suite finie y consiste à trouver une suite finie x, telle que la suite (finie ou infinie) $f(x)$ commence par y, c'est-à-dire $f(x) = y \vee z$.

Définition 1.6.31 La *complexité (de Kolmogorov)* d'une suite x relativement à l'application calculable f, notée $\mathrm{KM}_f(x)$, est définie par

$$\mathrm{KM}_f(x) = \inf\{l(y) : y \text{ est une descritpion de } x \text{ relativement à } f\}.$$

Définition 1.6.32 Une application calculable $f : X \to X$ est dite *optimale* si pour toute application calculable g, il existe une constante C_1 telle que

$$\mathrm{KM}_f(x) \leq \mathrm{KM}_g(x) + C_1,$$

uniformément pour toute suite x.

Théorème 1.6.33 *[?] L'ensemble des applications optimales est non vide.*

Définition 1.6.34 On appelle *entropie* d'une suite sa complexité relativement à une application optimale.

Remarque : On voit que la notion d'entropie dépend de l'application optimale choisie. Il existe cependant une constante C_2 telle que si $\mathrm{KM}_1(\cdot)$ et $\mathrm{KM}_2(\cdot)$ sont des entropies relatives à deux applications optimales différentes, la différence

$$|\mathrm{KM}_1(x) - \mathrm{KM}_2(x)| \leq C_2$$

est uniformément bornée. On note alors $\mathrm{KM}(x)$ l'entropie à une constante près.

Si on choisit comme fonction calculable la fonction identité, $f = \mathrm{id}$, on observe que $\mathrm{KM}_f(x) = l(x)$. On a donc la borne évidente $\mathrm{KM}(x) \leq l(x) + \mathcal{O}(1)$ ce qui montre que l'entropie d'une suite ne peut pas excéder sa longueur que d'une constante. Les suites où l'inégalité précédente dévient une égalité sont les suites complexes :

Définition 1.6.35 Une suite infinie $x \in X$ est dite *chaotique* s'il existe une constante C telle que

$$|\mathrm{KM}(x|_n) - n| \leq C$$

pour tout $n \in \mathbf{N}$.

Ainsi, les suites chaotiques sont les suites dont la description nécessite un algorithme aussi long que la suite elle même. Les suites aléatoires doivent être chaotiques.

Revenant maintenant aux générateurs algorithmiques des nombres aléatoires, on constate que les suites produites ne sont pas chaotiques. Par exemple, l'entropie de la suite congruentielle est finie puisqu'il suffit de connaître la racine x_0 pour déterminer toute la suite.

En conclusion :

Toute suite de nombres aléatoires générée de manière algorithmique n'est pas chaotique !

Suites typiques

Reste à donner un sens précis à la notion de suite typique. On s'attend à ce qu'une telle suite vérifie tous les théorèmes connus de la théorie de probabilités. Ceci nous amène à la discussion des ensembles exceptionnels sur lesquels les théorèmes de probabilités sont violés.

Notons $X_x = \{y \in X : x \text{ est un segment initial de } y\}$.

Définition 1.6.36 Un ensemble $A \subset X$ est *négligeable* si pour tout $\epsilon > 0$, il existe une famille finie ou dénombrable de suites $\{x^{(i)}\}$, avec $x^{(i)} \in X$ pour tout i, telle que

$$A \subset X_{x^{(0)}} \cup X_{x^{(1)}} \cup X_{x^{(2)}} \cup \cdots$$

avec

$$2^{-l(x^{(0)})} + 2^{-l(x^{(1)})} + 2^{-l(x^{(2)})} + \cdots < \epsilon.$$

Un ensemble A est négligeable s'il existe un recouvrement de A par une famille d'ensembles des mots à segment initial fixé de masse totale inférieure à ϵ.

Définition 1.6.37 Un ensemble $A \subset X$ est dit *effectivement négligeable* s'il existe une application calculable $f : \mathbb{Q}^+ \times \mathbb{N} \to X$ telle que pour tout rationnel $\epsilon > 0$

$$A \subset X_{f(\epsilon,0)} \cup X_{f(\epsilon,1)} \cup X_{f(\epsilon,2)} \cup \cdots$$

($f(\epsilon, i)$ est un mot de X) avec

$$2^{-l(f(\epsilon,0))} + 2^{-l(f(\epsilon,1))} + 2^{-l(f(\epsilon,2))} + \cdots < \epsilon.$$

Remarque : On n'exige pas que f soit totale. Si f n'est pas définie pour certain couples, on utilise la convention $X_{f(\epsilon,n)} = \emptyset$ et $l(f\epsilon, n) = \infty$.

Un ensemble est *comptable* s'il existe un programme qui imprime tous ses éléments (programme sans fin si l'ensemble est infini). Un ensemble est comptable s'il est le domaine de valeurs d'une fonction calculable.

Théorème 1.6.38 *Un ensemble A est effectivement négligeable s'il existe un ensemble comptable $W \subseteq \mathbb{Q}^+ \times X$ tel que, pour tout $\epsilon > 0$,*

$$A \subseteq \cup_{(\epsilon,x)\in W} X_x$$

et

$$\sum_{(\epsilon,x)\in W} 2^{-l(x)} < \epsilon.$$

Théorème 1.6.39 (Martin-Löf) *Il existe un ensemble universel effectivement négligeable qui inclut tout ensemble effectivement négligeable.*

Définition 1.6.40 On appelle une suite *typique* si elle n'est pas contenue dans l'ensemble effectivement négligeable maximal.

On peut se sentir soulagé ! Il existe un ensemble que l'on peut construire à l'aide d'un algorithme fini ; toute suite aléatoire des zéros et des uns à l'extérieur de cet ensemble vérifie tous les théorèmes des probabilités.

1.7 Pseudo-aléatoire ou quasi-aléatoire ?

En guise de conclusion de ce chapitre, on peut dire que l'on dispose de plusieurs méthodes algorithmiques permettant de générer des suites de variables aléatoires. Les derniers paragraphes, sur les aspects probabilistes de systèmes dynamiques déterministes et sur la complexité de Kolmogorov, nous ont persuadé que *stricto sensu* on ne peut pas générer des suites vraiment aléatoires ; tout ce que l'on peut espérer c'est de générer des suites qui vérifient les théorèmes des probabilités. On verra dans la suite de ce livre que la méthode Monte Carlo est une méthode lente de résolution numérique. La précision de la solution après N étapes est de l'ordre de $\sqrt{N}$. On s'est donc posé la question de savoir si des suites algorithmiques peuvent donner de meilleurs résultats sous certaines conditions.

On verra que la réponse dépend de nos exigences. Si on veut construire des suites utilisables pour tout problème de simulation — elles doivent donc vérifier tous les théorèmes connus du calcul des probabilités — la réponse est non ; ces suites sont généralistes et on peut les utiliser à tout problème avec la certitude qu'elles fournissent lentement mais sûrement la solution recherchée. Ces suites sont appelées *pseudo-aléatoires*. Si, par contre, on aimerait que la suite ne vérifie qu'un théorème des probabilités, on peut construire de suites spécialisées qui donnent une solution rapide au problème étudié mais échoueront totalement à d'autres applications. Ces suites spécialisées sont appelées *quasi-aléatoires* ou *à faible écart*.

1.7.1 Exemples de suites à fiable écart

Suites de Richtmyer

Fixons un vecteur $(\alpha_a, \ldots, \alpha_n) \in \mathbb{R}^n$. On forme la suite $\mathbf{x}_k \in [0,1]^n$ pour $k = 1, 2, 3, \ldots$ par

$$\mathbf{x}_k = \begin{pmatrix} k\alpha_1 \\ \vdots \\ k\alpha_n \end{pmatrix}.$$

Cette suite, si les $\alpha_i \in \mathbb{R} \setminus \mathbb{Q}$, est appelée une suite de Richtmyer.

Elle a un comportement très différent du comportement de la suite congruentielle scalaire. Elle vérifie le théorème de grands nombres mais elle échoue à des épreuves statistiques.

Suties de van der Corput

Si on fixe un entier b, tout entier n a une décomposition en base b,

$$n = \sum_{i=0}^{\infty} d_i(n) b^i.$$

Quoique la somme ci-dessus s'étend jusqu'à l'infini, en réalité pour tout n, il n'y a que $\mathcal{O}(\log_b n)$ digits non-nuls ; il s'agit donc d'une somme finie.

On introduit la tranformée radicale inverse par

$$\phi_b(n) = \sum_{i=0}^{\infty} \frac{d_i(n)}{b^{i+1}} \in [0,1].$$

La suite de van den Corput est alors définie par

$$x_k = \phi_b(k), \text{pour } k = 1, 2, \ldots$$

1.8 Exemples des mauvais générateurs (encore !) utilisés de nos jours

Les habitudes sont difficiles à changer, surtout les mauvaises ! Il est étonnant de constater la paresse intelectuelle de plusieurs praticiens de la simulation qui, au lieu de programmer des générateurs convenables, s'obstinent à utiliser des générateurs avec des périodes très courtes ou avec de très mauvaises propriétés statistiques. Voici une liste, malheureusement non exhaustive, de quelques atrocités utilisées par le passé et qui continuent à être encore utilisées.

– Générateur `RANDU` : Il s'agit de la congruence

$$x_n = 65539 x_{n-1} \mathrm{mod} 2^{31} \text{ pour } n = 1, 2 \ldots.$$

Ce générateur était fourni avec la bibliothèque mathématique du système IBM 360. Même en utilisant comme racine x_0 un premier — ce qui n'était nullement mentionné dans la documentation de la bibliothèque — afin de maximiser la période à 2^{29}, ce générateur a de très mauvaises propriétés statistiques,

- Fonction `Random` en PASCAL ISO, utilisé aussi par le logiciel SAS. Il s'agit du générateur

$$x_n = 16807 x_{n-1} \mathrm{mod} 2^{31} \text{ pour } n = 1, 2 \ldots .$$

Étant donné que $16807 \mathrm{mod} 8 = 7$ ce générateur a une période encore plus courte que le précédent. En outre, il présente certaines caractéristiques si manifestement non aléatoires que le concepteur de logiciel SAS lui a adjoint, dans des versions antérieurs de son logiciel, une fonction de « battage » pour le rendre plus aléatoire. Ce « bricolage » rend cependant un très mauvais service parce qu'il n'améliore pas significativement ses propriétés statistiques tout en brouillant les pistes ; cette action, émanant d'un concepteur spécialisé dans les logiciels statistiques est purement incompréhensible.

- Générateur `rand` du ssytème UNIX. Il s'agit du générateur

$$x_n = (1103515245 x_{n-1} + 12345) \mathrm{mod} 2^{31} \text{ pour } n = 1, 2 \ldots .$$

Le constructeur met en garde contre le mauvais comportement statistique de ce générateur mais il l'implémente tout de même !

- Générateur `random` de TURBOPASCAL. Il s'agit du générateur

$$x_n = (129 x_{n-1} + 907633385) \mathrm{mod} 2^{32} \text{ pour } n = 1, 2 \ldots .$$

Ce générateur a de très mauvaises propriétés statistiques. Pour pallier les manques de ce générateur, le constructeur utilise, dans les versions plus récentes, une « amélioration » qui consiste à calculer le le nombre $U \in [0, 1]$ par l'opération suivante :

$$\begin{aligned} U'_n &= X_n / 2^{31} \\ U_n &= \begin{cases} 2 - U'_n & \text{si } U'_n > 1 \\ U'_n & \text{sinon.} \end{cases} \end{aligned}$$

Émanant d'un concepteur qui a basé sa réputation sur les qualités pédagogiques de son langage pour l'apprentissage de la programmation structurée, l'implémentation de ce générateur ne peut que laisser encore plus perplexe.

En résumé, il faut éviter à tout prix d'utiliser des générateurs installés si on n'a pas accès aux fichiers source de la programmation. De nos jours il n'y a aucune raison de continuer à utiliser ces générateurs obsolètes et dangereux.

1.9 Calcul distribué et générateurs des nombres au hasard

Les simulations stochastiques sont des méthodes numériques qui convergent très lentement. De nos jours, on dispose plusieurs méthodes pour réduire le temps nécessaire à l'obtention d'un résultat avec une précision donnée. L'idée de base de tous ces méthodes est d'effectuer certains étapes du programme parallèlement au lieu de les exécuter séquentiellement. Ceci peut se réaliser de plusieurs manières :

1. *Par vectorisation du processeur.* Le processeur central dispose à côté des registres scalaires des registres vectoriels qui permettent d'effectuer des blocs de boucles en une étape d'horloge (cf. annexe ? ? ?) comme sur le CRAY-1 par exemple.
2. *Par parallélisation du programme.* On se sert des divers processeurs d'un ordinateur multiprocesseur en parallèle comme sur le SUN Entreprise par exemple.
3. *Par distribution du calcul sur un réseau d'ordinateurs interconnectés.* On se sert des processeurs de divers ordinateurs. La parallélisation se fait alors de manière logicielle, en se servant de programmes tels que Virtual Parallel Machine (VPM) ou MPI pour distribuer les tâches à travers le réseau.

S'agissant de simulation Monte Carlo, il y a cependant une difficulté qui surgit lorsqu'on tente de paralléliser l'algorithme de simulation. Pour illustrer ce problème, prenons l'exemple du générateur standard avec une racine x_0. Ce générateur est défini par le système dynamique (X, T) avec $X = \{0, \ldots, 2^{31} - 2\}$ et $Tx = 16807x \bmod (2^{31} - 1)$. L'espace X est partitionné par cette évolution en deux parties invariantes sous T : un singleton contenant $\{0\}$ contenat l'unique point fixe 0 et son complémentaire $X \setminus \{0\}$ contenant l'unique orbite périodique (en vertue du théorème **??**). Initialiser le générateur avec une autre racine x_0' équivaut à parcourir la même orbite périodique (puisqu'elle est unique) à partir d'un autre point d'entrée. Si x_0 et x_0' sont deux racines arbitraires différentes, on note

$$\tau_{x_0,x_0'} = \inf\{n \geq 1 : T^n(x_0) = x_0'\}$$

le temps nécessaire pour que l'orbite émanant de x_0 atteigne x_0' et qui est nécessairement fini. Supposons donc qu'en parallélisant le programme, on l'a scindé en deux sous-tâches dont chacune nécessite K nombres au hasard. Si on initialise le générateur avec deux racines x_0 et x_0' telles que $\tau_{x_0,x_0'} < K$, une partie de la suite de variables utilisées par la première sous-tâche sera *identique* à une partie de la suite utilisée par la deuxième sous-tâche. Donc les deux sous-tâches ne seront pas indépendantes.

Une manière peu coûteuse pour pouvoir paralléliser est suggérée par le tableau **??** où l'on a parcouru la totalité de l'orbite émanant de $x_0 = 1$ et en marquant la valeur de $x_{k\times 10^8}$ pour $k = 0, \ldots, 21$. Ceci nous permet de disposer de 21 lots de 10^8 valeurs pseudo-aléatoires indépendantes[4].

De méthodes plus avancées sont utilisées pour construire de générateurs paralélisables plus performants (cf. article de Marsaglia [**?**]).

Notices bibliographiques

La meilleure source d'information sur le générateurs continue à être le livre de Knuth [**?**].

Une bonne introduction mathématique aux systèmes dynamiques est fournie par l'excellent livre de Lasota et MacKey [**?**] tandis que des aspects plus intuitifs sont présentés dans la revue d'Eckmann et Ruelle [**?**]. L'approche gibbsien est donné dans [**?**].

Les développements sur les notions de stochasticité, chaoticité et complexité de Kolmogorov sont très bien présentés dans l'article d'Uspenskii, Semenov et Shen [**?**].

[4]Attention, le 22e lot ne contient que 47 483 647 valeurs indépendantes.

iter	Racine(iter)	iter	Racine(iter)
0	1	1 100 000 000	1 103 177 160
100 000 000	1 209 575 029	1 200 000 000	999 244 642
200 000 000	449 294 716	1 300 000 000	552 035 842
300 000 000	1 292 894 662	1 400 000 000	1 218 407 032
400 000 000	527 371 189	1 500 000 000	1 947 017 488
500 000 000	1 912 880 129	1 600 000 000	1 749 956 682
600 000 000	52 980 039	1 700 000 000	904 907 723
700 000 000	2 116 779 609	1 800 000 000	1 982 270 339
800 000 000	87 504 891	1 900 000 000	2 354 258
900 000 000	184 641 623	2 000 000 000	325 871 955
1 000 000 000	933 757 703	2 100 000 000	1 871 752 643

TAB. 1.3 – En initialisant avec `Racine(iter)` on peut utiliser le générateur standard pour créer un lot de 100 000 000 nombres au hasard indépendemment du reste. En utilisant la racine correspondante à une autre valeur de `iter`, on est certain de disposer d'un autre lot de 100 000 000 nombres au hasard indépendant du précédent qui pourrait être généré en parallèle.

1.10 Exercices

1. Vous pouvez utiliser la routine généraliste suivante pour programmer l'algorithme de génération de variables aléatoires indépendantes selon la loi uniforme sur $[0,1]$ en utilisant la méthode congruentielle (affine).

```
subroutine unifgl(a,c,m,ix,u)
double precision a, c, m, ix
intefer k1
real u
ix = a * ix + c
k1 = ix / m
ix = ix - k1 * m
u = ix / m
end
```

   Cette routine, par l'utilisation des variables `double precision` permet de traiter convenablement les dépassements de éventuels sur les machines de 32 bits. Application : $a = 16807$, $c = 0$ et $m = 2^{31} - 1$.

2. Une manière différente pour programmer le générateur

$$x_i = ax_{i-1} \mathrm{mod} m$$

   en en faisant usage que d'entiers-machine consiste à écrire le sous-programme FORTRAN suivant, utilisant une astuce de programmation[5] due à [?] :

```
subroutine unif(ix,u)
k1 = ix/127773
ix = 16807*(ix-k1*127773)-k1*2836
```

[5] L'instruction `if (ix.lt.0) ix = ix+2147483647` peut paraître quelque peu surréaliste ; elle est en outre spécifique des ordinateurs à 32 bits. Si on veut comprendre sa raison d'être, on peut utilement consulter l'annexe D.

```
if (ix.lt.0) ix = ix+2147483647
u = ix*4.656612875e-10
end
```

Comparer les résultats obtenus pour 1000000 tirages avec le programme généraliste et le programme ci-dessus. (Si l'initialisation se fait avec la même racine, il faut que les deux suites obtenues soient identiques).

3. Utiliser la routine d'appel de l'horloge de votre système informatique pour comparer les vitesses respectives du programme généraliste et du programme de Bratley.
4. Se servir de la routine `unif` pour générer des variables aléatoires indépendantes et discrètes, uniformément distribuées sur $\{0, \cdots, k-1\}$, pour k un entier donné.
5. Programmer l'algorithme polaire pour la génération de variables aléatoires distribuées selon la loi $\mathcal{N}(0,1)$. Se servir de ce programme pour générer une suite de variables aléatoires distribuées selon la loi $\mathcal{N}(m, \sigma^2)$.
6. Programmer l'algorithme de génération de variables aléatoires distribuées selon la loi exponentielle de paramètre λ.
7. Programmer l'algorithme de génération de variables aléatoires distribuées selon la loi χ^2_ν.
8. Une variable aléatoire X suit la loi de Cauchy centrée en a et de largeur à mi-hauteur b si

$$\mathbb{P}(X \in dx) = \frac{b}{\pi(b^2 + (x-a)^2)} dx.$$

Proposer un algorithme pour échantillonner selon la loi de Cauchy en disposant d'un générateur de loi uniforme sur l'intervalle $[0,1[$. Programmer cet algorithme.

9. Soient $(X_n)_{n=1,\cdots,N}$ et $(Z_n)_{n=1,\cdots,N}$ deux suites de variables aléatoires indépendantes, distribuées selon la loi uniforme sur $[0,1]$ mais qui ne sont pas mutuellement indépendantes. En supposant que leur coefficient de corrélation est $\rho(X_n, Z_n) = \alpha > 0, \forall n$ et en supposant que (X_n, Z_n) sont les coordonnées cartésiennes d'un point aléatoire dans le carré $[0,1]^2$, quelle est à votre avis la forme du nuage des points $(X_n, Z_n)_{n=1,\cdots,N}$?
10. Générer deux suites $(X_n)_{n=1,\cdots,N}$ et $(Y_n)_{n=1,\cdots,N}$ de variables aléatoires indépendantes et mutuellement indépendantes, distribuées selon la loi uniforme sur $[0,1]$. Pour un paramètre α donné, construire une suite $(Z_n)_{n=1,\cdots,N}$ de variables aléatoires indépendantes dont le coefficient de corrélation avec les variables $(X_n)_{n=1,\cdots,N}$ est α. En considérant que (X_n, Z_n) sont les coordonnées cartésiennes d'un point dans le carré $[0,1]^2$, afficher les N points ainsi obtenus pour diverses valeurs de $\alpha \in [-1,1]$. Votre prédiction faite lors de la question précédente était-elle correcte ?
11. Pour le système dynamique donné par la transformation de l'intervalle $[0,1]$ par l'application $S(y) = 4y(1-y)$ tracer les graphes de 4 premières itérées Pf_0, $P^2 f_0$, $P^3 f_0$ et $P^4 f_0$ pour $f_0 \equiv 1$ et P l'opérateur de Perron-Frobenius. Montrer que $f_* = 1/\pi\sqrt{y(1-y)}$ est un point fixe de P. Tracer le graphe de f_*. Quelle est votre conjecture ?
12. Calculer l'opérateur de Perron et Frobenius pour $T(y) = ry \mathrm{mod} 1$, $r \in \mathbb{N}$ et estimer numériquement la limite $\lim_{n\to\infty} P^n f(y)$ pour $f(y) = 2y \mathrm{mod} 1$ et $y \in [0,1]$.
13. Pour $X = [0,1]^2$ on définit trois transformations $T_k : X \to X$, $k = 1,2,3$ par

$$\begin{aligned} T_1(x,y) &= (\sqrt{2}+x, \sqrt{3}+y) \mathrm{mod} 1 \\ T_2(x,y) &= (x+y, x+2y) \mathrm{mod} 1 \\ T_3(x,y) &= (3x+y, x+3y) \mathrm{mod} 1 \end{aligned}$$

Choisir 10000 points $A = \{Y_0^i, i = 1, \ldots, 10000\}$ régulièrement disposés dans $[0, 0.1[^2$ et afficher les 6 premières itérées de A pour les trois applications ci-dessus.