Université de Lorraine — École Nationale Supérieure des Mines de Nancy
Département « Génie Industriel & Mathématiques Appliquées »
Parcours « Ingénierie Mathématique » — Semestre 8
Année universitaire 2024–2025

# Méthodes de Monte-Carlo *

édition spéciale du 4 avril 2025

*par Rémi* Peyre †, *maitre de conférences*

∗. Réf. 8KUAAN11

†. remi.peyre@mines-nancy.univ-lorraine.fr

# Table des matières

# Chapitre 1

# Préambule

## 1.1 Méthodes de Monte-Carlo

### Qu'est-ce qu'une méthode de Monte-Carlo ?

L'objet de ce cours est de montrer comment on peut se servir de simulations *aléatoires* pour mener des calculs *déterministes*. D'où le titre de ce cours, des telles méthodes stochastiques étant appelées « méthodes de Monte-Carlo » :

**Définition (AA).** De manière générale, ce qu'on appelle une *méthode Monte-Carlo* est une technique visant à calculer une quantité *déterministe* par le biais d'un procédé *aléatoire*. ♡

*Exemple* (AB). Une des plus célèbres méthodes Monte-Carlo (à laquelle un chapitre de ce cours sera consacré) est le *recuit simulé*, qui sert à trouver le minimum *global* d'une fonction *non convexe* de $\mathbb{R}^d$ dans $\mathbb{R}$, en particulier lorsque la dimension $d$ est grande. L'idée consiste à suivre une méthode de descente de gradient perturbée par un bruit dont l'intensité diminue progressivement. En termes mathématiques, notant $f$ la fonction à minimiser (supposée de classe $\mathrm{C}^1$), partant d'un point $X_0 \in \mathbb{R}^d$ arbitraire, on simule la solution de l'ÉDS

$$\mathrm{d}X_t = -\nabla f(X_t)\mathrm{d}t + \sigma(t)\mathrm{d}B_t, \tag{AC}$$

où $\sigma(t)$ est une fonction déterministe qui tend lentement vers 0.

On peut alors montrer que, sous certaines conditions assez générales (coercivité de $f$, décroissance suffisamment lente de $\sigma(t)$), quel que soit le point de départ choisi, $f(X_t)$ converge en loi vers la constante $\min f$ — ce qui implique aussi, si $f$ atteint son minimum en un unique point $x_*$, que $X_t$ converge en loi vers $x_*$. ♧

Dans ce cours, la méthode que nous étudierons le plus en détail, qui fera l'objet de la partie I, est la méthode de Monte-Carlo pour le calcul d'une espérance. Si la méthode en tant que telle est relativement simple (chapitre 2), un enjeu important pour bien la maitriser en pratique est de connaitre et de savoir implémenter un certain nombre de techniques, dites de « réduction de la variance » (chapitre **??**), qui permettent de rendre les calculs considérablement plus efficaces dans de nombreux cas.

Dans une seconde partie, nous profiterons du thème de ce cours pour introduire les processus stochastiques à temps continu, qui sont un outil essentiel de la modélisation probabiliste (en particulier dans les domaines de gestion du risque en

assurance et en finance, mais plus généralement pour l'étude de toutes sortes de séries temporelles à haute fréquence : consommation électrique, progression d'une épidémie, trajectoire d'un avion soumis à des turbulences, ...), et qui sont justement un cadre dans lequel on peut fréquemment être amené à utiliser des méthodes de Monte-Carlo. Les processus stochastiques en tant continu sont un domaine très vaste auquel le présent cours se limitera à une initiation. Nous nous focaliserons plus spécifiquement sur le cas des processus à temps continu *markoviens*, qui peuvent être considérés comme la limite, vue sur une grande échelle de temps, du concept de chaines de Markov qui vous a été présenté au semestre précédent. Deux types de processus stochastiques retiendront particulièrement notre attention : d'abord les *processus à sauts*, qui sont un outil simple, mais néanmoins indispensable en pratique (de nombreux phénomène réels évoluant par sauts (l'évolution du score au cours d'un match de rugby, par exemple !), et dont on peut montrer qu'il permet en fait, dans un certain sens, d'approximer n'importe quel processus stochastique markovien. Ensuite, nous verrons un cas très important de processus stochastiques : les (solutions des) *équations différentielles stochastiques*[*] (EDS), qui sont une limite naturelle de nombreux phénomènes stochastiques lorsque le bruit de manifeste sur des échelles de temps abritrairement petites.

*Remarque* **(AD)** (Méthodes quasi-Monte-Carlo). Je place ici un mot sur les méthodes « quasi-Monte-Carlo », qui ne sont pas au programme de ce cours, mais que la plupart d'entre vous rencontreront probablement au cours de leur parcours d'ingénieurs : autant donc que ce polycopié vous mentionne de quoi il s'agit ! ☺

(Ce qui suit est écrit en supposant déjà connue la méthode de Monte-Carlo pour le calcul d'une espérance). L'écueil principal de la Monte-Carlo pour le calcul d'une espérance est son taux de convergence en $O(N^{-1/2})$, ce qui est plutôt lent... Il se trouve que, notamment dans le calcul d'intégrales de pas trop grande dimension, on peut sous certaines conditions faire sensiblement mieux avec la méthode dite *quasi-Monte-Carlo*. Cette méthode consiste, comme la méthode de Monte-Carlo « standard », à estimer $\mathbb{E}(f)$ par une quantité de la forme

$$N^{-1} \sum_{i=1}^{N} f(x_i), \tag{AE}$$

à ceci près que les $x_i$ ne sont plus aléatoires ! On peut en effet montrer que, sous réserve de certaines propriétés de régularité de $f$, il est possible de choisir les $x_i$ (d'une façon qui ne dépend pas du comportement spécifique de $f$) de sorte que la convergence de cette quantité soit en $O(N^{-1})$.

Cependant, malgré sa vitesse de convergence largement supérieure, la méthode quasi-Monte-Carlo n'est pas une panacée pour autant : les conditions de régularité à exiger sur $f$ sont plus restrictives ; la précision de la méthode n'est pas facile à évaluer ; et la méthode n'est pas aussi robuste à la dimension que la méthode de Monte-Carlo... Quoi qu'il en soit, malgré la similarité des formules et du nom, les bases mathématiques de la méthode de Monte-Carlo et de la méthode quasi-Monte-Carlo n'ont en fait pas grand-chose à voir, cette dernière ne faisant essentiellement aucun appel au hasard. ♧

[*]. En réalité, ce que les mathématiciens entendent par « équation différentielle stochastique » peut être plus général que ce que nous traiterons dans ce cours : disons donc que nous nous intéresserons aux EDS "standard".

### Historique des méthodes de Monte-Carlo

Historiquement, c'est en 1949 que le physicien gréco-américain Nicholas Metropolis et le mathématicien américain d'origine polonaise Stanisław Ulam publient l'article fondateur de cette méthode de calcul et lui donnent son nom. Pour être plus exact, l'idée de procéder à des tirages aléatoires pour évaluer des intégrales compliquées était dans l'air du temps parmi la communauté des physiciens, mais l'apport majeur de Metropolis & Ulam fut de proposer la technique d'échantillonnage préférentiel (cf. § 4.1), qui améliore largement l'efficacité de la méthode... Pour l'anecdote, c'est dans le cadre des recherches du projet *Manhattan* sur le développement de la bombe atomique que ces chercheurs (avec quelques autres dont notamment John von Neumann) avaient commencé à développer leurs idées.

Des développements importants des méthodes de Monte-Carlo furent l'algorithme de Metropolis-Hastings pour la simulation de certaines variables aléatoires en physique statistique (travaux dus notamment à Marshall Rosenbluth en 1953 et à Keith Hastings en 1970), algorithme qui à son tour fut la base de la méthode du recuit simulé (1983) pour trouver des extrema globaux de fonctions définies sur des espaces de grande dimension. Plus récemment (2008), on a aussi parlé des méthodes de Monte-Carlo à l'occasion de leur utilisation dans des logiciels joueurs de go (très grossièrement, l'idée est que l'ordinateur évalue la qualité d'une position en imaginant que les joueurs terminent leur partie en jouant au hasard : c'est ce qu'on appelle le *Monte Carlo Tree Search*), où ces méthodes ont permis des progrès spectaculaires [†].

La dénomination « Monte-Carlo [‡] » provient simplement de l'appel au hasard dans les algorithmes, par allusion au célèbre quartier de Monaco réputé pour son casino...

## 1.2 Points de formalisme

### Quelques notations

— Dans ce cours, je considèrerai que le symbole '$\mathbb{P}$' désigne simplement le *concept* « probabilité », et non pas une loi de probabilité en particulier. Dans de nombreux cas, une seule loi de probabilité entre en jeu, et dans ce cas $\mathbb{P}$ désigne implicitement cette loi-ci. Sinon, on se servira d'un exposant au-dessus du symbole '$\mathbb{P}$' (ou d'un de ses dérivés, comme '$\mathbb{E}$', 'Var', &c.) pour préciser le contexte probabiliste ; et les lois de probabilité spécifiques seront notées par des lettres comme $\boldsymbol{P}$, $\boldsymbol{Q}$, ..., mais jamais par le symbole '$\mathbb{P}$' à double-barre. Par exemple, l'expression « $\mathbb{P}^{X\sim P}(f(X)\in A)$ » se lira « probabilité, pour $X$ une variable aléatoire de loi $\boldsymbol{P}$, que $f(X)$ appartienne à $A$ », et désignera donc, $\boldsymbol{P}$ étant une certaine distribution de probabilité (sur un espace que nous pouvons convenir d'appeler « $\mathcal{X}$ »), la masse donnée par $\boldsymbol{P}$ à l'ensemble $\{x\in\mathcal{X} \mid f(x)\in A\}$ des éléments de $\mathcal{X}$ dont l'image par $f$ est dans $A$ : dans le formalisme que vous avez appris en cours de probabilité, c'est que qu'on aurait noté $\boldsymbol{P}(f^{-1}(A))$. De même, « $\mathbb{E}^{X\sim P}(f(X))$ » se

[†]. Avec notamment le logiciel *MoGo*, développé au sein de laboratoires de recherche français ; et dont certaines idées ont été reprises par *AlphaGo*, qui fut en 2017 le premier logiciel de Go à battre le champion du monde pour le format de jeu standard...

[‡]. Noter que les deux mots sont reliés par un tiret en français, mais pas en anglais. — Dans tous les cas, ne pas mettre d'accent sur le 'e' de « Monte », qui est un mot de langue ligure...

réfère à la quantité $\int_{x\in\mathcal{X}} f(x)\boldsymbol{P}(\mathrm{d}x)$ [§]. L'intérêt de ces notations peut vous apparaitre obscure à première vue ; mais dans certains cas elles permettent d'avoir des expressions à la fois bien plus compactes et bien plus parlantes pour les quantités qu'on souhaite manipuler... !

— La mesure de Lebesgue sur $\mathbb{R}^d$ est notée $\mathrm{vol}_d(\bullet)$, ou plus simplement « vol(•) » en l'absence d'ambigüité.

— La notation « d$x$ » pourra désigner un accroissement infinitésimal de la quantité $x$, mais aussi, selon le contexte, une zone infinitésimale autour de $x$ (par exemple dans la formule de transfert : $\mathbb{E}^{X\sim P}(f(X)) = \int f(x)\boldsymbol{P}(\mathrm{d}x)$, ou dans la définition de la densité : $\big(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q}\big)(x) = \boldsymbol{P}(\mathrm{d}x) \,/\, \boldsymbol{Q}(\mathrm{d}x)$).

— Je reprendrai certaines conventions utilisée en cours de statistique : en particulier, les variables aléatoires seront généralement notées par des majuscules romaines ($\boldsymbol{A}, \boldsymbol{B}, \ldots$) ou par des minuscules grecques *grasses* ($\boldsymbol{\alpha}, \boldsymbol{\beta}, \ldots$), et les valeurs qu'elles peuvent prendre seront alors désignées par la minuscule romaine ou la minuscule grecque *maigre* correspondante ($a, b, \ldots$, resp. $\alpha, \beta$, $\ldots$) ; quant à la valeur particulière prise par la v.a. lors de l'expérience réelle, on la distingue en ajoutant une coche (✓) en indice ($a_{\checkmark}, b_{\checkmark}, \alpha_{\checkmark}, \beta_{\checkmark}, \ldots$).

— Dans les codes `python`, nous supposerons toujours importées, *sans* préfixe, les fonctions des modules `math` et `random` (ces modules sont en effet tellement fondamentaux pour ce cours qu'on a envie d'utiliser leurs fonctions directement). Les exemples d'exécution de code données dans le polycopié sont normalement précédés implicitement de `set.seed(0)`, afin qu'on puisse d'assurer de leur reproductibilité. On supposera également toujours importé le module `time`, afin de pouvoir mesurer la performance des codes. Lorsque le code fait appel à certaines fonctions auxiliaires créées pour la circonstance, celles-ci seront supposées provenir d'un module `aux` : ainsi, si quelque part le code contient la ligne « `aux.simuler()` », cela signifie qu'on est censé avoir écrit en annexe une fonction `simuler` effectuant la tâche indiquée par son nom (en l'occurrence, une simulation ◡̈), mais qu'inclure explicitement le code de `simuler` dans le polycopié m'aurait paru excessif. Un répertoire contenant l'ensemble des codes du polycopié sera mis à disposition en accompagnement de ce polycopié.

## Concepts statistiques

Dans la mesure où un certain nombre de résultats de ce cours s'exprimeront à l'aide du vocabulaire de la statistique, j'ai voulu rappeler dans cette section les concepts statistiques que nous serons appelés à mentionner.

**Définition (AF)** (Modèle statistique)**.** Un modèle statistique est la donnée d'un espace mesurable $\mathcal{X}$ et d'une famille de distributions de probabilité $\theta \mapsto \mathrm{Loi}(\boldsymbol{X} \mid \boldsymbol{\theta} = \theta) \in \mathscr{M}_1(\mathcal{X})$ pour $\theta$ parcourant un certain espace $\boldsymbol{\Theta}$. On appelle $\boldsymbol{\theta}$ le *paramètre caché*. Une *expérience statistique* est le tirage au sort d'une réalisation de la loi $\mathrm{Loi}(\boldsymbol{X} \mid \boldsymbol{\theta} = \theta)$, où le paramètre caché $\theta_{\checkmark}$, quoique fixé, est *inconnu*. Cette réalisation (qui serait ici notée '$x_{\checkmark}$') est appelée *observation (effective)*. ♡

[§]. La notation $\mathbb{P}^{X\sim P}$ sous-entend également que $\boldsymbol{X}$ est la seule quantité fondamentale sur laquelle porte l'aléa relatif au symbole '$\mathbb{P}$' : par exemple, dans l'expression « $\mathbb{P}^{X\sim P}(\boldsymbol{X} \leqslant \boldsymbol{Y})$ », on signifie que $\boldsymbol{Y}$ doit être traitée comme une constante à l'intérieur de la parenthèse.

**Définition (AG)** (Quantité d'intérêt)**.** Une *quantité d'intérêt* (explicative) est, en toute généralité, une fonction *déterministe* de $\boldsymbol{\theta}$, que nous noterons traditionnellement $\gamma(\boldsymbol{\theta})$. ♡

**Définition (AH)** (Estimateur)**.** Un *estimateur* est, en toute généralité, une fonction *déterministe* de l'observation $X$ (ici vue comme une variable aléatoire). ♡

**Définition (AI)** (Estimateur convergent)**.** Une famille d'estimateurs $(\hat{\gamma}^{(n)}(X))_{n\in\mathbb{N}}$ étant donnée [¶], on dit que $\hat{\gamma}^{(n)}(X)$ est un estimateur *convergent* d'une quantité d'intérêt $\gamma(\boldsymbol{\theta})$ quand, *quelle que soit la valeur $\theta$ du paramètre caché dans $\boldsymbol{\Theta}$*, sous la loi $\mathbb{P}(\bullet \mid \boldsymbol{\theta} = \theta)$, la variable aléatoire $\hat{\gamma}^{(n)}(X)$ converge en probabilité vers $\gamma(\theta)$ lorsque $n \to \infty$. ♡

**Définition (AJ)** (Estimateur fortement convergent)**.** Dans la définition ci-dessus, quand on peut remplacer « en probabilité » par « presque-surement », on dit $\hat{\gamma}^{(n)}$ est un estimateur *fortement convergent* de $\gamma(\boldsymbol{\theta})$. ♡

**Définition (AK)** (Intervalle de confiance)**.** Un intervalle $I(X)$ (c'est-à-dire une statistique qui a la forme d'un intervalle) est appelée *intervalle de confiance au niveau de confiance* $(1-\alpha)$ (ou *au niveau de risque* $\alpha$) pour la quantité d'intérêt $\gamma(\boldsymbol{\theta})$ si, pour tout $\theta \in \boldsymbol{\Theta}$,

$$\mathbb{P}(I(X) \not\ni \gamma(\theta) \mid \boldsymbol{\theta} = \theta) \leqslant \alpha. \tag{AL}$$

♡

**Définition (AM)** (Intervalle de confiance asymptotique)**.** Un intervalle $I(X)$ dépendant d'un entier $n$ (c'est-à-dire une suite d'intervalles $I^{(n)}(X)$, mais souvent le paramètre $n$ est sous-entendu) est appelé *intervalle de confiance asymptotique au niveau* $(1-\alpha)$ pour la quantité d'intérêt $\gamma(\boldsymbol{\theta})$, si, pour tout $\theta \in \boldsymbol{\Theta}$,

$$\overline{\lim_{n\to\infty}}\, \mathbb{P}(I^{(n)}(X) \not\ni \gamma(\theta) \mid \boldsymbol{\theta} = \theta) \leqslant \alpha. \tag{AN}$$

♡

## Outil informatique

Les méthodes de Monte-Carlo sont indissociables de l'outil informatique : ce sont en effet des procédures très gourmandes en calcul, qu'il serait impossible d'implémenter efficacement à la main ! De nombreux codes informatiques accompagneront donc ce polycopié, et de nombreux exercices de travaux dirigés vous demanderont de répondre sous forme de codes informatiques. Qui dit code informatique dit langage de programmation ; et par souci de cohérence, il m'a fallu choisir un langage de programmation unique pour l'ensemble de ce cours... Mon choix s'est porté sur python (v3.*), un langage généraliste que vous avez déjà l'habitude de manipuler, ce qui vous permettra de vous focaliser sur les algorithmes en eux-mêmes plutôt que sur les subtilités de codage. Néanmoins, il faut bien être conscients que, dans les situations pratiques auxquelles vous serez confrontés, les langages utilisés par votre entreprise pourront être totalement différents : C++, Java, VBA, ... ! Il est donc important de ne voir python que comme un *exemple* de langage de programmation servant à *illustrer* les concepts algorithmiques de ce cours : mais ce qui compte, ce sont les algorithmes *en eux-mêmes*... ! Pour cette raison, je m'efforcerai de limiter autant que possible d'employer des fonctionnalités de python de trop "haut niveau" : par exemple, il existe des modules de python qui pré-implémentent une fonction permettant de simuler des v.a. de loi exponentielle ; mais il m'a semblé beaucoup plus jusicieux pédagogiquement de (re)-voir comment simuler une telle v.a. à partir d'une loi $\mathrm{Unif}^{\mathrm{me}}(0,1)$...

[¶]. En pratique, on parlera souvent de « l'estimateur $\hat{\gamma}(X)$ », la dépendance en $n$ étant implicite.

## 1.3 Simulation des variables aléatoires

La méthode de Monte-Carlo repose de manière essentielle sur la capacité à générer des réalisations (indépendantes) d'une variable aléatoire de loi donnée. Quelques rappels méritent donc d'être faits sur la génération des variables aléatoires. Dans toute cette section, nous supposerons que nous disposons d'un outil à même de générer des (réalisations de) variables aléatoires (indépendantes) uniformément distribuées sur $[0,1]$, ainsi qu'un outils à même de générer des v.a. normales standard Normale$(0,1)$.

De manière générale, lorsqu'on a affaire à une variables aléatoire "compliquées" à simuler, celle-ci peut souvent s'écrire comme une fonction de plusieurs variables aléatoires "plus simples" indépendantes. Nous ne nous intéresserons donc ici qu'aux méthodes pour simuler ces variables aléatoires plus simples.

### Méthode de la fonction quantile

**Définition (AO)** (Fonction de répartition)**.** Rappelons que la *fonction de répartition* d'une loi $P$ sur $\mathbb{R}$ est la fonction $F\colon \mathbb{R} \to [0,1]$ définie par $F(x) \coloneqq \mathbb{P}^{X\sim P}(X \leqslant x)$. ♡

**Définition (AP)** (Fonction quantile)**.** Sous les hypothèses précédentes, on appelle *fonction quantile* de $P$ la fonction de $[0,1] \to \bar{\mathbb{R}}$ définie par

$$F^{-1}(p) \coloneqq \inf\{x \in \mathbb{R} \mid F(x) \geqslant p\}. \tag{AQ}$$

♡

*Remarque* (AR)*.* On a alors, pour $P$-presque-tout $x \in \mathbb{R}$, que $F^{-1}(F(x)) = x$. En outre, si $P$ est sans atome, on a pour presque-tout $p \in [0,1]$ que $F(F^{-1}(p)) = p$, ce qui permet alors de déterminer (presque-partout) $F^{-1}(p)$ comme la solution $x$ de l'équation $F(x) = p$. ♧

! **Lemme (AS)** (Méthode de la fonction quantile)**.** *Pour $P$ une distribution de probabilité sur $\mathbb{R}$ ayant pour fonction quantile $F^{-1}$, si $U$ suit une loi uniforme sur $[0,1]$, alors $F^{-1}(U)$ suit la loi $P$.* ◇

*Remarque* (AT)*.* Il arrive que la « fonction de répartition complémentaire » $\bar{F}(x) \coloneqq \mathbb{P}^{X\sim P}(X \geqslant x)$ ait une expression plus simple que $x$. Dans la mesure où $\bar{F}$ n'est autre que la fonction de répartition de $P$ quand on a renversé l'ordre sur $\mathbb{R}$, toute cette sous-section peut être réécrite en remplaçant « $F$ » par « $\bar{F}$ », mutatis mutandis. J'appellerai alors l'analogue du lemme (AS) « méthode de la quantile complémentaire ». ♧

### Méthode de Cholesky

La méthode de Cholesky permet de simuler une variable aléatoire gausienne vectorielle (centrée) à partir de la donnée de sa matrice de covariance.

! **Lemme (AU).** *Soit $C$ une matrice (réelle) symétrique semindéfinie positive*[⟦⟧] *$n\times n$. Alors, si $L \in \mathbb{R}^{n\times p}$ est une matrice telle que $LL^\mathsf{T} = C$; si $\vec{X} \in \mathbb{R}^p$ est un vecteur*

[⟦⟧]. On rappelle qu'une matrice réelle carrée symétrique $M$ est dite « semidéfinie positive » lorsque, pour tout vecteur $\vec{v}$, on a $\vec{v}^\mathsf{T} M \vec{v} \geqslant 0$. Rappelons que le fait que les entrées de $M$ soient positives n'est ni une condition nécessaire, ni une condition suffisante pour que $M$ soit semidéfinie positive !

*gaussien standard (c.-à-d. que $X_1, X_2, \dots, X_p$ sont des v.a. normales standard i.i.d.), le vecteur $\boldsymbol{L}\vec{X} \in \mathbb{R}^n$ soit la loi* Normale$(\vec{0}, \boldsymbol{C})$. ◇

Si ce lemme est utile en pratique, c'est parce, lorsque $\boldsymbol{C}$ est *définie* positive, il est facile de trouver $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ vérifiant la condition du lemme : c'est ce qu'on appelle l'*algorithme de Cholesky* (qui retourne, en outre, une matrice triangulaire inférieure, alors déterminée de façon unique). Cet algorithme est par exemple implémenté dans la bibliothèque `numpy` de `python` : la fonction `numpy.linalg.cholesky` appliquée à une matrice réelle définie positive $\boldsymbol{C}$ renvoie l'unique matrice $\boldsymbol{L}$ de même taille et triangulaire inférieure telle que $\boldsymbol{L}\boldsymbol{L}^\mathsf{T} = \boldsymbol{C}$.

*Remarque* (AV). L'algorithme de Cholesky étant inapte à traiter des matrices positives *non définies*, on peut alors être confronté à un souci pour simuler des vecteurs gaussiens *dégénérés*, c.-à-d. dont la matrice de covariance est de rang non plein... Il est alors utile de se rappeler le lemme suivant : si $(X_1, \dots, X_n)^\mathsf{T}$ est un vecteur gaussien de matrice de covariance $\boldsymbol{C}$, alors, notant $\boldsymbol{S}$ un ensemble d'indices tels que les vecteurs-colonnes de $\boldsymbol{C}$ d'indices $j \in \boldsymbol{S}$ forment une base de $\operatorname{Im}\boldsymbol{C}$, alors le vecteur gaussien $(X_j)_{j\in\boldsymbol{S}}$ est non dégénéré ; et pour tout $j' \notin \boldsymbol{S}$, il existe des coefficients (uniquement déterminés) $(a_j)_{j\in\boldsymbol{S}}$ tels que $\sum_{j\in\boldsymbol{S}} a_j X_j = X_{j'}$, qu'on peut calculer en résolvant le système $\operatorname{Cov}(\sum_{j\in\boldsymbol{S}} a_j X_j, \vec{X}_{\boldsymbol{S}}) = \vec{0}$. ♣

## Méthode de rejet

**Lemme (AW)** (Méthode de rejet)**.** *Soit $\boldsymbol{P}$ une loi sur $\mathcal{X}$ et soit $\boldsymbol{Q}$ une loi sur $\mathcal{X}$ telle que $\boldsymbol{Q}$ soit à densité par rapport à $\boldsymbol{P}$ et que la densité $\mathrm{d}\boldsymbol{Q}/\mathrm{d}\boldsymbol{P}$ soit de la forme $x \mapsto \boldsymbol{Z}^{-1} f(x)$, pour $f(\bullet)$ une fonction connue explicitement et $\boldsymbol{Z} \coloneqq \int f(x)\boldsymbol{P}(\mathrm{d}x)$ le facteur de normalisation approprié (qu'on ne suppose pas nécessairement connu explicitement). Supposons en outre que $f(\bullet)$ soit uniformément bornée par une constante explicite $\boldsymbol{M}$. Alors, si $(X_i, U_i)_{i\in\mathbb{N}}$ est une suite de couples i.i.d. de loi $\boldsymbol{P} \otimes \mathrm{Unif}^{\mathrm{me}}(0,1)$ [**], notant*

$$i_{\star\checkmark} \coloneqq \min\{i \in \mathbb{N} \mid f(x_{i\checkmark}) \geqslant \boldsymbol{M}u_{i\checkmark}\}, \tag{AX}$$

*$x_{i_{\star\checkmark}}$ (est la réalisation d'une variable aléatoire qui) suit la loi $\boldsymbol{Q}$.* ◇

### Digression : Génération de la loi normale à partir de la loi uniforme

Dans le cadre de ce cours, nous avons supposé que, outre notre générateur de variables aléatoires de loi $\mathrm{Unif}^{\mathrm{me}}(0,1)$, nous disposions aussi d'un générateur de variables aléatoires de loi normale standard Normale$(0,1)$. Lorsque ce n'est pas le cas, il est bon de rappeler comment on peut procéder pour simuler la loi normale à partir de la loi uniforme :

**Lemme (AY)** (Méthode de Box-Muller)**.** *Si $U_0$ et $U_1$ sont des variables aléatoires indépendantes de loi* $\mathrm{Unif}^{\mathrm{me}}(0,1)$*, alors le couple de variables aléatoires $(\boldsymbol{G}_0, \boldsymbol{G}_1)$ défini par :*

$$(\boldsymbol{G}_0, \boldsymbol{G}_1) \coloneqq 2|\log U_0|\big(\cos(2\pi U_1), \sin(2\pi U_1)\big) \tag{AZ}$$

*est tel que $\boldsymbol{G}_0$ et $\boldsymbol{G}_1$ sont indépendantes et suivent chacune la loi* Normale$(0,1)$*. En particulier, $\boldsymbol{G}_0$ suit la loi normale standard.* ◇

[**]. Autrement dit, pour tout $i$, $X_i$ et $U_i$ dont indépendantes et suivent resp. les lois $\boldsymbol{P}$ et $\mathrm{Unif}^{\mathrm{me}}(0,1)$.

*Remarque* (BA). On peut aussi générer la loi normale à partir de la loi uniforme en utilisant la méthode de la fonction quantile (en particulier lorsque la fonction réciproque $\mathrm{erf}^{-1}$ de la fonction d'erreur est déjà implémentée dans le logiciel). ♧

*Remarque* (BB). En réalité, en termes de vitesse d'exécution, les deux méthodes évoquées ci-dessus (Box-Muller et fonction quantile) ne sont pas très efficaces : dans les logiciels professionnels, en général, on choisit plutôt d'implémenter la génération des lois normales par une méthode de rejet (notamment sa variante dite de la *Ziggourat*), que l'on code en outre en bas niveau pour encore plus de vitesse. ♧

## 1.4 Nombres pseudo-aléatoires

L'implémentation informatique des méthodes de Monte-Carlo faisant appel à la génération pseudo-aléatoire, il convient d'en parler un peu ici.

### Caractère *pseudo*-aléatoire des fonctions de type `random`

Dans les utilisations informatiques, les fonctions aléatoires utilisées (p. ex., en langage `python` (v3), les fonctions `random` et `gauss` du module `random`) sont généralement, en fait, des fonctions *pseudo-aléatoires* : en réalité, les calculs qu'elles produisent sont tout ce qu'il y a de plus déterministes ! L'utilisation des ces nombres comme « aléatoires » repose donc sur un "acte de foi" :

**Principe (BC).** *Nous admettrons que les nombres pseudo-aléatoires fournis par nos logiciels se comportent* comme *des réalisations de variables aléatoires indépendantes (ayant la loi qu'elles sont censées avoir).* ◇

*Remarque* (BD). « Se comporter comme des réalisations de variables aléatoires » signifie par exemple que, si $u_1, \dots, u_N$ sont censées être des réalisations de lois $\mathrm{Unif}^{\mathrm{me}}(0,1)$ indépendantes, alors pour tout $p \in [0,1]$, le nombre de $u_i$ inférieures ou égales à $p$ vaut $pN$, à quelques $N^{1/2}$ près. Plus généralement, cela signifie en substance *justement* que les calculs de Monte-Carlo donneront des résultats corrects (au risque statistique près). ♧

Cet acte de foi est-il justifié ? La réponse est « oui » ! En effet, du fait justement de l'importance des méthodes de Monte-Carlo dans la recherche appliquée et dans l'industrie, la qualité des générateurs pseudo-aléatoires a fait depuis quelques décennies l'objet d'investigations poussées. Pour les générateurs pseudo-aléatoires contemporains (par exemple le « Mersenne twister »), on sait qu'il est pratiquement impossible de mettre en défaut ces générateurs, à moins de faire des tests extraordinairement exigeants ou d'utiliser spécifiquement l'information sur la façon de fonctionner du générateur — ce qui ne correspond pas aux situations pratiques. Dans les situations que vous rencontrerez, vous pourrez ainsi considérer comme exclu que la nature *pseudo*-aléatoire du générateur vous pose souci — ce souci est bien moins à craindre, par exemple, que celui des erreurs d'arrondis... ! Si j'aborde le sujet ici, c'est surtout à cause de la question de la graine, que nous verrons un peu plus loin.

### La question de la graine

Dans le détail, tous les générateurs pseudo-aléatoires fonctionnent sur le même principe. Tout d'abord, lorsqu'il y a besoin de générer une réalisation pseudo-aléatoire de loi déterminée, le logiciel s'appuie sur une suite de bits peuso-aléatoires

i.i.d. équidistribués sur $\{0,1\}$ [††] ; et lit le nombre nécessaire de ces bits pour générer sa variable aléatoire en fonction de ceux-ci [‡‡] — cette fonction étant déterminée de sorte que, si les bits étaient *vraiment* i.i.d. uniformes, la loi de l'objet qu'on construit à partir d'eux serait vraiment celle demandée [∗].

Ces bits pseudo-aléatoires sont générés par paquets de $k$, à mesure que le logiciel en a besoin, de la façon suivante. Il existe dans les registres du logiciel une variable réservée que nous noterons ici $\mathsf{g}$, à valeurs dans un ensemble $G$ fini mais gigantesque [†], initialisée à une certaine valeur $g_0$ ; le logiciel dispose en outre d'une certain fonction judicieusement choisie $f : G \to G$ et d'une autre fonction $h : G \to H$, où $H$ désigne l'ensemble $\{0, 1, 2, \dots, 2^k - 1\}$. À chaque fois que le logiciel a besoin d'un nouveau paquet de $k$ bits aléatoires, il procède de la façon suivante :

1. $\mathsf{g}$ est remplacé par $f(\mathsf{g})$ ;
2. On calcule le nombre $h(\mathsf{g})$ ; on l'exprime en numération binaire — avec $k$ bits, quitte à ajouter des zéros non significatifs au début — ; et ces $k$ bits sont renvoyés comme étant pseudo-aléatoires.

Si le générateur pseudo-aléatoire est bien conçu, l'itération de $f$ produira un processus très irrégulier ayant la périodicité maximale $|G|$ ($G$ étant gigantesque, c'est donc comme si la suite des itérées de $f$ était apériodique) ; et les valeurs $p(\mathsf{g})$ successivement calculées seront pseudo-aléatoires i.i.d. uniformes sur $H$, ce qui correspond bien à des bits i.i.d. uniformes après conversion en binaire.

Une observation frappe alors immédiatement : si l'initialisation de la graine est identique d'une exécution à l'autre du logiciel, alors les bits pseudo-aléatoires, et donc les variables pseudo-aléatoires générées à partir de ces bits, seront exactement les mêmes à chaque exécution [‡] ! Cette situation présente à la fois un inconvénient et un avantage, énoncés respectivement par les deux points ci-dessous :

*Point* (BE) (Absence d'indépendance entre deux exécutions séparées d'un pseudo-aléa). Les variables pseudo-aléatoires obtenues lors de deux éxécutions distinctes d'un logiciel donné, ou sur deux machines différentes [§], ne sont donc pas indépendantes (en l'absence de précaution supplémentaire), ce qui interdit de distribuer

!

[††]. Du point de vue théorique, tout ce qui compte est le caractère i.i.d. et le fait que la loi soit connue. Pour ce qui est du choix d'utiliser des bits uniformes, on pourrait en revanche faire le tirage parmi un autre ensemble que $\{0,1\}$ ou prendre une distribution non équidistibuée ; quoique ce ne soit jamais le choix retenu en pratique.

[‡‡]. Soulignons ici que, même lorsque l'ordinateur prétend tirer des nombres *réels*, ceux-ci sont bien entendu *représentés* dans la machine sous une forme *tronquée* (la mémoire de l'ordinateur n'est pas infinie !) (typiquement un nombre « réel » est décrit par 64 bits), de sorte que le tirage d'un « réel » aléatoire se fait en fait au sein d'un nombre *fini* de possibilités et n'exige par conséquent de recourir qu'à un nombre fini de bits.

[∗]. Par exemple, si mon logiciel utilise une représentation binaire en virgule fixe avec 53 bits après la virgule, pour tirer une variable aléatoire de loi Uniforme$(0,1)$, le logiciel tirera 53 bits (pseudo)-aléatoires $b_1, b_2, \dots, b_{53}$ en renverra la valeur (en binaire) $0, b_1 b_2 \dots b_{k_2}$, qui suit bien (aux questions d'arrondis près) la loi uniforme sur $[0,1]$ lorsque les $b_i$ soient i.i.d. uniformes.

[†]. Par exemple, pour le « Mersenne twister », $|G| = 2^{19\,937-1}$ (et $k = 32$).

[‡]. Si la façon dont la suite de bits pseudo-aléatoires est utilisée fournir les variables « aléatoires » diffère d'une exécution à l'autre (parce qu'on n'aura pas demandé les mêmes calculs à la machine), ces variables pseudo-aléatoires ne seront alors bien entendu pas *exactement* identiques ; mais elles seront tout de même fortement corrélées, ce qui présente le même inconvénient qu'évoqué ci-dessous.

[§]. Ou même, dans le cas de deux logiciels différents mais utilisant la même générateur pseudo-aléatoire.

"naïvement" le calcul entre plusieurs exécutions du logiciel (cf. § 2.3) sous peine de résultats gravement erronés. ♧

*Point* (BF) (Reproductibilité du pseudo-aléa). Les systèmes de génération pseudo-aléatoire fournissent des résultats « aléatoires »... *complètement reproductibles* ! Par exemple, si vous prétendez que tel résultat résulte de l'exécution de tel code, une personne exécutant le même code de son côté trouvera bien le même résultat [¶] ; de sorte qu'aucune suspicion ne pourra être retenue contre vous d'avoir "truqué" votre aléa pour obtenir des résultats qui vous arrangent... Un autre exemple d'utilisation de la reproductibilité est si vous faites une simulation qui génère un flux énorme de données, trop gros pour être transmis, voire pour être stocké ; et que voulez transmettre cette simulation à quelqu'un en vue de lui en faire observer certains phénomènes : vous pourrez alors vous contenter d'envoyer votre code-source à un tiers ; il observera exactement les mêmes phénomènes ! ♧

Concernant le premier point évoqué ci-dessus, si votre générateur de nombres pseudo-aléatoires est de bonne qualité (nous admettrons que c'est le cas), une solution efficace pour obtenir deux suites de nombres pseudo-aléatoires *indépendantes* d'une exécution du logiciel à l'autre est d'imposer soi-même une graine différente d'une exécution à l'autre. On a en effet, pour un bon générateur aléatoire, le phénomène suivant :

! **Principe (BG).** *On admettra comme acte de foi que, si on réalise deux (ou plus) déroulements du générateur pseudo-aléatoires avec deux initialisations différentes de la graine, à moins de le faire vraiment (vraiment !) exprès* [‖], *les deux (ou plus) suites de bits obtenues (et donc les valeurs aléatoires données par les fonctions appelées) se comporteront comme si elles étaient* indépendantes. ◇

*Exemple* (BH). En pratique, on pourra ainsi, si on veut s'assurer que chaque exécution du programme d'une fois à l'autre est indépendante, initialiser la graine aléatoire avec la date et l'heure du jour ; ou pour distribuer le calcul entre différentes machines, initialiser la graine de la première par 1, le graine de la seconde par 2, et ainsi de suite. ♧

[¶]. Cela n'est néanmoins valable que sous réserve de compatibilité au niveau de la compilation : pour certains langages, certains codes dits « non portables » pourront donner des résultats différents lorsqu'on les compile avec des compilateurs différents ou pour des processeurs différents.

[‖]. Une telle façon de le « faire exprès » serait d'initialiser une suite avec la graine $v_0$, et l'autre suite avec la graine $f(v_0)$, auquel cas les deux suites de bits pseudo-aléatoires seraient identiques à un décalage près... Mais évidemment, $f$ est choisie en pratique de façon suffisamment subtile pour que vous n'ayez aucune chance de tomber sur un tel cas par hasard !

# Première partie

# Méthode de Monte-Carlo pour le calcul d'une espérance

# Chapitre 2

# Principe de la méthode

## 2.1 Estimation d'une espérance par la méthode de Monte-Carlo

### Estimation d'une probabilité par la méthode de rejet

Le cas le plus élémentaire de méthode de Monte-Carlo est celui où l'on souhaite estimer une certaine *probabilité*, dans un contexte où on sait simuler l'aléa. Voyons sur un exemple comment cela peut fonctionner :

*Exemple* (BI) (Le championnat de basket). Un passionné de basketball souhaite évaluer la probabilité que son équipe préférée, Nancy, gagne le championnat national pour l'année à venir. Pour ce faire, notre passionné a mis au point une modélisation détaillée du déroulement du championnat, modélisation qui lui permet de calculer exactement la probabilité du résultat de chaque match en fonction des équipes en lice. Les détails techniques figurent dans l'annexe A.1.

Bien qu'on ait une formule exacte pour le résultat de chaque match individuel, si l'on souhaite en déduire les probabilités concernant le résultat global du championnat, l'approche théorique serait passablement inextricable, en raison du très grand nombre de possibilités à prendre en compte... Que faire alors ? Notre passionné à une idée. Il se dit : « finalement ; à la base, la probabilité que Nancy gagne le championnat, c'est la proportion de fois où Nancy gagnerait le championnat si des championnats indépendants avaient lieu dans un très très grand nombre d'univers parallèles ! Or, avec ma modélisation je suis capable de *simuler* le déroulement du championnat, et même d'en simuler un très grand nombre de réalisations indépendantes ! La proportion de fois où Nancy aura gagné lors de ces simulations devrait ainsi être une valeur approchée fiable de la véritable probabilité de victoire finale ».

Ainsi, notre passionné écrit le programme suivant [*] :

```
# La fonction "basket_rejet" estime la probabilité de victoire de
# l'équipe de Nancy par la méthode de rejet.
def basket_rejet():
    # "NSIM" est le nombre de simulations à effectuer.
    NSIM = 60000
    # "NANCY" est le numéro associé à l'équipe de Nancy.
    NANCY = 2
    # "NEQ" est le nombre d'équipes en lice.
```

[*]. Pour les fonctions auxiliaires, confer le répertoire de codes informatiques associé à ce polycopié.

```
    NEQ = 16
    # "bilan" est le nombre de simulations où l'équipe de Nancy a été
    # championne.
    bilan = 0
    # On lance les simulations
    for i in range(NSIM):
        # On lance la fonction "championne" qui simule la vainqueure du
        # championnat. S'il s'agit de Nancy, on ajoute 1 à "bilan" ;
        # sinon on lui ajoute 0.
        bilan += (aux.championne() == NANCY)
    # "proportion" est la proportion de victoires de Nancy sur cet
    # ensemble de simulations.
    proportion = bilan / NSIM
    # On affiche le résultat.
    print('La probabilité estimée de victoire pour l\'équipe de Nancy est de',
          '{:.2f} %.'.format(100 * proportion))
    return
```

L'exécution lui donne :

```
>>> tic = time(); basket_rejet(); toc = time(); \
... print('Temps de calcul : {:.2f} s.'.format(toc - tic))
La probabilité estimée de victoire pour l'équipe de Nancy est de 10.90 %.
Temps de calcul : 4.38 s.
```

♧

En fait, l'idée que vient d'avoir notre passionné n'est autre que ce qu'on appelle la « méthode de rejet », qui correspond à la méthode de Monte-Carlo "naturelle" pour estimer une probabilité. Sur le plan mathématique, nous pouvons formaliser cette méthode de la façon ainsi :

**Définition (BJ).** Soit $P$ une loi de probabilité qu'on sait simuler, portant sur un certain espace $\Omega$ ; soit $A \subseteq \Omega$ tel qu'on sache déterminer efficacement, pour $\omega \in \Omega$, si on a (ou non) $\{\omega \in A\}$ ; supposons que nous intéressons à évaluer la quantité $\mathbb{P}(P \in A)$ [†] $=: \mu$. !

Dans ce cas, la *méthode de rejet* pour estimer $\mu$ consiste à réaliser un grand nombre $n$ de simulations indépendantes de la loi $P$, à compter le nombre $k$ de ces simulations qui sont tombées dans $A$, et à estimer $\mu$ par la proportion $\hat{\mu}_{\checkmark}^{(n)}$ de telles simulations :

$$\hat{\mu}_{\checkmark}^{(n)} := \frac{k}{n}. \tag{BK}$$

En termes plus formels, cela revient à introduire $n$ v.a.i.i.d. $\omega_0, \omega_1, \dots, \omega_{n-1}$, et à estimer $\mu$ par la statistique

$$\frac{\sum_{i=0}^{n-1} \mathbf{1}_{\omega_i \in A}}{n} =: \hat{\mu}, \tag{BL}$$

dont la réalisation est

$$\frac{\#\{i \in [\![0, n[\![ \mid \omega_{i\checkmark} \in A\}}{n}, \tag{BM}$$

les réalisations $\omega_{i\checkmark}$ des v.a. $\omega_i$ étant obtenues par simulation. ♡

[†]. Je rappelle que, par « $\mathbb{P}(P \in A)$ », j'entends « probabilité de l'évènement $\{\omega \in A\}$ lorsque $\omega$ est une v.a. de loi $P$ » : en l'occurrence, cela correspond simplement à la masse que donne la loi $P$ à l'ensemble $A$, i.e. à ce qu'on note habituellement « $P(A)$ ».

*Remarque* (BN). Comme suggéré par le vocabulaire ci-dessus, la méthode de Monte-Carlo par rejet peut être vue comme une *estimation statistique* : $\mu$ jouant le rôle du *paramètre caché* (qui n'est pas *réellement* caché, mais qu'on ne sait pas *calculer* numériquement) et les $\omega_i$ jouant le rôle de l'*observation*, l'« expérience » statistique consistant ici en une *simulation* ! Notez que, dans le cadre de ce cours, les notations ne distingueront pas selon qu'on parle du paramètre caché *en tant que variable aléatoire* ou *en tant que réalisation* (ce que j'aurais rendu par les notations respectives '$\mu$' et '$\mu_{\checkmark}$' le cas échéant). ♧

La version fréquentielle de la loi des grands nombres nous assure que l'estimateur de Monte-Carlo est *convergent* :

**Théorème (BO).** *(On se place ici à valeurs fixées de $P$ et $A$). Lorsque $n$ tend vers l'infini, la variable aléatoire $\hat{\mu}^{(n)}$ converge en probabilité vers la v.a. constante égale à $\mu$. La convergence est même* forte *: si on tire une infinité de v.a.i.i.d. $\omega_0, \omega_1, \ldots$, alors la suite $\hat{\mu}^{(n)}$ des estimateurs obtenus à partir de ces v.a. pour les différentes valeurs de $n$ converge presque-surement vers $\mu$ lorsque $n \to \infty$. (Et la convergence a également lieu dans $\mathrm{L}^p$ pour tout $p < \infty$).* ◇

## Estimation d'une espérance

La méthode présentée dans la définition (BJ) est en fait un cas particulier d'une méthode plus générale, qui permet d'évaluer des *espérances* :

! **Définition (BP).** Soit $P$ une loi de probabilité sur un certain espace $\Omega$, qu'on sait simuler, et soit $f : \Omega \to \mathbb{R}$ une fonction qu'on sait calculer numériquement ; supposons que la loi $f(P)$ soit de classe d'intégrabilité $\mathrm{L}^1$, et que nous nous intéressions à évaluer la quantité $\mathbb{E}(f(P))$ [‡] $=: \mu$.

Dans ce cas, la *méthode de Monte-Carlo (naturelle)* pour estimer $\mu$ consiste à réaliser un grand nombre $n$ de simulations indépendantes de la loi $P$, à calculer les valeurs de $f$ pour chacune de ces simulations, et à estimer $\mu$ par la moyenne empirique des ces valeurs : autrement dit, notant nos simulations $\omega_{0\checkmark}, \ldots, \omega_{(n-1)\checkmark}$, on approchera $\mu$ par

$$\hat{\mu}^{(n)}_{\checkmark} := \frac{1}{n} \sum_{i=0}^{n-1} f(\omega_{i\checkmark}). \tag{BQ}$$

On peut aussi voir cela en termes statistiques en disant que $\hat{\mu}^{(n)}_{\checkmark}$ est la réalisation de l'estimateur empirique (pour la fonctionnelle « espérance de la mesure-image par $f(\bullet)$ ») 

$$\hat{\mu}^{(n)} := \frac{1}{n} \sum_{i=0}^{n-1} f(\omega_i) \tag{BR}$$

associé au modèle statistique d'échantillonnage dont les observations sont les v.a.i.i.d. $\omega_i \sim P$. ♡

*Remarque* (BS). La définition ci-dessus englobe le cas particulier où $\Omega$ est $\mathbb{R}$ et où $f(\bullet)$ est l'identité. Dans ce cas, la méthode de Monte-Carlo se présente ainsi : si $Q$ est une loi sur $\mathbb{R}$ d'intégrabilité $\mathrm{L}^1$, qu'on sait simuler, l'estimateur de Monte-Carlo pour $\mathbb{E}(Q)$ consiste à considérer la moyenne empirique de simulations $x_{0\checkmark}$, $\ldots, x_{(n-1)\checkmark}$ tirées selon $Q$. ♧

[‡]. Je rappelle que, par « $\mathbb{E}(f(P))$ », j'entends « l'espérance de $f(\omega)$ pour $\omega$ une v.a. suivant la loi $P$ » : ce qui correspond, en termes plus rigoureux, à l'intégrale $\int_{\omega\in\Omega} f(\omega) P(\mathrm{d}\omega)$.

Là encore, on a convergence de l'estimateur, en vertu cette fois-ci de la loi des grands nombres :

**Théorème (BT).** *Lorsque $n$ tend vers l'infini, la variable aléatoire $\hat{\mu}^{(n)}$ converge en probabilité vers la v.a. constante égale à $\mu$ ; avec même convergence* forte *si on obtient les $\hat{\mu}^{(n)}$ à partir d'une suite infinie de v.a.i.i.d. $X_0, X_1, \ldots$. (Et la convergence a également lieu dans $\mathrm{L}^1$).* ◇

*Remarque* (BU). Attention, l'utilisation pratique de ce théorème peut être mise en défaut si les hypothèses ne sont pas réellement vérifiées : les simulations $X_i$ suivent-elles *exactement* la loi $P$? Sont-elles *vraiment* i.i.d. ? (Ou du moins, sont-elles indiscernables de v.a.i.i.d. en pratique ?)... ♧

Voyons un nouvel exemple de mise en pratique de la méthode de Monte-Carlo, illustrant cette fois-ci la définition (BP) :

*Exemple* (BV) (Le Keno). Un casino envisage de créer un jeu de machine à sous, appelé *Keno*. Dans ce jeu, le joueur doit miser sur certains numéros ; puis la machine tire au sort ses propres numéros ; et selon l'adéquation entre le pari du joueur et les numéros tirés, soit le joueur perd, soit il récupère un gain plus ou moins important (voir les détails techniques dans l'annexe A.2).

Avant que de mettre la machine en production, une question essentielle se pose : l'échelle des gains assurera-t-elle bien que le casino sera gagnant sur le long terme ; autrement dit, l'espérance de gain du joueur ne dépasse-t-elle pas sa mise ? (laquelle se trouve valoir 10 €). Or, de même que pour le championnat de basket, calculer théoriquement cette espérance serait affreusement complexe ! En revanche, on peut la simuler par la méthode de Monte-Carlo, ce qui produit le code suivant :

```
# La fonction "keno_simple" estime l'espérance de gain d'un joueur de Keno
# par la méthode de Monte-Carlo simple.
def keno_simple():
    # "NNUM" note le nombre total de numéros existant.
    NNUM = 80
    # "NCOCH" est le nombre de numéros à cocher.
    NCOCH = 20
    # "NSIM" est le nombre de simulations effectuées.
    NSIM = 1000000
    # On appelle "carton" la donnée des numéros cochés par le joueur:
    # carton[n] est le nombre de points misés sur le numéro n. Puisque la
    # loi du gain du joueur est la même quels que soient les numéros
    # cochés, on utilisera toujours le même carton, consistant à avoir
    # coché '0' pour 20 pt, '1' pour 19 pt, ..., et '19' pour 1 pt.
    carton = [NCOCH - n if n < NCOCH else 0 for n in range(NNUM)]
    # "total" est le total des résultats obtenus au cours des diverses
    # simulations.
    total = 0
    # On lance la boucle de simulations.
    for i in range(NSIM):
        # À l'aide de la fonction "tirage" pour simuler le tirage, de la
        # fonction "score" pour calculer le score à partir de "carton" et
        # "tirage", et de la fonction "gain" qui associe le gain correspondant
        # au score, on détermine la quantité d'argent gagnée par le joueur,
        # qu'on ajoute à "total".
        total += aux.gain(aux.score(carton, aux.tirage()))
    # "moyenne" est le gain moyen, qui estime l'espérance de gain.
```

```
    moyenne = total / NSIM
    # On affiche le résultat.
    print('L\'espérance de gain du Keno est estimée à ' +
          '{:.3f} euros.'.format(moyenne))
    # On quitte le programme.
    return
```

L'exécution donne :

```
>>> tic = time(); keno_simple(); toc = time(); \
... print('Temps de calcul : {:.2f} s.'.format(toc - tic))
L'espérance de gain du Keno est estimée à 9.523 euros.
Temps de calcul : 16.25 s.
```

Il semble donc que l'espérance de gain sera légèrement inférieure à 10 €, et ainsi que le jeu soit rentable. Cependant, vu que le résultat trouvé est tout de même très proche de 10 €, nous aimerions nous assurer de son niveau de précision... Cela sera vu dans la § 2.1. ♧

### Application au calcul d'intégrales

À ce stade, la méthode de Monte-Carlo pour le calcul d'une espérance semble si évidente que vous vous demandez sans doute pourquoi faire un cours dessus... Pourtant, un premier aspect conceptuel mérite déjà d'être souligné. En effet, bien que la notion d'espérance appartienne à la théorie des probabilités, la quantité $\mathbb{E}(\boldsymbol{P})$ n'en est pas moins *déterministe* : il ne s'agit pas d'une *variable aléatoire*, mais bien d'un *nombre* fixé de façon parfaitement exacte ! À l'inverse, l'estimateur $\hat{\boldsymbol{\mu}}$ que donne notre méthode, lui, n'est pas déterministe : deux simulations du même algorithme en donneront (en général) des résultats différents ! Nous somme donc bien dans le cadre évoqué en § **??**, consistant à procéder à des opérations aléatoires pour calculer une quantité déterministe : ce qui est déjà conceptuellement remarquable, vu sous cet angle ! ⟓

Essayons de mieux comprendre ce "paradoxe" pour l'exploiter dans la plénitude de sa force. Une façon de bien voir l'espérance $\mathbb{E}(\boldsymbol{P})$ comme une quantité déterministe est de se rappeler qu'il s'agit, techniquement parlant, d'une *intégrale* :

$$\mathbb{E}(\boldsymbol{P}) \coloneqq \int_{x\in\mathbb{R}} x\boldsymbol{P}(\mathrm{d}x), \tag{BW}$$

ou encore, si on s'intéresse à l'espérance d'une loi de la forme $\gamma(\boldsymbol{Q})$ :

$$\mathbb{E}(\gamma(\boldsymbol{Q})) = \int_{\omega\in\Omega} \gamma(\omega)\boldsymbol{Q}(\mathrm{d}\omega). \tag{BX}$$

Cela pousse alors à se demander si la méthode de Monte-Carlo ne pourrait pas être aussi utilisée pour estimer des intégrales "quelconques" ne s'écrivant pas naturellement sous la forme d'espérances, par exemple pour calculer l'intégrale, par rapport à la mesure de Lebesgue, d'une fonction ($\mathrm{L}^1$) de $\mathbb{R}^d$ dans $\mathbb{R}$.

Et de fait, c'est le cas : on peut effectivement utiliser la méthode de Monte-Carlo pour le calcul d'un espérance pour estimer des intégrales ne s'écrivant pas "naturellement" sous forme d'espérances ! C'est l'objet de cette sous-section. Le résultat sur lequel nous allons nous appuyer est le suivant :

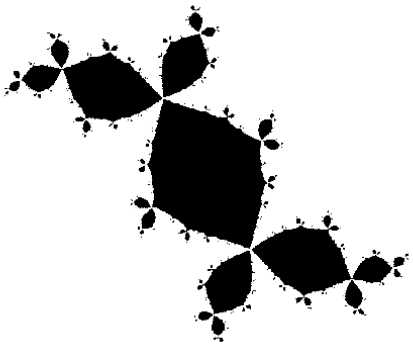

FIGURE 2.1 – Le lapin de Douady.

! **Proposition (BY).** *Pour $f$ une fonction intégrable sur $\mathbb{R}^d$, et $\boldsymbol{P}$ une mesure de probabilité sur $\mathbb{R}^d$ ayant par rapport à la mesure de Lebesgue une densité $\mathrm{d}\boldsymbol{P}/\mathrm{dvol}_d$ qui est non nulle partout où $f$ est non nulle, on a :*

$$\int_{\mathbb{R}^d} f(x)\,\mathrm{vol}_d(\mathrm{d}x) = \mathbb{E}^{x\sim P}\big(f(x)\,/\,\big(\mathrm{d}\boldsymbol{P}/\mathrm{dvol}_d\big)(x)\big). \tag{BZ}$$

*Cela permet alors d'utiliser l'estimateur de Monte-Carlo (BQ) pour évaluer $\int f(x)\mathrm{d}x$.* ♢

*Démonstration.* Notons $\boldsymbol{\Omega}$ l'ensemble des points où $\mathrm{d}\boldsymbol{P}/\mathrm{dvol}_d$ est non nulle ; la mesure de Lebesgue a alors sur $\boldsymbol{\Omega}$ une densité par rapport à $\boldsymbol{P}$ qui est $1/(\mathrm{d}\boldsymbol{P}/\mathrm{dvol}_d)$. D'autre part, puisque $f$ est nulle en-dehors de $\boldsymbol{\Omega}$, on a $\int_{\mathbb{R}^d} f(x)\,\mathrm{vol}_d(\mathrm{d}x) = \int_{\boldsymbol{\Omega}} f(x)\,\mathrm{vol}_d(\mathrm{d}x)$, d'où $\int_{\mathbb{R}^d} f(x)\,\mathrm{vol}_d(\mathrm{d}x) = \int_{\boldsymbol{\Omega}} f(x)\big(\mathrm{dvol}_d/\mathrm{d}\boldsymbol{P}\big)(x)\boldsymbol{P}(\mathrm{d}x) = \int_{\boldsymbol{\Omega}} f(x)\,/\,\big(\mathrm{d}\boldsymbol{P}/\mathrm{dvol}_d\big)(x)^{-1}\cdot\boldsymbol{P}(\mathrm{d}x) = \mathbb{E}^{x\sim P}\big(f(x)\,/\,\big(\mathrm{d}\boldsymbol{P}/\mathrm{dvol}_d\big)(x)\big)$, CQFD. ♤

*Remarque* (CA). Il reste néanmoins indispensable, pour appliquer la méthode de Monte-Carlo, de se ramener *in fine* à une espérance... ! Compte tenu de cette observation, à partir du chapitre suivant, en ce qui concerne la partie « méthode de Monte-Carlo pour le calcul d'une espérance », nous traiterons uniquement l'évaluation par la méthode de Monte-Carlo de quantités écrites *directement* sous la forme d'une espérance. Mais en fait, cela englobera aussi le cas des intégrales "ordinaires", que nous pourrons ré-écrire sous la forme d'espérances à l'aide de la technique exposée dans la proposition ci-dessus. ♧

*Remarque* (CB). Il y a très nombreuses possibilités pour choisir la loi $\boldsymbol{P}$ destinée à appliquer la proposition (BY) : en fonction du choix fait, on aboutira à autant d'estimateurs de Monte-Carlo *différents* pour la *même* intégrale, et qui auront en général des *performances* elles aussi différentes ! Nous développerons cette idée plus en détail dans la § 4.1 consacrée à l'« échantillonnage préférentiel », où nous verrons que l'argument peut même être appliqué à des intégrales qui s'écrivent "naturellement" sous la forme d'une espérance ! ♧

Nous allons maintenant voir un exemple d'utilisation de la méthode exposée dans la proposition (BY), qui nous donnera aussi l'occasion de présenter une méthode de choix particulièrement classique pour la loi auxiliaire $\boldsymbol{P}$.

*Exemple* (CC) (Le lapin de Douady). Le lapin de Douady est une figure du plan dont vous pouvez observer l'allure sur la figure 2.1. La définition de cette figure rend algorithmiquement facile de savoir si un point donné du plan appartient ou

non au lapin de Douady (voir l'annexe A.3 pour les détails) ; en revanche, le bord du lapin de Douady est très irrégulier [§], ce qui rend son aire délicate à évaluer : on pourrait certes considérer un maillage régulier de points et compter combien de ces points tombent dans le lapin ; mais qu'un tel maillage régulier serait-il pertinent ici... ? C'est pourquoi ici nous allons plutôt procéder par méthode de Monte-Carlo.

L'aire de cette figure est définie comme $\int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty}\mathbf{1}_{x+yi\in lapin}\,\mathrm{d}x\mathrm{d}y$. À priori, il s'agit donc d'une intégrale "générale" ; néanmoins, nous allons utiliser la technique de la proposition **(BY)** pour ré-écrire cette intégrale comme une espérance ! Pour ce faire, on utilise le fait [confer § A.3] que le lapin de Douady est complètement contenu dans le carré $[-2,2]\times[-2,2]$ (dont seul l'intérieur importe concernant l'aire). Si nous introduisons la probabilité uniforme sur le carré $]-2,2[\times]-2,2[$, notée $\boldsymbol{P}$, nous pouvons donc réécrire l'aire du lapin ainsi :

$$\begin{aligned}\int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty}\mathbf{1}_{x+yi\in lapin}\,\mathrm{d}x\mathrm{d}y &= \int_{x\in]-2,2[}\int_{y\in]-2,2[}\mathbf{1}_{x+yi\in lapin}\ \mathrm{vol}_2(\mathrm{d}x\times\mathrm{d}y)\\ &= \int_{(x,y)\in]-2,2[\times]-2,2[}\mathbf{1}_{x+yi\in lapin}\cdot 16\boldsymbol{P}(\mathrm{d}x\times\mathrm{d}y)\\ &= \mathbb{E}^{(x,y)\sim\mathrm{Unif}^{\mathrm{me}}(]-2,2[\times]-2,2[)}(16\cdot\mathbf{1}_{x+yi\in lapin}). \quad (\texttt{CD})\end{aligned}$$

Le choix de prendre une loi de référence uniforme sur un rectangle parallèle aux axes permet de simuler facilement celle-ci : en effet, dire que $(\boldsymbol{X},\boldsymbol{Y})$ est uniforme sur $]-2,2[\times]-2,2[$, c'est dire que $\boldsymbol{X}$ et $\boldsymbol{Y}$ sont indépendantes et chacune uniforme sur $]-2,2[$ ; et il est facile de simuler une v.a. uniforme sur $]-2,2[$ en observant que $\mathrm{Unif}^{\mathrm{me}}(-2,2) = 4\,\mathrm{Unif}^{\mathrm{me}}(0,1) - 2$.

Cela conduit au programme suivant pour estimer l'aire :

```
from random import *
from math import *
import time
import aux_lapin as aux

# La fonction "volLapin" estime l'aire du lapin de Douady par la méthode de
# Monte-Carlo, en échantillonnant selon la loi uniforme sur le carré
# [-2,2]×[-2,2].
def volLapin():
    # "NSIM" est le nombre de simulations.
    NSIM = 15000000
    # Les coordonnées du carré sur lequel on effectue l'échantillonnage sont
    # notées [XMIN, XMAX] × [YMIN, YMAX]
    ((XMIN, XMAX), (YMIN, YMAX)) = ((-2.0, 2.0), (-2.0, 2.0))
    #"somme" est la somme des simulations effectuées.
    somme = 0
    # Boucle de Monte-Carlo.
    for i in range(NSIM):
        # On tire z uniformément sur le carré [XMIN,XMAX]×[YMIN,YMAX], vu comme
        # un nombre complexe.
        z = ((XMIN + random() * (XMAX - XMIN)) +
             (YMIN + random() * (YMAX - YMIN)) * 1j)
        # Calcule la fonction dont on prendra l'espérance. La fonction "enlapin"
```

[§]. Ce bord est même tellement irrégulier qu'on dit qu'il s'agit d'un objet « fractal », avec une dimension intermédiaire entre 1 et 2 !

```
        # renvoie True si on est dans le lapin, et False sinon.
        somme += aux.enlapin(z) * (XMAX - XMIN) * (YMAX - YMIN)
    # "moyenne" est la moyenne des simulations effectuées.
    moyenne = somme / NSIM
    # Affichage du résultat.
    print('L\'aire du lapin de Douady est estimée à {:f}.'.format(moyenne))
    return
```

J'ai obtenu à l'exécution :

```
>>> lapin()
L'aire du lapin de Douady est estimée à 1.306582.
```

♧

La procédure utilisée dans cet exemple, où la probabilité auxiliaire que nous avons introduite était uniforme sur un rectangle, constitue la façon la plus "standard" d'utiliser la méthode de Monte-Carlo pour calculer une intégrale :

*Point* (CE). Lorsqu'on doit intégrer une fonction $f$ sur $\mathbb{R}^d$ à support compact, la façon la plus simple d'utiliser la méthode de Monte-Carlo consiste à trouver un rectangle parallèle aux axes contenant le support de $f$, et à appliquer la proposition (BY) en prenant pour loi $\boldsymbol{P}$ la distribution uniforme sur ce rectangle.

Concernant le choix du rectangle parmi tous les rectangles contenant le support de $f$, il est préférable de le prendre aussi petit que possible, pour des raisons qui apparaitront plus loin. ♧

Pour utile qu'il soit, le point ci-dessus ne permet néanmoins pas de traiter tous les problèmes d'intégration par Monte-Carlo, puisqu'il existe des cas où la fonction qu'on souhaite intégrer n'est pas à support compact. L'exemple ci-dessous illustre comment on peut procéder dans ce genre de cas.

*Exemple* (CF) (SOS analyse complexe). Un ingénieur a été amené à calculer à la main la valeur exacte de l'intégrale

$$I \coloneqq \int_{-\infty}^{\infty} \frac{x^2+1}{x^2+x+1}\left(\frac{\sin x}{x}\right)^2 \mathrm{d}x. \tag{CG}$$

Il a trouvé un résultat de

$$\pi + \frac{\pi}{2\sqrt{3}} - \frac{\pi \mathrm{e}^{-\sqrt{3}}\cos(1)}{4\sqrt{3}} - \frac{\pi \mathrm{e}^{-\sqrt{3}}\sin(1)}{4} \simeq 3{,}888\,221; \tag{CH}$$

cependant il a un doute, le calcul étant fort compliqué... Il demande alors à petite sœur en master de mathématiques de faire le calcul de son côté, et celle-ci trouve quant à elle un résultat de

$$\pi + \frac{\pi}{2\sqrt{3}} - \frac{\pi \mathrm{e}^{-\sqrt{3}}\cos(1)}{2\sqrt{3}} - \frac{\pi \mathrm{e}^{-\sqrt{3}}\sin(1)}{2} \simeq 3{,}727\,950. \tag{CI}$$

Pour trancher entre les deux options, notre ingénieur souhaite évaluer l'intégrale numériquement. Il pourrait certes procéder par méthodes des rectangles ; mais comme il ne sait pas trop comment s'y prendre pour gérer au mieux la discrétisation et la troncature, il décide de passer plutôt par la méthode de Monte-Carlo.

Dans un premier temps, il faut choisir une loi sur $\mathbb{R}$ ayant une densité non nulle partout où l'intégrande est non nulle — c.-à-d. en l'occurrence sur tout $\mathbb{R}$ (modulo négligeable) —, qui soit facile à simuler et dont la densité soit facile à calculer. Notre ingénieur choisit la loi $\boldsymbol{P}$ de densité

$$\boldsymbol{P}(\mathrm{d}x) = \frac{1}{2(1+|x|)^2}\,\mathrm{vol}(\mathrm{d}x), \tag{CJ}$$

dont la méthode de simulation est expliquée dans l'annexe A.4.

On peut alors écrire

$$\boldsymbol{I} = \mathbb{E}^{x\sim\boldsymbol{P}}\left(2(1+|x|)^2\frac{x^2+1}{x^2+x+1}\left(\frac{\sin x}{x}\right)^2\right), \tag{CK}$$

ce qui permet d'utiliser la méthode de Monte-Carlo, avec le programme suivant (voir les fonctions auxiliaires en annexe) :

```
from random import *
from math import *
import time
import aux_SOS as aux

# La fonction 'SOS_analyse' estime l'intégrale I par la méthode de Monte-Carlo,
# en utilisant la loi d'échantillonnage P suggérée par le polycopié.
def SOS_analyse():
    # 'NSIM' est le nombre de simulations effectuées.
    NSIM = 4000000
    # 'total' sera le total des valeurs simulées de y.
    total = 0
    for i in range(NSIM):
        # 'x' est une réalisation de la loi P.
        x = aux.echantillonnage()
        # 'y' est une réalisation de la quantité dont on calcule l'espérance
        # sous la loi P.
        y = aux.integrande(x) / aux.densite(x)
        total += y
    moyenne = total / NSIM
    print('L\'intégrale I est estimée à {:f}.'.format(moyenne))


# La fonction 'SOS_IC' est une adaptation de la fonction 'SOS' où on renvoie un
# intervalle de confiance (à 95 %) plutôt qu'une estimation. Le code se passe
# de commentaires.
def SOS_IC():
    NSIM = 4000000
    total = 0.0
    total2 = 0.0
    for i in range(NSIM):
        x = aux.echantillonnage()
        y = aux.integrande(x) / aux.densite(x)
        total += y
        total2 += y ** 2
    moyenne = total / NSIM
    variance = total2 / NSIM - moyenne ** 2
    variance_estr = variance / NSIM
```

```
    ectype_estr = sqrt(variance_estr)
    print('Au risque 5 %, l\'intégrale I est située dans l\'intervalle ' +
          '[{:f}, {:f}].'.format(moyenne - 1.96 * ectype_estr,
                                 moyenne + 1.96 * ectype_estr))
```

À l'exécution :

```
>>> tic = time(); SOS_analyse(); toc = time(); \
... print('Temps de calcul : {:.2f} s.'.format(toc - tic))
L'intégrale I est estimée à 3.728854.
Temps de calcul : 4.08 s.
```

Les deux premières décimales coïncident avec la valeur prévue par la petite sœur (et la troisième n'en diffère que d'une unité) ; il y a donc tout lieu de croire que c'est elle qui a raison. Cette conviction serait encore renforcée par le calcul de l'intervalle de confiance associé à l'estimateur de $I$ [confer § 2.1 ci-après], dont on verrait qu'il contient bien la valeur de la petite sœur et pas celle de l'ingénieur. ♧

## Intervalles de confiance

Nous savons que l'estimateur de Monte-Carlo est convergent (théorème (BT)), ce qui est un bon point ; mais en l'absence d'information sur la vitesse de sa convergence, l'interprétation de l'estimation demeure fort nébuleuse... Il est donc important d'être capable d'indiquer quelle est sa *précision* de notre estimation de $\mu$.

Bien entendu, le caractère aléatoire de l'etimateur de Monte-Carlo rend impossible d'obtenir une certitude *absolue* sur l'erreur commise : par exemple, si on cherche à évaluer la probabilité qu'une pièce non truquée tombe sur "pile", il est *possible* (quoique déliramment improbable) que tous les lancers faits dans le cadre de notre expérience donnent "face", et donc qu'on évalue à tort la probabilité de "pile" comme étant nulle ! Nous pouvons néanmoins tenter de donner un *intervalle de confiance* pour la quantité estimée, i.e. un intervalle (déterminé à partir des simulations) qui contiendra $\mu$ avec une très grande probabilité, fixée à l'avance (p. ex. 95 %). C'est à la construction de tels intervalles de confiance que s'intéresse cette section.

Le théorème central pour obtenir nos intervalles de confiance est le suivant :

**Théorème (CL).** *Soit, comme dans le théorème (BY), $P$ une loi de probabilité sur un ensemble $\Omega$, qu'on sait simuler, et $f : \Omega \to \mathbb{R}$ une fonction qu'on sait calculer. Supposons en outre que $f(P)$ soit de classe d'intégrabilité $L^2$, et non constante. Pour $n > 1$, soient $\omega_0, \dots, \omega_{n-1}$ des v.a.i.i.d. de loi $P$ ; notons comme précédemment $\hat{\mu}^{(n)}$ l'espérance empirique des $f(\omega_i)$ ; et notons $\hat{\sigma}^{(n)}$ leur écart-type empirique, dont la réalisation est*

$$\hat{\sigma}_{\checkmark} := \left( \frac{1}{n} \sum_{i=0}^{n-1} (f(\omega_{i\checkmark}) - \hat{\mu}_{\checkmark})^2 \right)^{1/2}. \tag{CM}$$

*Alors, lorsque $n$ tend vers l'infini, on a*

$$\frac{\hat{\mu}^{(n)} - \mu}{\hat{\sigma} n^{-1/2}} \xrightarrow{\text{loi}} \text{Normale}(0, 1). \tag{CN}$$

◇

!

*Démonstration.* Notant $\sigma \coloneqq \mathrm{Var}^{1/2}(f(P))$, nous savons, en vertu du théorème-limite central, que la loi de $(\hat{\mu}^{(n)} - \mu) / \sigma n^{-1/2}$ converge vers la loi normale standard. Néanmoins, dans l'équation (CN), ce n'est pas $\sigma$ lui-même qui intervient, mais $\hat{\sigma}^{(n)}$... Néanmoins, si nous arrivons à montrer que $\hat{\sigma}^{(n)}$ est un estimateur convergent de $\sigma$, alors le théorème sera prouvé, en vertu de l'argument suivant : si $(X_n)_{n\in\mathbb{N}}$ et $(Y_n)_{n\in\mathbb{N}}$ sont des suites de v.a. sur des espaces respectifs $E$ et $F$, telles que $\mathrm{Loi}(X_n)$ converge vers une certaine loi $P_\infty$ et que $\mathrm{Loi}(Y_n)$ converge vers une certaine loi *de Dirac* $\delta_{y_\infty}$, et si $\varphi : E \times F \to G$ est une fonction continue (au moins sur un voisinage de $E \times \{y_\infty\}$), alors la loi de $\varphi(X_n, Y_n)$ converge vers la loi $\varphi(P_\infty, y_\infty)$ [¶].

Pour finir la preuve, il reste à montrer que $\hat{\sigma}^{(n)}$ converge vers $\sigma$. Nous omettrons ici les exposants '$(n)$'. D'après les propriétés de la variance, on a $\hat{\sigma} = (\hat{\mu}_2 - \hat{\mu}^2)^{1/2}$, où $\hat{\mu}_2$ désigne le second moment empirique des $f(\omega_i)$, i.e. $\hat{\mu}_{2\checkmark} \coloneqq n^{-1} \sum_{i=0}^{n-1} f(\omega_{i\checkmark})^2$. Nous savons déjà que $\hat{\mu}$ est un estimateur (fortement) convergent de $\mathbb{E}(f(P))$ ; et d'après la loi des grands nombres appliquée à la loi $f(P)^2$ (qui est $\mathrm{L}^1$ puisque $f(P)$ est $\mathrm{L}^2$), $\hat{\mu}_2$ est également un estimateur (fortement) convergent de $\mathbb{E}(f(P)^2)$ : dès lors, par continuité de la fonction $(x, y) \mapsto (y - x^2)^{1/2}$, $\hat{\sigma}$ converge (fortement) vers $\big(\mathbb{E}(f(P)^2) - \mathbb{E}(f(P))^2\big)^{1/2} = \sigma$. ♤

Le théorème (CL) sera principalement utilisé via le corolaire suivant, qui fournit un intervalle de confiance pour les estimateurs de Monte-Carlo :

! **Corolaire (CO).** *Reprenons les hypothèses et les notations du théorème (CL) ; et soient* $z \in ]0, \infty[$ *et* $\alpha \in ]0, 1[$ *liés par la relation :* $\mathbb{P}\big(|\mathrm{Normale}(0,1)| > z\big) = \alpha$ [‖]. *Alors l'intervalle*

$$[\hat{\mu}^{(n)} \pm z\hat{\sigma}^{(n)} / n^{1/2}] \tag{CP}$$

*est un intervalle de confiance asymptotique (lorsque* $n \to \infty$*), au niveau de risque* $\alpha$*, pour la quantité d'intérêt* $\mu$*.* ♢

Avant l'illustrer les théorèmes (CL) et (CO) sur des exemples, voici quelques remarques auquel il convient de prendre garde en pratique :

*Remarque* (CQ). Pour que le théorème (CL) soit valide, il est indispensable que $f(P)$ soit de classe $\mathrm{L}^2$. Si $f(P)$ est $\mathrm{L}^1$ mais pas $\mathrm{L}^2$, alors l'estimateur de Monte-Carlo convergera certes ; mais l'intervalle de confiance associé ne sera pas valable !

Or, en pratique, il est souvent compliqué de prouver que $f(P)$ est effectivement $\mathrm{L}^2$ ; et pour ne rien arranger, il existe aussi des cas où on a certes intégrabilité $\mathrm{L}^2$, mais que la convergence asymptotique est tellement lente que l'intervalle de confiance est inutilisable en pratique... (confer remarque **??** *infra*). Du point de vue des applications pratiques, on ne cherchera donc pas (sauf exception) à prouver *théoriquement* le caractère $\mathrm{L}^2$, mais plutôt à vérifier *empiriquement* que les résultats obtenus *semblent* cohérents avec ce qu'on devrait observer dans le cas $\mathrm{L}^2$ (confer remarque **??** ci-après).

Dans la plupart de nos exercices, afin de mieux nous focaliser sur les nouveaux concepts abordés, nous admettrons implicitement qu'on est dans un cas où on a bien un comportement $\mathrm{L}^2$ et une convergence suffisamment rapide ; mais en situation réelle, il conviendrait de toujours s'assurer que c'est bien le cas ! (ou du moins qu'il n'y a pas d'indices du contraire). ♧

[¶]. Ici le rôle de $X_n$ est joué par $(\hat{\mu}^{(n)} - \mu) / \sigma n^{-1/2}$ ; le rôle de $Y_n$ est joué par $\hat{\sigma}^{(n)}$ ; le rôle de $P_\infty$ est joué par $\mathrm{Normale}(0, 1)$ ; le rôle de $y_\infty$ est joué par $\sigma$, et le rôle de $\varphi$ est joué par la fonction $(x, y) \mapsto \sigma x / y$.

[‖]. Autrement dit, si on nous donne $z$, on en déduit $\alpha$ comme valant $2\Phi(-z)$, où $\Phi(\bullet)$ désigne la fonction de répartition de la loi normale standard ; et si on nous donne $\alpha$, on en déduit $z$ comme valant $\Phi^{-1}(1 - \alpha/2)$, où $\Phi^{-1}(\bullet)$ est la fonction de quantile de la loi normale standard.

*Remarque* (CR). Si on souhaite contrôler *empiriquement* lu caracètre $\mathrm{L}^2$ de la loi $f(\boldsymbol{P})$, le mieux est d'utiliser le critère suivant : l'estimateur $\hat{\boldsymbol{\sigma}}^{(n)}$ de $\mathrm{Var}^{1/2}(f(\boldsymbol{P}))$ est convergent si et seulement si $f(\boldsymbol{P})$ est $\mathrm{L}^2$. Concrètement, cela se traduit ainsi : si on procède à trois ou quatre estimations indépendantes de $\mathrm{Var}^{1/2}(f(\boldsymbol{P}))$ (par exemple en utilisant les estimations déduites respectivement des quatre quarts successifs de nos simulations), on doit obtenir des valeurs très proches les unes des autres (en valeur relative) si on a effectivement intégrabilité $\mathrm{L}^2$, alors que ces valeurs seront (en général) sensiblement différentes entre elles lorsque l'intégrabilité n'est pas satisfaite. (Le phénomène peut aussi se produire en cas de convergence trop lente vers l'asymptotique, auquel cas la conclusion est la même : l'intervalle de confiance n'est pas fiable !). ♣

*Remarque* (CS). Par ailleurs, il faut bien garder à l'esprit que, même dans le cas $\mathrm{L}^2$, nos intervalles de confiance ne sont qu'asymptotiques : de sorte que, si on est trop loin du régime de convergence, les intervalles de confiance peuvent nous induire en erreur... La façon de gérer ce souci sera la même, *mutatis mutandis*, que pour l'intégrabilité $\mathrm{L}^2$ :

- Il peut y avoir *quelques* cas théoriques où on a des garanties explicites sur la vitesse de convergence du risque associé à l'intervalle de confiance ; mais *en pratique*, on est généralement obligé de se cantonner à un contrôle *empirique* de conformité (apparente) entre nos résultats et l'hypothèse que la convergence soit suffisamment rapide.
- Dans la plupart de nos exercices, on *admettra* que la convergence a lieu suffisamment vite dans le cas qui nous intéresse, afin de mieux nous focaliser sur les points nouveaux ; mais en situation industrielle, il conviendra de toujours vérifier qu'il semble bien y avoir convergence pour le nombre de simulations considéré !
- La procédure évoquée en remarque (CR), consistant à remarquer si l'estimateur $\hat{\boldsymbol{\sigma}}$ semble stable lorsqu'on le ré-évalue de façon indépendante, détecte en fait aussi bien les soucis de convergence que les défauts d'intégrabilité [**].

♣

*Remarque* (CT). Bien qu'il n'existe pas de critère décisif pour savoir si la convergence du niveau de risque asymptotique est atteinte, *en pratique*, il peut être utile de retenir le critère suivant :

- Si on souhaite obtenir un intervalle de confiance au risque $\boldsymbol{\alpha}$, et que le nombre $\boldsymbol{n}$ de simulations effectuées est inférieur à $\boldsymbol{\alpha}^{-2}$, alors ce n'est pas suffisant pour qu'on puisse espérer que la niveau de confiance asymptotique soit atteint ;
- Réciproquement, "dans la plupart des cas" (à savoir, quand on n'a pas de souci de type « queues lourdes »), si le nombre $\boldsymbol{n}$ de simulations utilisées est sensiblement supérieur à $\boldsymbol{\alpha}^{-2}$, alors cela suffit pour que le niveau réel de l'intervalle de confiance atteigne à peu près sa valeur asymptotique.

On notera en particulier que, pour obtenir un intervalle de confiance qui soit effectivement pertinent à un niveau de risque très faible (p.ex. $10^{-4}$), il convient de faire un nombre de simulations très élevé (en l'occurrence, au moins de l'ordre de $10^8$) ! ♣

Ces remarques étant posées, voyons comment elles peuvent améliorer utilement nos exemples resp. du Keno et de l'analyse complexe.

[**]. De ce fait, lorsque l'estimateur est instable, on ne peut pas savoir si cela est dû à un souci de convergence ou à un défaut d'intégrabilité !

*Exemple* (CU) (Le Keno, bis). Reprenons donc le problème du Keno que nous avons traité tout-à-l'heure (exemple (BV)). Nous avons vu précédemment que, pour une mise dépensée de 10 €, les gains moyens du joueur (récupération de la mise incluse) étaient d'environ 9,50 €, ce qui semble, à vue de nez, permettre au casino de faire un bénéfice. Cependant, quelle est la *précision* sur cette estimation de 9,50 € ? En effet, cette valeur étant proche du montant de la mise, il suffirait qu'il y ait 0,80 € d'erreur pour que le gain moyen du joueur soit en fait de 10,30 €, et que le casino se retrouve à perdre de l'argent... Il y a donc besoin de reprendre le programme précédent pour quantifier la précision de l'estimateur. L'enjeu financier d'une évaluation trop basse étant potentiellement très important, les dirigeants exigent que le risque de se tromper dans ce sens-là ne soit que de 1 ‰ ; dans le sens d'une estimation trop élevée, en revanchen, on se contentera d'un risque classique de 5 %.

Le programme est alors le suivant (avec les mêmes fonctions auxiliaires que précédemment) :

```
# La fonction "keno_IC" estime l'espérance de gain d'un joueur de Keno par la
# méthode de Monte-Carlo, en donnant un intervalle de confiance. Nous ne
# commentons que les différences par rapport à "keno_simple"
def keno_IC():
    NNUM = 80
    NCOCH = 20
    NSIM = 1000000
    total = 0
    # Pour calculer l'intervalle de confiance on a aussi besoin de calculer le
    # total des carrés des quantités simulées.
    total_carres = 0
    carton = [0 for n in range(NNUM)]
    for n in range(NCOCH):
        carton[n] = NCOCH - n
    for i in range(NSIM):
        # "legain" est le gain obtenu par le joueur. Comme on va l'utiliser deux
        # fois, on doit le stocker dans une variable d'abord.
        legain = aux.gain(aux.score(carton, aux.tirage ()))
        total += legain
        # On met aussi à jour "total_carres".
        total_carres += legain * legain
    moyenne = total / NSIM
    # On détermine aussi la variance (estimée) du gain.
    variance_gain = total_carres / NSIM - moyenne * moyenne
    # On en déduit l'écart-type (estimé) de la moyenne empirique
    ect_estr = sqrt (variance_gain / NSIM)
    # On calcule les coefficients pour l'intervalle de confiance --- noter qu'on
    # peut aussi faire cette étape au préalable et juste entrer les résultats
    # numériquement; en l'occurrence c0 vaut -1.65 et c1 vaut +3.10. Remarque:
    # on peut aussi utiliser la méthode norm de scipy.stats et demander « c0 =
    # norm.ppf(.05, 0, 1); c1 = norm.ppf(.999, 0, 1) » : cette méthode-là ayant
    # l'avantage d'être plus lisible, mais moins portable.
    from scipy.special import erfinv
    c0 = sqrt(2) * erfinv(-.90)
    c1 = sqrt(2) * erfinv(+.998)
    # On affiche le résultat.
    print('Avec un risque de 0,1 % pour la sous-estimation,',
          'resp. de 5 % pour la surestimation,',
          'l\'espérance de gain du Keno se situe dans l\'intervalle',
```

```
            '[{:.3f}, {:.3f}] EUR.'.format(moyenne + c0 * ect_estr,
                                           moyenne + c1 * ect_estr))
    return
```

J'ai obtenu le résultat suivant [††] :

```
>>> tic = time(); keno_IC(); toc = time(); \
... print('Temps de calcul : {:.2f} s.'.format(toc - tic))
Avec un risque de 1 % pour la sous-estimation, resp. de 5 % pour la
surestimation, l'espérance de gain du Keno se situe dans l'intervalle [9.376,
9.800] €.
Temps de calcul : 15.87 s.
```

Ouf, tout va bien : on est quasiment certain que l'espérance de gain est effectivement en-dessous de 10 € ! Mais on voit que l'incertitude sur la gain moyen était loin d'être négligeable, et que nous avons donc bien fait de vérifier... ♧

*Exemple* (CV). On peut adapter le code de l'exemple (CF) sur le calcul d'intégrale pour calculer aussi l'intervalle de confiance : voir le code correspondant sur dans le répertoire de codes. On obtient alors le résultat suivant :

```
>>> tic = process_time(); SOS_IC(); toc = process_time(); \
... print('Temps de calcul : {:.2f} s.'.format(toc - tic))
Au risque 5 %, l'intégrale I est située dans l'intervalle [3.725874, 3.731835].
Temps de calcul : 4.28 s.
```

Cela nous confirme que l'estimation numérique de $I$ est incompatible avec le calcul la petite sœur, et par ailleurs incompatible avec celui de l'ingénieur : tout porte donc à croire que c'est bien la première qui a raison ! ♧

## 2.2 Fonction d'une espérance

### Estimateur de la fonction d'une espérance

Il arrive que l'on cherche à estimer une quantité qui n'est pas directement une espérance, mais qui est néanmoins une *fonction* d'espérance. Par exemple, si nous reprenons l'exemple du championnat de basket, on pourrait s'intéresser à estimer, non pas la *probabilité* de victoire de l'équipe de Nancy, mais la juste *cote* de l'équipe, c'est-à-dire la quantité d'argent que devrait gagner un parieur ayant misé une unité pour que le pari soit équilibré. Concrètement, si $p$ est la probabilité de victoire de Nancy, la juste cote (pour un pari à « tant contre 1 ») sera $(1-p)/p =: c$. Or, contrairement à $p$, la cote $c$ ne s'exprime pas sous forme d'espérance... Mais cela n'est pas grave : en effet, puisque nous disposons d'un estimateur (fortement) convergent $\hat{p}$ pour $p$ et que $c$ s'exprime comme une fonction continue de $p$, on en déduit que l'estimateur par substitution $\hat{c} := (1-\hat{p})/\hat{p}$ convergera vers de $c$ !

Le raisonnement s'applique aussi bien pour une fonction de plusieurs espérances : si le pari concerne la probabilité de victoire de Nancy contre celle de l'équipe #1 (le pari étant annulé si c'est une autre équipe qui gagne), notant $q$ la probabilité de victoire de l'équipe #1, la juste cote est alors de $(q/p)$ ; et notant $\hat{q}$ l'estimateur de Monte-Carlo de $q$, $(\hat{q}/\hat{p})$ est alors un estimateur (fortement) convergent de $(q/p)$.

Nous pouvons formaliser ce qui précède par le théorème suivant :

[††]. Ici j'ai initialisé le générateur aléatoire avec la même graine que dans la version keno_simple, pour qu'on fasse exactement les mêmes simulations.

! **Théorème (CW).** *Soit $P$ une loi de probabilité sur un espace $\Omega$, qu'on sache simuler, et soient $f_0(\bullet), f_1(\bullet), \dots, f_{K-1}(\bullet)$ des fonctions de $\Omega$ dans $\mathbb{R}$, qu'on sache calculer numériquement, telles que les $f_k(P)$ soient toutes de classe $\mathrm{L}^1$. Notons $\mathbb{E}(f_0(P)) =: \mu_0, \dots, \mathbb{E}(f_{K-1}(P)) =: \mu_{K-1}$ ; et supposons que notre objectif soit d'estimer $\gamma(\mu_0, \dots, \mu_{K-1})$, pour $\gamma : \mathbb{R}^k \to \mathbb{R}$ une certaine fonction qui soit continue au point $(\mu_0, \dots, \mu_{K-1})$. Alors, si $\hat{\boldsymbol{\mu}}_0^{(n)}, \hat{\boldsymbol{\mu}}_1^{(n)}, \dots, \hat{\boldsymbol{\mu}}_{K-1}^{(n)}$ sont les estimateurs de Monte-Carlo usuels pour resp. $\mu_0, \mu_1, \dots, \mu_{K-1}$ qu'on obtient à partir de simulations i.i.d. $\boldsymbol{\omega}_0, \boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_{K-1}$ de la loi $P$, l'estimateur par substitution*

$$\gamma(\hat{\boldsymbol{\mu}}_0^{(n)}, \hat{\boldsymbol{\mu}}_1^{(n)}, \dots, \hat{\boldsymbol{\mu}}_{K-1}^{(n)}) \tag{CX}$$

*converge (fortement) vers notre quantité d'intérêt.* ◇

*Remarque* (CY). Dans ce théorème, vous on a utilisé les *mêmes simulations* de $\boldsymbol{P}$ pour chacun de nos $K$ estimateurs $\boldsymbol{\mu}_k^{(n)}$, ce qui économise ainsi du temps de calcul.

Néanmoins, si nous sommes dans un cas où on s'intéresse à une fonction $\gamma(\mu_0, \mu_1)$ de deux espérances qui proviennent de deux "univers" complètement différents, $\mu_0$ étant définie comme $\mathbb{E}(f_0(P_0))$ et $\mu_1$ comme $\mathbb{E}(f_1(P_1))$, où $P_0$ et $P_1$ sont des mesures de probabilités distinctes, portant sur des espaces respectifs $\Omega_0$ et $\Omega_1$ potentiellement différents, alors évidemment on sera obligé d'utiliser des simulations complètement différentes pour les lois $P_0$ et $P_1$. Concrètement, on prendra alors $n$ simulations i.i.d. $\hat{\boldsymbol{\omega}}_0^{(0)}, \dots, \hat{\boldsymbol{\omega}}_{n-1}^{(0)}$ de la loi $P_0$, et, indépendamment des précédentes, $n$ simulations i.i.d. $\hat{\boldsymbol{\omega}}_0^{(1)}, \dots, \hat{\boldsymbol{\omega}}_{n-1}^{(1)}$ de la loi $P_1$ ; on définira, pour $k \in \{0, 1\}$,

$$\hat{\mu}_{k\checkmark}^{(n)} := \frac{1}{n} \sum_{i=0}^{n-1} f_k(\omega_i^{(k)}) \, ; \tag{CZ}$$

et on estimera $\gamma(\mu_0, \mu_1)$ par $\gamma(\hat{\boldsymbol{\mu}}_0^{(n)}, \hat{\boldsymbol{\mu}}_1^{(n)})$.

En fait, ce cas n'est pas réellement différent du cas général, car on peut considérer que les couples $(\boldsymbol{\omega}_i^{(0)}, \boldsymbol{\omega}_i^{(1)})$ sont des simulations i.i.d. de la loi produit $P_0 \otimes P_1$ sur $\Omega_0 \times \Omega_1$, et que $f_k(P_k)$ est la mesure-image de $P_0 \times P_1$ par $f_k \circ \pi_k$, où $\pi_k$ désigne la projection canonique de $\Omega_0 \times \Omega_1$ dans $\Omega_k$. ♧

## Détermination de l'intervalle de confiance pour une fonction d'espérance

Nous venons de voir qu'il était très simple d'utiliser la méthode de Monte-Carlo pour estimer une fonction d'une ou plusieurs espérances, grâce à l'idée de substitution. Obtenir un intervalle de confiance associé à ces estimateurs, en revanche, s'avère plus délicat, ainsi que nous allons le voir à présent.

Dans le cas où on cherche à estimer une fonction d'*une seule* espérance, i.e. une quantité de la forme $\gamma(\mu)$ avec $\mu := \mathbb{E}(f(P))$ (avec $f(P)$ d'intégrabilité $\mathrm{L}^2$), on peut simplement dire que, puisque $[\hat{\boldsymbol{\mu}} \pm \hat{\boldsymbol{\sigma}} \,/\, n^{1/2}]$ est un intervalle de confiance asymptotique pour $\mu$, alors l'image directe de cet intervalle par la fonction $\gamma(\bullet)$ sera un intervalle de confiance asymptotique pour $\gamma(\mu)$. Dans les situations pratiques, la fonction $\gamma(\bullet)$ sera (continue et) soit croissante, soit décroissante (du moins sur la zone considérée), de sorte que cette image directe s'écrira, selon le cas, $[\gamma(\hat{\boldsymbol{\mu}} - \hat{\boldsymbol{\sigma}}), \gamma(\hat{\boldsymbol{\mu}} + \hat{\boldsymbol{\sigma}})]$ ou $[\gamma(\hat{\boldsymbol{\mu}} + \hat{\boldsymbol{\sigma}}), \gamma(\hat{\boldsymbol{\mu}} - \hat{\boldsymbol{\sigma}})]$.

☛ Attention, le paragraphe qui suit présente volontairement une méthode de mauvaise qualité, à des fins pédagogiques : mais cette méthode ne devra pas être utilisée en pratique !

On pourrait étendre le raisonnement précédent au cas d'une fonction de plusieurs espérances, même si ce serait un chouïa plus compliqué. Cela fonctionnerait ainsi : on déterminerait d'abord un intervalle de confiance au risque $\alpha / K$ (attention !) pour chacun des $\mu_k$ ; puis on en déduirait une zone de confiance au niveau $\alpha$, de forme parallélipipédique, pour le $K$-uplet $(\mu_0, \dots, \mu_{K-1})$ ; et on considèrerait enfin l'image directe de cette zone de confiance (laquelle, en pratique, sera normalement le plus petit intervalle contenant les $2^K$ images des coins du parallélipipède) pour en déduire un intervalle de confiance pour $\gamma(\mu_0, \dots, \mu_{K-1})$.

Cependant, dès lors que $K > 1$, il s'avère que cette idée de prendre l'image directe d'un pavé de confiance serait sensiblement sous-optimale ! Il existe en revanche une autre méthode, que nous allons présenter maintenant, et qui fournit pour sa part un résultat optimal (au premier ordre) : c'est uniquement cette méthode-là que vous devrez employer !

Pour présenter notre méthode, commençons par le cas simple où on cherche à estimer une fonction $\gamma(\mu)$ qu'une seule espérance $\mu := \mathbb{E}(f(\boldsymbol{P}))$, avec $f(\boldsymbol{P})$ d'intégrabilité $\mathrm{L}^2$. Nous supposerons ici que la fonction $\gamma(\bullet)$ est de classe $\mathrm{C}^1$, avec une dérivée non nulle dans la zone qui nous intéresse.

Soit $\hat{\boldsymbol{\mu}}^{(n)}$ notre estimateur de Monte-Carlo pour l'espérance $\mu$, $\hat{\boldsymbol{\gamma}}^{(n)} := \gamma(\hat{\boldsymbol{\mu}}^{(n)})$ l'estimateur pour $\gamma(\mu)$ qu'on en déduit par substitution ; et notons $\boldsymbol{\varepsilon}^{(n)} := \hat{\boldsymbol{\mu}}^{(n)} - \mu$ et $\boldsymbol{\eta}^{(n)} := \hat{\boldsymbol{\gamma}}^{(n)} - \gamma(\mu)$ les erreurs respectives commises par ces deux estimateurs.

Nous savons par le théorème-limite central que, lorsque $n \to \infty$, on a asymptotiquement $\mathrm{Loi}(\hat{\boldsymbol{\varepsilon}}^{(n)}) \approx n^{-1/2}\sigma \times \mathrm{Normale}(0,1)$, où $\sigma$ désigne l'écart-type de $f(\boldsymbol{P})$. Il s'ensuit que, dans cette asymptotique, on aura une probabilité extrêmement grande d'avoir $|\hat{\boldsymbol{\varepsilon}}| \ll 1$. Mais alors, pour calculer $\gamma(\hat{\boldsymbol{\mu}}^{(n)})$, on peut sans dommage remplacer la fonction $\gamma(\bullet)$ par son *approximation affine* au voisinage de $\mu$ :

$$\gamma(\hat{\boldsymbol{\mu}}^{(n)}) = \gamma(\mu + \hat{\boldsymbol{\varepsilon}}^{(n)}) \approx \gamma(\mu) + \gamma'(\mu) \times \hat{\boldsymbol{\varepsilon}}^{(n)} : \tag{DA}$$

autrement dit,

$$\boldsymbol{\eta}^{(n)} \approx \gamma'(\mu) \times \hat{\boldsymbol{\varepsilon}}^{(n)}. \tag{DB}$$

Or, comme nous l'avons dit, la loi de $\boldsymbol{\varepsilon}^{(n)}$ est approximativement $n^{-1/2}\sigma \times \mathrm{Normale}(0,1)$ : par conséquent, on en déduit finalement que

$$\begin{aligned}\mathrm{Loi}(\boldsymbol{\eta}^{(n)}) &\approx n^{-1/2}\gamma'(\mu)\sigma \times \mathrm{Normale}(0,1) \\ &= \mathrm{Normale}(0, n^{-1}\gamma'(\mu)^2\sigma^2) \approx \mathrm{Normale}(0, n^{-1}\gamma'(\hat{\mu}^{(n)}_{\checkmark})^2(\hat{\sigma}^{(n)}_{\checkmark})^2),\end{aligned} \tag{DC}$$

où dans la dernière appoximation nous avons simplement remplacé les quantités numériquement inconnues $\mu$ et $\sigma$ par leurs estimateurs respectifs. Cela suggère, pour avoir un intervalle de confiance au niveau $\alpha$ pour $\gamma(\mu)$, de prendre

$$[\gamma(\hat{\boldsymbol{\mu}}^{(n)}) \pm |\gamma'(\hat{\boldsymbol{\mu}}^{(n)})|\hat{\boldsymbol{\sigma}}^{(n)} / \sqrt{n}] : \tag{DD}$$

le théorème (DI) *infra* nous dire qu'il s'agit effectivement là d'un intervalle de confiance asymptotique.

Maintenant que nous avons vu la démarche pour construire l'intervalle de confiance pour une fonction d'un espérance, voyons à présent cas d'une fonction de *plusieurs* espérances, i.e. d'une quantité de la forme $\gamma(\mu_0, \dots, \mu_{K-1})$ avec

$\mu_k = \mathbb{E}(f_k(\boldsymbol{P}))$. D'un point de vue strictement technique, il va s'agir exactement de la même démarche que précédemment, à ceci près que la fonction $f(\bullet)$ dont nous considèrerons l'espérance sera cette fois-ci à valeurs *vectorielles* (dans $\mathbb{R}^K$, les composantes respectives de notre fonction correspondant aux $f_k(\bullet)$), et que la fonction $\gamma(\bullet)$ ira de $\mathbb{R}^K$ dans $\mathbb{R}$. En pratique cependant, le formalisme technique sera un peu plus compliqué, car l'analogue de la variance pour un vecteur aléatoire ne correspond pas à un scalaire, ni même à un vecteur, mais à une *matrice* : à savoir, la matrice des covariances. La covariance entre les différentes variables $f_k(\boldsymbol{\omega})$ va donc jouer ici un rôle essentiel !

Posons $\boldsymbol{V} \coloneqq \mathrm{Var}^{\boldsymbol{\omega}\sim\boldsymbol{P}}(f_0(\boldsymbol{\omega})); \dots ; f_{K-1}(\boldsymbol{\omega}))$ la matrice des covariances du vecteur aléatoire formé par les $f_k(\boldsymbol{\omega})$, i.e. la matrice carrée symétrique de taille $\boldsymbol{K}\times\boldsymbol{K}$ dont les entrées sont données par

$$\mathfrak{v}_{k,k'} \coloneqq \mathrm{Cov}^{\boldsymbol{\omega}\sim\boldsymbol{P}}(f_k(\boldsymbol{\omega}), f_{k'}(\boldsymbol{\omega})). \tag{DE}$$

Le théorème-limite central nous dit alors que, lorsqu'on réalise $\boldsymbol{n}$ simulations de $\boldsymbol{P}$ et qu'on en déduit nos estimateurs de Monte-Carlo $\hat{\boldsymbol{\mu}}_k^{(n)}$, on a approximativement

$$\begin{pmatrix} \hat{\boldsymbol{\mu}}_0^{(n)} \\ \vdots \\ \hat{\boldsymbol{\mu}}_{K-1}^{(n)} \end{pmatrix} \sim \mathrm{Normale}\left(\begin{pmatrix} \hat{\mu}_0 \\ \vdots \\ \mu_{K-1} \end{pmatrix}, n^{-1}\boldsymbol{V}\right) \tag{DF}$$

Par conséquent, en linéarisant la fonction $\gamma(\bullet)$ au voisinage de $(\mu_0; \dots ; \mu_{K-1})$, on en déduit que

$$\gamma(\hat{\boldsymbol{\mu}}_0^{(n)}, \dots, \hat{\boldsymbol{\mu}}_{K-1}^{(n)}) \approx \mathrm{Normale}\big(\gamma(\mu_0, \dots, \mu_{K-1}), n^{-1}\vec{\boldsymbol{D}}^{\mathsf{T}}\boldsymbol{V}\vec{\boldsymbol{D}}\big), \tag{DG}$$

où

$$\vec{\boldsymbol{D}} \coloneqq \big(\vec{\nabla}\gamma\big)(\mu_0, \dots, \mu_{K-1}). \tag{DH}$$

On peut alors, comme précédemment, remplacer $\vec{\boldsymbol{D}}$ et $\boldsymbol{V}$ par leurs estimateurs de Monte-Carlo respectifs, et en déduire un intervalle de confiance pour $\gamma(\mu_0, \dots, \mu_{K-1})$.

Le théorème suivant formalise ce que nous venons de dire :

! **Théorème (DI).** *Soit $\boldsymbol{P}$ une loi sur un certain espace $\boldsymbol{\Omega}$, qu'on sait simuler ; soient $f_0(\bullet), \dots, f_{K-1}(\bullet)$ des fonctions de $\boldsymbol{\Omega}$ dans $\mathbb{R}$ qu'on sait évaluer numériquement, telles que les $f_k(\boldsymbol{P})$ soient d'intégrabilité $\mathrm{L}^2$ ; et soit $\gamma\colon \mathbb{R}^K \to \mathbb{R}$ une fonction de régularité $\mathrm{C}^1$. Notant $\mu_0, \dots, \mu_{K-1}$ les espérances respectives de $f_0(\boldsymbol{P}), \dots, f_{K-1}(\boldsymbol{P})$, $\vec{\boldsymbol{D}}$ le gradient de $\gamma(\bullet)$ évalué en $(\mu_0; \dots ; \mu_{K-1})$, et $\boldsymbol{V}$ la matrice des covariances du vecteur aléatoire $(f_0(\boldsymbol{\omega}), \dots, f_{K-1}(\boldsymbol{\omega}))$ lorsque $\boldsymbol{\omega}$ suit la loi $\boldsymbol{P}$, supposons que la quantité (automatiquement positive) $\vec{\boldsymbol{D}}^{\mathsf{T}}\boldsymbol{V}\vec{\boldsymbol{D}}$ est non nulle. Alors, pour $\boldsymbol{\alpha}$ et $z$ liés par la relation $\mathbb{P}(|\mathrm{Normale}(0,1)| > z) = \alpha$, on obtient un intervalle de confiance asymptotique de niveau $\boldsymbol{\alpha}$ pour $\gamma(\mu_0, \dots, \mu_{K-1})$ qde la façon suivante : on considère $\boldsymbol{n}$ simulations indépendantes $\omega_{0\checkmark}, \dots, \omega_{(n-1)\checkmark}$ de $\boldsymbol{P}$ ; on note $\hat{\mu}_{k\checkmark}^{(n)}$ les moyennes empiriques respectives des $f_k(\omega_{k\checkmark})$, on note $\hat{\boldsymbol{V}}_{\checkmark}$ la matrice des covariances empiriques des vecteurs $(f_0(\omega_{i\checkmark}), \dots, f_{K-1}(\omega_{i\checkmark}))$ ; et on considère l'intervalle*

$$\Big[\gamma(\hat{\mu}_{0\checkmark}^{(n)}, \dots, \hat{\mu}_{(K-1)\checkmark}^{(n)}) \pm z\sqrt{\vec{\nabla}^{\mathsf{T}}\gamma(\hat{\mu}_{0\checkmark}^{(n)}, \dots, \hat{\mu}_{(K-1)\checkmark}^{(n)}) \cdot \hat{\boldsymbol{V}}_{\checkmark}^{(n)} \cdot \vec{\nabla}\gamma(\hat{\mu}_{0\checkmark}^{(n)}, \dots, \hat{\mu}_{(K-1)\checkmark}^{(n)})}\,/\,n^{1/2}\Big]. \tag{DJ}$$

◇

*Exemple* (DK). Dans le cadre du championnat de basket, supposons que l'enjeu soit, non pas d'évaluer la probabilité de victoire de Nancy, mais le ratio entre la probabilité de victoire de Limoges (qui correspond à l'équipe numéro 1 dans notre modélisation) et celle de Nancy [‡‡]. Notons $\boldsymbol{N}$ l'évènement « Nancy gagne » et $\boldsymbol{L}$ l'évènement « Limoges gagne » ; la méthode de Monte-Carlo nous donne alors, avec la même boucle de calculs, des estimateurs de $p_N \coloneqq \mathbb{E}(\mathbf{1}_N)$ et $p_L \coloneqq \mathbb{E}(\mathbf{1}_L)$, estimateurs notés resp. $\hat{p}_N$ et $\hat{p}_L$. Notre but est de savoir quelle sera la marge d'erreur sur l'estimateur $\hat{q} \coloneqq \hat{p}_L \,/\, \hat{p}_N$. Nous savons que, au premier ordre, on a

$$\begin{pmatrix} \hat{p}_L \\ \hat{p}_N \end{pmatrix} \overset{\text{loi}}{=} \begin{pmatrix} p_L \\ p_N \end{pmatrix} + \text{Normale}(\vec{0}, \boldsymbol{C}) n^{-1/2} + \text{o}(n^{-1/2}), \tag{DL}$$

où $\boldsymbol{C}$ est la matrice de covariance du vecteur aléatoire $(\mathbf{1}_L, \mathbf{1}_N)^\mathsf{T}$, que nous pouvons estimer là encore à partir des mêmes simulations. Or $q \coloneqq p_L \,/\, p_N$ et $\hat{q} \coloneqq \hat{p}_L \,/\, \hat{p}_N$ se déduisent respectivement de $(p_L, p_N)$ et $(\hat{p}_L, \hat{p}_N)$ par l'application « quotient » : $(x, y) \mapsto x \,/\, y$, dont les dérivées partielles sont :

$$\frac{\partial(x \,/\, y)}{\partial x} = \frac{1}{y} \,; \tag{DM}$$

$$\frac{\partial(x \,/\, y)}{\partial y} = -\frac{x}{y^2}. \tag{DN}$$

On en déduit que, toujours au premier ordre, on aura

$$\hat{q} = q + \text{Normale}\big(0, (1/p_N, -p_L \,/\, p_N^2) \boldsymbol{C} (1/p_N, -p_L \,/\, p_N^2)^\mathsf{T}\big) n^{-1/2} + \text{o}(n^{-1/2}). \tag{DO}$$

Bien entendu, la valeur exacte de la variance $(1/p_N, -p_L \,/\, p_N^2) \boldsymbol{C} (1/p_N, -p_L \,/\, p_N^2)^\mathsf{T}$ qui apparait ci-dessus n'est pas connue exactement ; mais l'estimation empirique que nous pourrons en faire sera suffisamment précise pour que l'intervalle reste convergent !

Cela conduit au code ci-dessous, où nous avons en outre utilisé que, compte tenu du fait que $\mathbf{1}_N$ et $\mathbf{1}_L$ sont des indicatrices disjointes, il y a en l'occurrence une relation directe entre $\boldsymbol{C}$ et $(p_N, p_L)$ (sinon, nous aurions dû nous servir de notre boucle de Monte-Carlo pour y estimer aussi $\mathbb{E}(\mathbf{1}_N^2)$, $\mathbb{E}(\mathbf{1}_L^2)$ et $\mathbb{E}(\mathbf{1}_L \mathbf{1}_N)$) :

$$\boldsymbol{C} = \begin{pmatrix} p_L(1-p_L) & -p_L p_N \\ -p_L p_N & p_N(1-p_N) \end{pmatrix}. \tag{DP}$$

```
# La fonction "basket_ratio" estime le ratio entre les probabilités de victoire
# respectives de Limoges et de Nancy.def basket_rejet():
def basket_ratio():
    NSIM = 60000
    NANCY = 2
    LIMOGES = 1
    NEQ = 16
    # "bilanN" et "bilanL" sont les nombres de simulations respectives où Nancy
    # et Limoges ont été championnes.
    bilanN = 0
    bilanL = 0
    for i in range(NSIM):
```

[‡‡]. Cela correspond à fixer la cote d'un pari entre Nancy et Limoges dans lequel chacun récupère sa mise si aucun des deux équipes ne gagne.

```
        # Notez qu'on utilise la même simulation, à chaque passage dans la
        # boucle, pour savoir qui de Nancy et de Limoges a été championne.
        c = aux.championne()
        bilanN += (c == NANCY)
        bilanL += (c == LIMOGES)
    # "pN" et "pL" sont les estimateurs respectifs des probabilités de victoire
    # de Nancy et Limoges.
    pN = bilanN / NSIM
    pL = bilanL / NSIM
    # On en déduit, dans un premier temps, l'estimateur de q : cette partie est
    # immédiate...
    estr_q = pL / pN
    # Matrice de covariance (estimée) entre les deux indicatrices ; en
    # l'ccurrence les sommes de carrés et de produits se déduisent directement
    # des sommes elles-mêmes, donc tout se calcule à partir de pN et pL.
    C = array([[pL * (1 - pL), -pL * pN], [-pL * pN, pN * (1 - pN)]])
    # Matrice de covariance de l'estimation jointe de (pL, pN).
    Cest = C / NSIM
    # On en déduit, en appliquant la différentielle de la fonction quotient en
    # (l'estimation de) (pL, pN), la variance de l'estimateur de q.
    Dquotient = array([[1 / pN], [-pL / pN ** 2]])
    varq = float(Dquotient.T @ Cest @ Dquotient)
    marge = 1.96 * varq ** (1 / 2)
    print('Le ratio de probabilités est {:.4f} ± {:.4f}'.format(
        estr_q, marge))
    return
```

♧

## 2.3 Intérêt et limites de la méthode de Monte-Carlo

### Robustesse

Nous avons vu dans la § 2.1 que, notant $n$ est la quantité de calculs effectuée, l'erreur commise par la méthode de Monte-Carlo décroit comme $n^{-1/2}$ (sous réserve que $f$ soit $\mathrm{L}^2$). Il est remarquable que ce mode de convergence ne dépend pas de la structure précise du problème ! En particulier, au vu du paragraphe 2.1, cela signifie que grâce à la méthode de Monte-Carlo on pourra calculer (presque) n'importe quelle intégrale avec une précision en $\mathrm{O}(n^{-1/2})$. En comparaison, les méthodes classiques de calcul numérique d'intégrales ont une façon de converger qui se dégrade quand la dimension de l'espace d'intégration augmente ou quand la régularité de la fonction diminue : par exemple, en dimension $d$ la méthode des rectangles avec $n$ points d'évaluation converge en $\mathrm{O}(n^{-2/d})$ si la fonction à intégrer est lisse, et en $\mathrm{O}(n^{-1/d})$ seulement si la fonction n'est que lipschitzienne... On en retiendra le principe suivant :

!! *Point* (DQ). La méthode de Monte-Carlo est particulièrement bien adaptée au calcul d'intégrales :
- — Quand la dimension de l'espace d'intégration est grande (typiquement à partir de la dimension 4 ou 5) ;
- — Et d'autant plus que la fonction à intégrer est peu régulière.

♧

## Parallélisation

Un avantage important de la méthode de Monte-Carlo est sa capacité à la *parallélisation*. On dit qu'un calcul peut être mené de façon *parallèle* lorsqu'il peut être décomposé, pour l'essentiel, en une multitude de sous-calculs *indépendants* : informellement, un calcul est parallélisable lorsque cent unités de calcul (intelligemment coordonnées) peuvent l'effectuer plus rapidement qu'un seul [∗].

Par exemple, simuler l'évolution de la trajectoire d'une bille dans un potentiel est une tâche typiquement *non* parallélisable, puisqu'on est obligé de calculer successivement la position de la bille aux différents moments... À l'inverse, la méthode de Monte-Carlo, avec ses simulations indépendantes, se parallélise très bien :

*Point* (DR). La méthode de Monte-Carlo pour le calcul d'une espérance peut être parallélisée extrêmement efficacement, de la façon suivante. Supposons par exemple qu'on dispose de 100 unités de calculs pour effectuer $10^9$ simulations. Si chaque unité de calcul effectue indépendamment $10^7$ simulations et calcule la moyenne empirique obtenue pour ses propres simulations, il n'y a à la fin plus qu'à mettre les 100 moyennes partielles en commun pour obtenir la moyenne globale. ♧ !

*Remarque* (DS). Attention, dans le point précédent, il faut bien veiller à ce que chaque unité de calcul effectue les simulations avec des aléas *indépendants* ! En particulier, si la méthode de Monte-Carlo recourt à des nombres *pseudo*-aléatoires (comme c'est généralement le cas), il faut s'assurer que les différentes unités de calcul utilisent pour leurs générateurs pseudo-aléatoires des graines qui ne risquent pas d'interférer. ♧

*Remarque* (DT). Dans les exercices de ce cours, nous n'aborderons guère ces questions de parallélisation, dont l'implémentation requerrait des outils de programmation un peu plus avancés que ce dont vous avez l'habitude, et/ou spécifiques à telle ou telle plateforme de programmation. Néanmoins, gardez bien à l'esprit la grande facilité avec laquelle la méthode de Monte-Carlo pour le calcul d'une espérance peut être parallélisée : car, pour certaines applications du monde industriel, cela pourra présenter un intérêt majeur... ! ♧

*Remarque* (DU). En pratique, il peut exister des formes de parallélisation au sein d'une même machine, en exploitant plusieurs « cœurs » dans un même processeur, ou en utilisant intelligemment le « pipeline » dans un cœur donné.

En pratique, dans la mesure où ce parallélisme s'appuie sur les particularités de l'architecture matérielle du processeur, son exploitation requiert de passer par une bibliothèque logicielle spécialisée pour utiliser optimalement les spécificités du matériel : par exemple, pour le langage python, PyTorch est une telle bibliothèque, très utilisée pour la programmation des réseaux de neurones. ♧

*Remarque* (DV). Une forme particulière de parallélisation consiste en la *vectorisation*. On dit qu'un calcul est mené de façon vectorielle lorsqu'on traite applique simultanément la même opération à un grand nombre de données : par exemple, à deux « vecteurs » de nombre réels $[a_0, \dots, a_{N-1}]$ et $[b_0, \dots, b_{N-1}]$ de même taille étant donnés, on peut appliquer l'instruction « quotient terme à terme » pour obtenir, *en une seule étape*, le vecteur $[a_0 \,/\, b_0, \dots, a_{N-1} \,/\, b_{N-1}]$ [†]. Là où une architecture matérielle

[∗]. Attention : un calcul parallélisable peut certes être rendu très rapide si on dispose de suffisamment d'unités de calcul ; mais cela n'est avantageux que si le facteur limitant est le *temps* : à l'inverse, si le facteur limitant est la consommation de chaque unité de calcul, on ne gagne rien à diviser le calcul en cent sous-calculs, voire on y perd en fonction du type de découpage...

[†]. On parle alors de parallélisme « SIMD », pour « *Single Instruction, Multiple Data* » (une instruction, plusieurs multiples).

"standard" requerrait, pour ce faire, de passer par une boucle de taille $N$, certains processeurs (et notamment les cartes graphiques) prévoient des fonctionnalités permettant de procéder à de tels calculs vectoriels de façon à la fois simple à implémenter, rapide et/ou moins gourmande en consommation électrique.

Là encore, l'exploitation de ces fonctionnalités matérielles requiert en pratique l'utilisation d'une bibliothèque spécialisée. Certains langages de programmation, comme MATLAB pour l'analyse numérique ou *R* pour le traitement de données, qui sont conçus dès le départ pour traiter leurs données sous forme vectorielle, implémentent nativement de telles bibliothèques. ♧

## Raffinement

*Point* (DW). Si un premier calcul de Monte-Carlo a été effectué (disons avec $2\times10^6$ simuations) et que sa précision s'avère insuffisante (disons qu'il aurait fallu $5\times10^6$ simulations), la méthode de Monte-Carlo présente cet avantage considérable qu'on peut s'appuyer sur les calculs déjà effectués comme une première étape du calcul raffiné ! En effet, si on a stocké quelque part les résultats relatifs à nos 200 000 premières simulations, il n'y aura plus que $3\times10^6$ simulations à faire indépendamment pour arriver au total désiré de $5\times10^6$.

En outre, si l'on souhaite ainsi raffiner notre calcul, on n'a pas besoin de connaitre le résultat détaillé des $2\times10^6$ premières simulations : il suffit d'avoir gardé en mémoire le nombre de simulations effectuées, le total (ou la moyenne) des résultats des simulations, et éventuellement le total des carrés (ou la variance empirique) des simulations ! Il ne faut jamais hésiter à garder en mémoire ces résultats intermédiaires-là [‡], au cas où on souhaiterait raffiner le calcul ultérieurement, quand bien même cette éventualité nous semble peu probable. ♧

*Remarque* (DX). D'une certaine manière, on peut dire que le raffinement est une forme de parallélisation *dans le temps* ! ⌒ ♧

## Le caractère aléatoire, un problème ?

À l'inverse, la méthode de Monte-Carlo présente aussi des handicaps non négligeables. Un premier reproche assez évident est que cette méthode ne permet pas d'avoir une certitude absolue sur la validité d'un résultat. Cependant, il s'agit là plutôt d'une préoccupation de théoriciens : *en pratique*, cette limitation ne doit pas vous effrayer ! En effet, pour ce qui concerne les applications au monde réel, une probabilité inférieure à $10^{-10}$ est complètement négligeable [§] : donc, si vous avez un moyen de descendre l'incertitude jusqu'à un niveau aussi faible, c'est *concrètement* comme si vous aviez obtenu une certitude absolue !

*Remarque* (DY). Nous avons vu dans la remarque (CT) que, pour qu'on puisse considérer que l'asymptotique d'un intervalle de confiance au niveau $10^{-10}$ est atteinte, il faudrait au moins de l'ordre de $10^{20}$ simulations : ce qui semble rendre illusoire d'atteindre un niveau de risque aussi faible... Cependant, il existe d'autres moyens de contrôler les risques, qui permettent d'obtenir un niveau extrêmement robuste sans

[‡]. En pratique, on peut même se contenter de retenir simplement l'intervalle de confiance fourni par le calcul, car les quantités nécessaires au raffinement peuvent être reconstituées à partir de celui-ci ! ◡

[§]. Pour vous donner une idée, une probabilité de $10^{-10}$, c'est à peu près le risque que vous mouriez de la rupture d'un anévrisme non détecté pendant la séance de cours consacrée à ce passage, et bien moins que la probabilité que votre entreprise ait fait faillite sans que vous le sachiez !

trop démultiplier les simulations. En particulier, si on lance 40 simulations indépendantes qui ont chacune au moins 50 % de chances de fournir un intervalle correct, la probabilité que le résultat ne soit dans *aucun* des intervalles de confiance trouvés sera de seulement $10^{-12}$ : on peut ainsi, par réunion de 40 intervalles indépendants à 50 % chacun, obtenir un intervalle de confiance de niveau $10^{-12}$ (certes particulièrement large) qui soit fiable ![¶] ♧

### Caractère asymptotique des intervalles de confiance

C'est sans doute le pire piège auquel vous serez confrontés pour appliquer la méthode de Monte-Carlo. On a tendance à oublier que les intervalles de confiance donnés n'ont le niveau de précision annoncé que *sous l'hypothèse que la fonction dont on évalue l'espérance soit* $L^2$, et que même sous cette hypothèse ils ne sont valables qu'*asymptotiquement* !!

Or, dans bien des cas, on n'est pas en mesure d'être capable de prouver des intervalles de confiance non asymptotiques pour les données qu'on calcule (voire parfois on ne sait même pas s'assurer qu'elles sont $L^2$). Cela pose un vrai problème, et certains des exercices associés à ce cours montrent comment on peut être piégé et obtenir des intervalles de confiances faux sans qu'aucun signe évident nous en ait averti ! C'est donc *le* point sur lequel il convient d'être prudent.

*Remarque* (DZ). Il existe en fait des moyens d'obtenir des intervalles de confiance non asymptotiques ; cependant, pour cela il faut être capable d'obtenir un contrôle *explicite* sur la loi de probabilité : par exemple, avoir une borne explicite sur la v.a. dont nous considérons l'espérance. Ce point ne sera pas abordé plus en détail dans le cadre de ce polycopié ; mais certains des exercices associés au cours donneront quelques idées sur la question. ♧

### Précision

Par exemple, si nous souhaitons utiliser la méthode de Monte-Carlo pour un calcul de $\pi$ (ce qui est tout-à-fait possible), la précision du calcul pour un cout de calcul $n$ sera en $n^{-1/2}$, soit $O(\log n)$ décimales exactes. Alors que les méthodes non probabilistes permettent d'obtenir assez facilement un nombre de décimales exactes en $O(n^{1/2})$ voire en $O(n^{1-o(1)})$...

Cela dit, en contexte industriel, on n'a généralement pas besoin de descendre à des niveaux de précision extrêmement fins, de sorte qu'il sera rare que l'écueil ci-dessus soit complètement rédhibitoire... Il n'en demeure pas moins que, dès lors qu'on aura besoin de résultats assez précis, la grande lenteur des méthodes de Monte-Carlo dans ce cas pourra constituer une bonne raison d'y privilégier des méthodes déterministes concurrentes !

## 2.4 Appendice : $z$-valeurs et risque

Lorsqu'on manipule des intervalles de confiance, on est amené à jongler entre deux façons d'exprimer à quel point notre intervalle de confiance est large :

[¶]. Attention toutefois : pour que l'idée ci-dessus fonctionne correctement, il faut qu'on ait une garantie *absolue* sur le fait que nos intervalles de confiance individuels soient bien à 50 %... Ce qui, en pratique, est très difficile à avoir ! ⁖

— D'une part, le *niveau de risque*, traditionnellement noté '$\alpha$', qui est la probabilité (asymptotique) que l'intervalle ne contienne pas la vraie valeur recherchée ;
— Et d'autre part, ce qu'on appelle les *z-valeurs* associées aux bornes de l'intervalle de confiance : la $z$-valeur d'une borne $b$ (traditionnellement notée '$z$', d'où son nom) correspondant au quotient $(b - \hat{\mu}_{\checkmark}) / \mathrm{Var}^{1/2}(\hat{\boldsymbol{\mu}})$, autrement dit à l'écart entre l'estimation $\hat{\mu}_{\checkmark}$ et la borne $b$, mesuré en nombre d'écarts-types [‖] de l'estimateur.

Lorsque les estimateurs ont un comportement (asymptotiquement) normal (comme c'est le cas pour nous), et qu'on considère des intervalles de confiance centrés sur l'estimateur (auquel cas les $z$-valeurs associées aux bornes sont opposées), on a la relation suivante entre le niveau de risque $\alpha$ et la valeur absolue $z$ des $z$-valeurs des bornes de l'intervalle :

$$\mathbb{P}(|\mathrm{Normale}(0,1)| > z) = \alpha. \tag{EA}$$

La figure 2.1 indique les correspondances entre certaines valeurs classiques pour $z$ et $\alpha$.

| $z$ | $\alpha$ |
|---|---|
| 0 | 1 |
| 1 | 32 % |
| 1,5 | 14 % |
| 2 | 4,6 % |
| 2,5 | 1,3 % |
| 3 | 2,7 ‰ |
| 4 | $6{,}4 \times 10^{-5}$ |
| 5 | $5{,}8 \times 10^{-7}$ |
| $\infty$ | 0 |

| $\alpha$ | $z$ |
|---|---|
| 20 % | 1,29 |
| 10 % | 1,65 |
| 5 % | 1,96 |
| 2 % | 2,33 |
| 1 % | 2,58 |
| 1 ‰ | 3,30 |
| $10^{-4}$ | 3,90 |
| $10^{-5}$ | 4,42 |
| $10^{-6}$ | 4,90 |

TABLE 2.1 – Quelques correspondances entre $z$-valeurs et niveaux de risque dans le cas normal. Le sens des arrondis a été choisi de sorte qu'on ait toujours $\mathbb{P}(|\mathrm{Normale}(0,1)| > z) \leqslant \alpha$.

.

Lorsque les niveaux de risque ou les $z$-valeurs spécifiant l'intervalle sont moins classiques, on aura besoin d'utiliser la fonction de répartition de la loi normale standard, fonction traditionnellement notée $\Phi(\bullet)$, déterminée par l'intégrale suivante :

$$\Phi(x) \coloneqq \mathbb{P}\big(\mathrm{Normale}(0,1) \leqslant x\big) = (2\pi)^{-1/2} \int_{-\infty}^{x} \mathrm{e}^{-x^2/2}\,\mathrm{d}x. \tag{EB}$$

Cette fonction a le mauvais gout de ne pas être « élémentaire », càd. de ne pas pouvoir s'écrire à partir des fonctions de base. Heureusement, la plupart des logiciels de calcul numérique implémentent, parmi leurs « fonctions spéciales », une fonction appelée *fonction d'erreur*, notée erf, définie par :

$$\begin{aligned} \mathrm{erf} : \mathbb{R} &\to ]\pm 1[ \\ x &\mapsto \frac{2}{\sqrt{x}} \int_0^x \mathrm{e}^{-x^2}\,\mathrm{d}x, \end{aligned} \tag{EC}$$

[‖]. Plus exactement, l'unité de base est ici l'*estimation* de l'écart-type *efficace*.

de sorte qu'on a

$$\Phi(x) = \big(1 + \mathrm{erf}(x/\sqrt{2})\big)/2. \qquad (\mathtt{ED})$$

Il est aussi souvent utile de pouvoir calculer la bijection réciproque $\Phi^{-1}$ de la fonction de répartition $\Phi$. Notant $\mathrm{erf}^{-1} : ]-1,1[ \to \mathbb{R}$ la bijection réciproque de erf (qui est elle aussi généralement programmée dans les logiciels de calcul numérique), on a d'après (ED) :

$$\Phi^{-1}(p) = \sqrt{2}\,\mathrm{erf}^{-1}(2p-1). \qquad (\mathtt{EE})$$

# Chapitre 3

# Vers la réduction de la variance

## 3.1 Notion d'efficacité

Le théorème (CO) nous dit que, sous réserve que $f$ soit de classe $L^2$, la largeur de l'intervalle de confiance décroit comme $n^{-1/2}$ quand le nombre de simulations $n$ augmente. Cela est dû au fait que l'estimateur $\hat{\mu}^{(n)}$ tombe à une distance d'ordre $O(n^{-1/2})$ de la véritable valeur $\mathbb{E}(f)$ : en effet, la variance de $\hat{\mu}^{(n)}$ décroit en $O(n^{-1})$, puisque d'après la formule d'additivité des variances pour les variables indépendantes, $\mathrm{Var}(\hat{\mu}^{(n)}) = (n^{-1})^2 \times n \,\mathrm{Var}(f) = n^{-1}\,\mathrm{Var}(f)$. De l'autre côté, le cout total du calcul (qui peut correspondre, selon les circonstances, au nombre de simulations, au temps de calcul, à la consommation de ressources de la ferme de calcul, voire au cout monétaire) croît en général linéairement en $n$. Cela suggère la définition suivante :

! **Définition (EF)** (Efficacité d'une estimation de Monte-Carlo)**.** Nous appellerons *efficacité* d'un calcul de Monte-Carlo l'inverse de la variance (effective) de $\hat{\mu}$ divisée par le cout de calcul. L'efficacité est essentiellement indépendante du nombre de simulations, et correspond, dans le cas de base (celui de la § 2.1) à l'inverse de la variance de la v.a. $f$ dont on estime l'espérance, multipliée par le nombre de simulations effectuées par unité de coût.

À coût de calcul fixé, la précision du calcul d'une quantité par la méthode de Monte-Carlo sera donc d'autant meilleure que l'efficacité du calcul est grande.

Quand le cout de calcul considéré est le nombre de simulations effectuées, nous parlerons d'« *efficacité par simulation* ». Pour la méthode de Monte-Carlo basique (BQ), cette efficacité par simulation est simplement l'inverse de $\mathrm{Var}(f)$. ♡

*Remarque* (EG) (Variance effective). Dans cette remarque, j'explique l'adjectif « effective » qui apparait entre parenthèses dans la définition de l'efficacité. Dans certaines situations, la quantité $\gamma$ qu'on cherche à estimer n'est déduite qu'*indirectement* de l'estimateur de Monte-Carlo. Dans un tel cas, sous réserve que l'estimateur de Monte-Carlo sur lequel on s'appuie ait des fluctuations gaussiennes décroissant en $n^{-1/2}$, il en va de même pour l'estimateur $\hat{\gamma}$ de $\gamma$ qui s'en déduit : il existe une constante $\sigma_\gamma^{\mathrm{eff}} < \infty$ telle que

$$\frac{\hat{\gamma}^{(n)} - \gamma}{\sigma_\gamma^{\mathrm{eff}} n^{-1/2}} \xrightarrow{\text{loi}} \mathrm{Normale}(0,1). \qquad \text{(EH)}$$

Le cas échéant, c'est la quantité $\sigma_\gamma^{\mathrm{eff}} n^{-1/2}$ qui est appelée « variance effective » de $\hat{\gamma}^{(n)}$. Mais il se peut dans certains cas que cela ne corresponde pas du tout à la variance de $\hat{\gamma}$, en particulier lorsque celle-ci est infinie... ! ♧

*Remarque* (EI). La définition (EF) parle de la variance (effective) de l'estimateur $\hat{\mu}$, autrement dit, du carré de la demi-largeur de l'intervalle de confiance *à* 1 *sigma* associé à cet estimateur. Mais en pratique, nous ne calculerons les efficacités que pour les *comparer* les unes aux autres (en valeur relative) : typiquement, pour dire que telle méthode est tant de fois plus efficace que telle autre. Il est donc suffisant de retenir que l'efficacité correspond à ce qu'on obtient, par unité de cout de calcul, en termes d'inverse de carré de largeur d'intervalle de confiance : et ce, sans se préoccuper si on est en train de parler d'intervalles à 1 sigma, à 95 % ou à 3 sigmas... Tout ce qui compte, c'est que, lorsqu'on dit qu'une méthode B est 4 fois plus efficace qu'une méthode A, cela signifie que le cout requis à la méthode B pour obtenir un niveau de précision donné est 4 fois inférieur à ce qu'il est pour la méthode A, ou encore, qu'à cout de calcul donné, l'intervalle fourni par la méthode B (pour un niveau de risque donné) sera $4^{1/2} = 2$ fois plus précis que celui fourni par la méthode A ! ♧

*Remarque* (EJ). Si l'on souhaite évaluer empiriquement l'efficacité d'un calcul, on pourra estimer la variance par la variance empirique (comme nous l'avons fait pour trouver les intervalles de confiance), ce qui fournira un estimateur convergent de l'efficacité. ♧

*Remarque* (EK). L'efficacité est une grandeur *dimensionnée* : si, par exemple, la quantité à évaluer se mesure en € et que le cout de calcul considéré est le temps de processeur requis (qui se mesure en s), l'efficacité sera en $€^{-2} \cdot s^{-1}$. ♧

## 3.2 Notion de réduction de la variance

### Qu'est-ce que la réduction de la variance ?

Nous avons vu dans la § 3.1 que le cout d'une calcul de Monte-Carlo était proportionnel au carré de la précision souhaitée, avec une constante de proportionnalité que nous avons appelée *efficacité*. En pratique, on recontre de nombreuses situations où les circonstances du problème font que le cout de calcul devient une véritable contrainte, et on cherche donc à limiter le cout de ce calcul. Mais comment ? Ce que nous allons voir dans le prochain chapitre, c'est que certaines manipulations algébriques permettent de ramener un calcul d'espérance par méthode de Monte-Carlo à un *autre* calcul de Monte-Carlo, dont le *résultat* est le même, mais dont l'*efficacité* sera différente (et, nous l'espérons, meilleure dans les cas qui nous intéressant !). C'est ce qu'on appelle la *réduction de la variance*.

Comme nous venons de le dire, les *techniques de réduction de la variance* se proposent, à l'aide de certaines manipulations algébriques, de ramener un calcul par Monte-Carlo à un autre calcul d'efficacité meilleure. L'appellation de "réduction de la variance" vient de ce que, dans nombre des méthodes exposées ci-dessous, on essaye d'abord d'améliorer l'efficacité *par simulation*, laquelle est précisément égale à l'inverse de la variance de la quantité simulée : améliorer l'efficacité par simulation, c'est donc réduire la variance.

*Remarque* (EL). Cela dit, il convient de garder à l'esprit que l'amélioration de l'efficacité *par simulation* ne se traduit pas nécessairement par l'amélioration de l'efficacité *tout court*, car la complexification de l'algorithme peut conduire à une augmentation du cout de calcul par simulation. ♧

### Le niveau zéro de la réduction de la variance : améliorer son code !

Les techniques de réduction de la variance qui vont être présentées dans ce chapitre fonctionnent au niveau *algorithmique* : pour améliorer l'efficacité, on change fondamentalement la *nature* du calcul effectué. Mais la première façon d'améliorer l'efficacité est d'abord d'optimiser son code de calcul ! En effet, pour faire calculer la même chose à votre ordinateur, l'ordre ou l'organisation des calculs peut lui demander beaucoup plus d'efforts...

Donc, utiliser des techniques de réduction de la variance, c'est très bien, mais si on ne conçoit pas bien son code, on ruine bêtement ses efforts ! Un code mal conçu peut faire perdre un facteur 2 voire 10 dans la vitesse d'exécution, et même, s'il est très mal conçu, rendre le programme totalement inopérant en pratique. Surtout, cela n'est pas valable que pour la méthode de Monte-Carlo, mais pour *toute* implémentation informatique !

*Exemple* (EM). À titre d'exemple, voyons comment nous pourrions améliorer notre code du Keno. Une première remarque qui vient à l'esprit est que l'appel à la fonction **gain**, qui donne le gain financier obtenu à partir d'un certain score, n'est pas utilisée de façon très intelligente... En effet, à chaque fois que nous avons à calculer un gain, nous allons refaire toute la boucle de calcul afférente au calcul de ce gain, alors qu'il est bien évident que nous pourrions calculer une fois pour toutes les différentes valeurs de gains associées aux différents scores (qui ne sont qu'en nombre fini et petit), et nous contenter de *lire* ensuite ces résultats stockés dans un tableau ! On obtient ainsi le programme suivant :

```
# La fonction "keno_v2" fait le même travail que "keno_IC", mais avec un code
# amélioré au niveau du calcul des gains, qu'on pré-réalise avant la boucle.
def keno_v2():
    NNUM = 80
    NCOCH = 20
    NSIM = 1000000
    total = 0
    total_carres = 0
    carton = [0 for n in range(NNUM)]
    for n in range(NCOCH):
        carton[n] = NCOCH - n
    # "table_gains" est une liste telle que "table_gains[s]" sera la valeur du
    # gain pour un score de s. Il aurait pu sembler plus logique d'utiliser un
    # dictionnaire, mais cela aurait été bien moins efficace à l'exécution.
    GAIN_MAX = NCOCH * (NCOCH + 1) // 2
    table_gains = [aux.gain(s) for s in range(GAIN_MAX + 1)]
    for i in range(NSIM):
        legain = table_gains[aux.score(carton, aux.tirage())]
        total += legain
        total_carres += legain * legain
    moyenne = total / NSIM
    variance_gain = total_carres / NSIM - moyenne * moyenne
    ect_estr = sqrt (variance_gain / NSIM)
    from scipy.special import erfinv
    c0 = sqrt(2) * erfinv(-.90)
    c1 = sqrt(2) * erfinv(+.998)
    print('Avec un risque de 0,1 % pour la sous-estimation,',
          'resp. de 5 % pour la surestimation,',
          'l\'espérance de gain du Keno se situe dans l\'intervalle',
```

```
        '[{:f}, {:f}] EUR.'.format(moyenne + c0 * ect_estr,
                                   moyenne + c1 * ect_estr))
    return
```

Sur ma machine, l'exécution du programme passe d'environ 19,1 secondes à 18,2 secondes, ce qui est sensible quoique pas très intéressant [*]. En fait, une analyse plus poussé montrerait que notre modification du programme rendu le calcul du gain deux fois plus rapide, mais que cela ne s'est pas reflété sur le temps global, car l'essentiel du temps de calcul était consacré à la simulation du tirage et au calcul du score. Toujours est-il que nous avons tout de même gagné un peu de temps ; avec bien entendu un résultat qui est exactement le même (sous réserve d'avoir pris la même graine pour le générateur pseudo-aléatoire), puisque les calculs relatifs à la simulation étaient identiques.

Mais on peut faire encore mieux ! Remarquons en effet que nous simulons l'ensemble des numéros tirés du carton, alors que tout ce qui nous intéresse est de savoir combien de numéros sont tirés parmi ceux de '0' à '19' (rappelons en effet qu'on peut jouer toujours la même grille, en cochant '20' pour le numéro '0', etc.). On peut choisir de regarder successivement les numéros de '0' à '20' et voir, conditionnellement à ceux qui ont été tirés ou non parmi les précédents, s'ils vont être tirée ou non. Cela conduit à une fonction `tirage_reduit` qui renvoie un 20-vecteur valant '`true`' en sa case $i$ quand le numéro coché pour $i$ a été tiré et '`false`' sinon. Le programme obtenu étant alors :

```
# La fonction "keno_v3" fait le même travail que "keno_v2", mais avec un
# code encore amélioré, où on ne simule les numéros tirés que parmi ceux
# qui sont cochés.
def keno_v3():
    NCOCH = 20
    NSIM = 1000000
    total = 0
    total_carres = 0
    # Cette fois-ci on utilise un carton þréduitÞ appelé "carton_red", qui ne
    # mentionne que les numéros cochés.
    carton_red = [NCOCH - n for n in range(NCOCH)]
    GAIN_MAX = NCOCH * (NCOCH + 1) // 2
    table_gains = [aux.gain(s) for s in range(GAIN_MAX + 1)]
    for i in range(NSIM):
        # La fonction "tirage_red" donne quels numéros ont été tirés parmi les
        # numéros 0 à 19; et la fonction "score_red" en déduit le score.
        legain = table_gains[aux.score_red(carton_red, aux.tirage_red())]
        total += legain
        total_carres += legain * legain
    moyenne = total / NSIM
    variance_gain = total_carres / NSIM - moyenne * moyenne
    ect_estr = sqrt (variance_gain / NSIM)
    from scipy.special import erfinv
    c0 = sqrt(2) * erfinv(-.90)
    c1 = sqrt(2) * erfinv(+.998)
    print('Avec un risque de 0,1 % pour la sous-estimation,',
          'resp. de 5 % pour la surestimation,',
          'l\'espérance de gain du Keno se situe dans l\'intervalle',
```

[*]. La façon dont cet arrangement accélère (ou pas) les calculs peut dépendre sensiblement du langage de programmation utilisé, via ce qui se passe à l'intérieur de l'ordinateur. À l'époque où ce cours reposait sur Matlab, on passait ainsi de 22,4 s à 16,6 s, ce qui était déjà nettement mieux.

```
        '[{:f}, {:f}] EUR.'.format(moyenne + c0 * ect_estr,
                                   moyenne + c1 * ect_estr))
    return
```

Cette fois-ci le gain de temps de calcul est impressionnant, puisqu'on passe à environ 5,7 secondes ! Noter que cette fois, les résultats obtenus ne sont pas *strictement* identiques, même en prenant une graine peudo-aléatoire identique, car la simulation utilise les appels à la fonction `random` de façon différente ; par contre, la précision du résultat est bien la même (l'intervalle de confiance a la même largeur), dans la mesure où les opérations effectuées sont identiques *en loi*.

On pourrait encore faire quelques améliorations à la marge, mais je voudrais en montrer une dernière plus radicale : changer de langage de programmation ! Le langage `python` que nous avons utilisé jusque-là est un langage conçu pour être *pratique*, mais il n'est que modérément *rapide* : en effet, c'est un langage d'assez haut niveau (on ne contrôle pas très bien les calculs effectivement faits par le processeur), interprété (le logiciel ne découvre les lignes du programme qu'au moment de l'exécution, et ne peut donc pas préparer d'optimisation en fonction de la structure globale du programme), et faiblement typé (à chaque fois que *Pyhton* rencontre un objet, il doit savoir s'il va le traiter comme un entier, un réel, une liste, ...). Tout cela perd beaucoup de temps à l'exécution : un langage de plus bas niveau, compilé, plus fortement typé devrait donc être susceptible de faire beaucoup mieux. Essayons donc avec le roi des langages de bas niveau, à savoir le langage C : le temps de calcul est alors de... 0,5 secondes ! Autant dire que, si notre préoccupation était vraiment la rapidité, on aurait mieux fait de tout de suite commencer par là... ♧

*Remarque* `(EN)`. Certaines optimisations de code peuvent avoir un résultat pratiquement nul, parce qu'on a amélioré une partie du code qui ne prenait qu'une toute partie du temps total. Il faut un peu d'expérience de la programmation pour savoir quelles sont les opérations les plus lourdes pour le processeur, et à quels endroits un code non optimisé risque donc de nous couter vraiment cher... Bien sûr, le bon sens aide aussi : il est ainsi évident qu'il y a énormément plus d'intérêt à optimiser une opération répétée un très grand nombre de fois à l'intérieur d'une boucle, plutôt qu'une opération effectuée une fois pour toutes au début du programme ! En outre, pour les développeurs professionnels, il existe des outils d'analyse permettant de savoir dans quelle partie du code le programme passe plus ou moins de temps, ce qui permet de guider la démarche d'optimisation... ♧

*Remarque* `(EO)`. Certains logiciels ou certains langages ont leurs spécificités qu'il est bon de connaitre pour accélérer l'exécution. Ainsi, sous MATLAB, les opérations portant sur les matrices sont pré-compilées pour être particulièrement rapides à l'exécution, de sorte qu'il est en général beaucoup plus efficace d'utiliser une fonction MATLAB toute faite plutôt que de passer par une boucle pour faire le même calcul case par case... Ces considérations sont surtout intéressantes lorsqu'on est spécialisé dans un langage donné. Dans la mesure où l'objectif de ce cours est de présenter les méthodes de Monte-Carlo de manière générale, sans mettre en avant un langage particulier, nous n'exploiterons dans la suite que les optimisations de code reposant sur des considérations informatiques générales, sans faire attention aux spécificités propres à MATLAB. ♧

### Les techniques de réduction de la variance peuvent se combiner

La suite de ce chapitre va présenter les principales techniques de réduction de la variance indépendamment, chacune à part. Mais vous ne devez pas perdre de vue que rien n'interdit de combiner plusieurs techniques de réduction de la variance successivement pour une efficacité encore meilleure ! Nous en verrons des exemples dans les exercices.

### La liste des techniques de réduction de la variance n'est pas close

Les techniques de réduction de la variance que nous allons présenter recouvrent la très grande majorité de celles que vous allez rencontrer en pratique. Mais gardez à l'esprit que ce n'est pas une liste close : n'importe quelle manipulation, si elle améliore l'efficacité, est une réduction de la variance, qu'elle fasse partie de ce cours ou non.

# Chapitre 4

# L'échantillonnage préférentiel

## 4.1 Principe de l'échantillonnage préférentiel

! **Théorème (EP).** *Soit $P$ une mesure de probabilité sur $\Omega$ et $f$ une variable aléatoire réelle intégrable sous $P$. Soit $Q$ une autre mesure de probabilité sur $\Omega$ telle que $P$ soit à densité par rapport à $Q$ ; alors :*

$$\mathbb{E}_P(f) = \mathbb{E}_Q\big((\mathrm{d}P/\mathrm{d}Q) \times f\big). \tag{EQ}$$

♢

*Démonstration.* Cela résulte immédiatement de la définition de la densité. ♤

! *Remarque* (ER). En réalité, l'espérance de $f$ n'est pas forcément une espérance sur $\Omega$ tout entier, mais seulement sur la partie où $f$ est non nulle ; par conséquent, on peut dans le théorème (EP) n'exiger que $P$ n'ait une densité par rapport à $Q$ que sur $\{\omega \in \Omega \mid f(\omega) \neq 0\}$. ♧

! **Définition (ES).** Lorsque, pour estimer $\mathbb{E}_P(f)$ par la méthode de Monte-Carlo, on emploie le théorème (EP) afin de se ramener à une méthode de Monte-Carlo où on fait les tirages selon la loi $Q$, on dit qu'on utilie la méthode d'*échantillonnage préférentiel*. La loi $Q$ est alors appelée *loi d'échantillonnage*. ♡

! *Remarque* (ET). Pour pouvoir appliquer la méthode d'échantillonnage préférentiel, il faut : (outre la nécessité de savoir calculer $f$),

— Qu'on sache simuler la loi $Q$ ;

— Qu'on sache calculer la densité $\mathrm{d}P/\mathrm{d}Q$. ♧

*Remarque* (EU). Le théorème (EP) ressemble beaucoup au théorème (BY)... En fait, c'est exactement le même ! La seule "différence" provient de ce que, lorsque nous cherchions à évaluer une intégrale de Monte-Carlo, nous n'avions pas de mesure de probabilité *canonique* pour écrire l'intégrale comme une espérance, de sorte que la question portait sur le *choix* d'une mesure de probabilité ; alors que dans l'énoncé ci-dessus, nous disposons *déjà* d'une mesure de probabilité naturelle, de sorte que la méthode d'échantillonnage préférentiel d'apparente plutôt à un *changement* de loi de probabilité. ♧

Nous verrons dans la suite de cette section que l'échantillonnage préférentiel est pertinent lorsque la loi $Q$ donne plus de poids aux $\omega$ qui contribuent le plus à l'espérance de $f$. Mais avant cela, voyons tout de suite deux exemples pour illustrer ce concept essentiel :

*Exemple* (EV) (La centrale nucléaire). Une ingénieure doit certifier le risque qu'une centrale nucléaire subisse un accident majeur à l'occasion d'une certaine opération de maintenance quotidienne. Après modélisation, elle en arrive à l'observation que trois facteurs sont impliqués dans un tel risque, facteurs qu'elle appelle respectivement $X$, $Y$ et $Z$, et que ces trois vecteurs sont distribués selon la loi normale

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \sim \text{Normale}(\vec{m}, \boldsymbol{C}). \tag{EW}$$

(voir la § A.5 pour les détails techniques). L'accident se produit quand les trois facteurs $X$, $Y$ et $Z$ deviennent *tous les trois* positifs ; il faut donc évaluer $\mathbb{P}\big((X, Y, Z)^\mathsf{T} \in \mathbb{R}_+^3\big) =: p$. La norme de certification impose que $p$ soit inférieur à $10^{-10}$.

L'ingénieure décide d'utiliser la méthode de Monte-Carlo pour évaluer $p$. Dans un premier temps, elle écrit un programme "naïf" dont vous trouverez le code sur *Arche*. Avec $10^7$ simulations, elle obtient le résultat suivant :

```
>>> centrale_naif(10000000)
La probabilité d'accident est estimée à 0.000000e+00 ± 0.000000e+00.
(Note : Intervalle de confiance standard à 1,96 sigmas).
```

À première vue, l'affaire semble entendue : il n'y aurait strictement aucun risque. « Mais en fait, réfléchit l'ingénieure, ce résultat est dégénéré : il signifie simplement que toutes les simulations ont été négatives... Or, si le risque véritable est de $10^{-8}$, ce n'est pas en faisant seulement $10^7$ simulations qu'on va pouvoir observer quelque chose !... ». L'ennui, c'est que l'ordinateur poussif de notre ingénieure ne permet guère de dépasser $10^7$ simulations... D'où l'idée d'utiliser une technique d'échantillonnage préférentiel pour obtenir un résultat significatif.

L'ingénieure choisit d'utiliser comme loi d'échantillonnage préférentiel la loi $\boldsymbol{Q} \coloneqq \text{Normale}(\vec{0}, \boldsymbol{C})$ correspondant à un vecteur gaussien de même variance, mais centré au point $(0, 0, 0)^\mathsf{T}$. La véritable loi $\boldsymbol{P}$ du vecteur gaussien a bien une densité par rapport à $\boldsymbol{Q}$, qui est effectivement calculable, comme le montre la § A.5. On arrive alors au code suivant (voir les fonctions auxiliaires en annexe) :

```
# La fonction "centrale_pref" applique la méthode d'échantillonnage
# préférentiel pour calculer le risque d'accident.
def centrale_pref(Nsim):
    m = array([[-3.0], [-4.0], [-5.0]])
    C = array([[1, .1, .2], [.1, 1, .3], [.2, .3, 1]])
    L = cholesky(C)
    # "Cinv" est la matrice inverse de C, utilisée pour le calcul de la
    # densité.
    Cinv = inv(C)
    # La variable dont on estime l'espérance n'étant plus une indicatrice, il
    # faut maintenir séparément la somme et la somme des carrés.
    somme_f = 0
    somme_f2 = 0
    for i in range(Nsim):
        # Cette fois-ci on échantillonne XYZ selon la loi Q.
        XYZ = aux.tirer_Q(L)
        # La fonction dont on évalue l'espérance doit tenir compte de la
        # densité.
        f = (XYZ[0, 0] >= 0 and XYZ[1, 0] >= 0 and XYZ[2, 0] >= 0) * aux.densitePQ(m, Cinv, XYZ)
```

```
        somme_f = somme_f + f
        somme_f2 = somme_f2 + f * f
    moy = somme_f / Nsim
    var_f = somme_f2 / Nsim - moy * moy
    ect_estr = sqrt (var_f / Nsim)
    print('La probabilité d\'accident est estimée à',
          '{:e} ± {:e}.'.format(moy, 1.96 * ect_estr))
    print('(Note : Intervalle de confiance standard à 1,96 sigmas).')
    return
```

L'exécution donne :

```
>>> centrale_pref(5000000)
La probabilité d'accident est estimée à 4.608046e-11 ± 3.192084e-13.
(Note : Intervalle de confiance standard à 1,96 sigmas).
```

Cette fois-ci le résultat n'est plus dégénéré, ce qui nous rassure sur la fiabilité de l'intervalle de confiance. On trouve que les accidents arriveront avec une probabilité d'environ $5 \times 10^{-11}$, ce qui permet donc de certifier la centrale. ♣

*Remarque* (EX). Dans l'exemple ci-dessus, on remarquera que la complexité accrue des calculs a rendu le temps de calcul simulation environ deux fois plus lent dans la version préférentielle que dans la version naïve. De ce fait, nous n'avons pu procéder qu'à deux fois moins de simulations ; mais cela n'empêche pas le résultat d'être incomparablement meilleur ☺ ♣

! *Remarque* (EY). Il est essentiel de bien comprendre que la méthode d'échantillonnage préférentiel est une méthode d'échantillonnage *sous* $\boldsymbol{Q}$, *pour la fonction « fonction d'intérêt multipliée par densité »*. Ainsi, si nous notons $f$ la fonction d'intérêt et $\rho$ la densité $\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q}$, la variance qui doit intervenir dans le calcul de l'intervalle de confiance est $\mathrm{Var}_{\boldsymbol{Q}}(\rho f)$, et non pas $\mathrm{Var}_{\boldsymbol{Q}}(f)$ ou $\mathrm{Var}_{\boldsymbol{P}}(f)$... ! ♣

*Exemple* (EZ). Le concept d'échantillonnage préférentiel étant particulièrement important, nous allons en donner un second exemple, cette fois-ci dans un cadre discret. — Lorsque les probabilités $\boldsymbol{P}$ et $\boldsymbol{Q}$ portent sur un espace discret, $\big(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q}\big)(\omega)$ est simplement $\boldsymbol{P}(\{\omega\})/\boldsymbol{Q}(\{\omega\})$. Nous considérons à nouveau la problème du championnat de basket, mais cette fois-ci nous supposons que l'équipe de Nancy est la n° 13 et a donc très peu de chances de remporter le championnat. Dans ces conditions, l'échantillonnage selon la loi de probabilité "naturelle" (notée $\boldsymbol{P}$) n'est sans doute pas le plus adapté, car on tombe très rarement sur des victoires finales de Nancy... Nous proposons alors la loi d'échantillonnage préférentiel suivant, notée $\boldsymbol{Q}$ :

> Sous la probabilité $\boldsymbol{Q}$, les matchs n'impliquant pas Nancy se déroulent normalement, mais les matchs impliquant Nancy ont leur résultat tiré au sort *comme si Nancy jouait avec le niveau de l'équipe n° 0*. (À part cela, les résultats des différents matchs sont toujours indépendants).

La densité de $\boldsymbol{P}$ par rapport à $\boldsymbol{Q}$ peut être, là encore, calculée explicitement : l'annexe A.1 montre comment.

On obtient alors le code suivant :

```
# La fonction "basket_pref" estime la probabilité de victoire de l'équipe de
# Nancy, qui porte ici le numéro 13, par la méthode de rejet (pour la loi
# d'échantillonnage préférentiel définie dans le polycopié). "NSIM" est le
```

```
# nombre de simulations demandées. Je ne commente que les différences avec le
# code de basket_13.
def basket_pref():
    NSIM = 10000
    NANCY = 13
    # La probabilité que nous cherchons ne s'écrivant plus comme l'espérance
    # d'une indicatrice, nous devons maintenir séparément la somme et la somme
    # des carrés.
    somme_f = 0
    somme_f2 = 0
    for i in range(NSIM):
        # La fonction "championne_Q" à laquelle nous allons faire appel
        # fonctionne comme "championne", mais a deux différences importantes
        # près: 1. La loi de simulation est la loi d'échantillonnage Q; 2. La
        # fonction renvoie non seulement qui est la championne, mais aussi
        # quelle est la densité dP/dQ au point simulé. On stocke respectivement
        # ces deux valeurs de retour dans "ch" et "densite".
        simulation = aux.championne_Q()
        # Nous appelons "f" la réalisation de la fonction dont on va prendre
        # l'espérance.
        f = (simulation['ch'] == NANCY) * simulation['densite']
        somme_f += f
        somme_f2 += f * f
    moy = somme_f / NSIM
    var_f = somme_f2 / NSIM - moy * moy
    ect_estr = (var_f / NSIM) ** (1 / 2)
    print('La probabilité de victoire de Nancy est',
          '{:f} ± {:f} %.'.format(100 * moy, 100 * 1.96 * ect_estr))
    return
```

À l'exécution :

```
>>> topchrono = time.process_time(); basket_pref(); print('Temps de calcul :',
... '{:f} s.'.format(time.process_time() - topchrono))
La probabilité de victoire de Nancy est 0.491505 ± 0.031006 %.
Temps de calcul : 2.163371 s.
```

Si nous comparons avec le code naïf correspondant (voir sur *Arche*) :

```
>>> topchrono = time.process_time(); basket_13(); print('Temps de calcul :',
... '{:f} s.'.format(time.process_time() - topchrono))
La probabilité de victoire pour de Nancy est 0.530000 ± 0.058098 %.
Temps de calcul : 4.537327 s.
```

Là encore, si le nombre de simulations exécutées en un laps de temps donné est plus faible, le gain d'efficacité par simulation est suffisant pour qu'on gagne un facteur environ 8 sur l'efficacité globale. ♧

*Remarque* (FA). Dans l'exemple ci-dessus, il est particulièrement flagrant sur cet exemple que les intervalles de confiance donnés avec ou sans échantillonnage préférentiel sont tout-à-fait compatibles. Cela n'a rien qui doive nous surprendre, dans la mesure où il s'agit de deux méthodes différentes pour calculer la *même* quantité. Pour triviale qu'elle soit, cette remarque est néanmoins utile pour détecter d'éventuels problèmes, soit algorithmiques (erreur dans le calcul de la densité, ...), soit pratique (sous-échantillonnage, ...). ♧

## 4.2 Choix de l'univers d'intégration

Passer du principe général de l'échantillonnage préférentiel à sa mise en pratique recèle un certain nombre de subtilités qui sont assez déroutantes au début. Une de ces subtilités consiste à déterminer l'espace sur lequel les distributions $\boldsymbol{P}$ et $\boldsymbol{Q}$ portent. Il s'avère en effet qu'il y a à priori plusieurs choix possibles concernant l'espace jouant le rôle de « $\Omega$ » dans le théorème (EP), que ce choix a une importance cruciale pour pouvoir mettre en pratique l'échantillonnage préférentiel, et qu'il n'est pas trivial... ! Nous allons ci-dessous consacrer deux études de cas à expliquer la façon dont nous avons tranché ce point pour les exemples respectifs de la centrale nucléaire et du tournoi de basket.

### Étude de cas : La centrale nucléaire

Notre méthode de Monte-Carlo pour le problème de la centrale nucléaire, en l'absence d'échantillonnage préférentiel, passe par la simulation du triplet $(X, Y, Z)$ ; celui-ci étant décrit par un loi normale multivariée, nous l'avons simulé par la méthode de Cholesky : à savoir, nous avons l'abord tiré trois v.a.i.i.d. normales standard $N_0, N_1, N_2$, puis nous avons ensuite défini $(X, Y, Z)$ comme une fonction affine de ces variables (à savoir : $(X; Y; Z) \coloneqq \boldsymbol{L}\cdot(N_0; N_1; N_2)+\vec{m}$, où $\boldsymbol{L}$ est la matrice de Cholesky de $\boldsymbol{C}$). (Dans le cas de la loi d'échantillonnage $\boldsymbol{Q}$, le principe général est le même, sauf qu'on ne tient pas compte de $\vec{m}$ pour simuler $X, Y, Z$).

Ainsi, nos simulations "de base" sont les v.a. $N_0, N_1, N_2$. En termes de formalisme, il est donc naturel de considérer que l'univers $\Omega$ sur lequel porte notre tirage aléatoire est celui des triplets $(n_0, n_1, n_2) \in \mathbb{R}^3$, qu'on munit cet univers de la loi de probabilité Normale$(0, 1)^{\otimes 3}$, et que $X, Y, Z$ sont des v.a. obtenues à partir de $N_0$, $N_1$, $N_2$ par les transformations affines susmentionnées. Cette interprétation de notre simulation est certes correcte... mais elle ne permet pas de passer à l'échantillonnage préférentiel !

En effet, lorsqu'on s'intéresse à l'échantillonnage préférentiel, on a besoin que nous deux contextes de simulation correspondent à la simulation de la *même* variable aléatoire, définie sur le *même* univers probabiliste, mais muni de deux lois *différentes*. Or, lorsque nous regardons comment on simule l'occurrence d'une panne sous la loi d'échantillonnage préférentiel, ce n'est pas cela que nous faisons : en effet, dans notre simulation préférentielle, nous tirons $(N_0, N_1, N_2)$ sous la *même* loi que précédemment, mais en lui appliquant ensuite une transformation *différente* pour en déduire $(X, Y, Z)$ ! Il est donc impossible, en l'occurrence, de considérer cette l'interprétation 'naïve' de l'univers probabiliste $\Omega$ pour appliquer la technique de l'échantillonnage préférentiel à notre problème.

Au lieu de cela, notre démarche a été la suivante : nous avons considéré que les éventualités de notre univers probabiliste (ce que la formule (EQ) aurait appelé '$\omega$') étaient *directement* les triplets $(x, y, z)$ décrivant les variables $X, Y, Z$, et que, selon le cas, on utilisait l'une ou l'autre loi pour déterminer ces triplets ! Et cette fois-ci, la variable aléatoire $f$ dont on détermine l'espérance correspond bien à la même chose dans les deux contextes de simulation : en effet, c'est l'indicatrice du fait que $x$, $y$ et $z$ soient tous les trois positifs. En outre, grâce aux formules pour la densité des vecteurs gaussiens, nous étions capables de connaitre la densité relative entre les deux lois de $(X, Y, Z)$ : ainsi, nous avions bien toutes les cartes en main pour d'appliquer la technique d'échantillonnage préférentiel ! Ainsi, pour transcrire notre

cadre formel (EQ) dans ce contexte concret, nous avons considéré que l'éventualité fondamentale « $\omega$ » n'était pas constituée par les tirages fondamentaux de notre code, mais par une quantité *intermédiaire* intervenant dans notre simulation. C'est un peu déroutant, mais mathématiquement c'est parfaitement correct... !

Très bien ; nous comprenons donc mieux comment nous avons interprété le cadre formel dans notre implémentation. Mais on peut alors se demander : « pourquoi s'être arrêté "au milieu du gué" et avoir considéré que le rôle de l'éventualité $\omega$ était joué par le triplet $(x, y, z)$ : n'aurait-on pas pu aller jusqu'au bout et décider que l'éventualité $\omega$ est carrément le fait qu'il y ait une panne ou non ?! ». Après tout, en effet, tout semble bien réuni pour appliquer l'échantillonnage préférentiel : dans les deux cas l'indicatrice de l'accident vit dans le même espace (à savoir $\{0, 1\}$), dans les deux cas, nous lui faisons subir la même transformation (à savoir, aucune !) pour déterminer la variable $f$ dont on prend l'espérance avant de calculer l'espérance ; et en ce qui concerne la densité, déterminer la densité relative entre deux variables de Bernoulli de paramètres différents est trivial, puisque ce sont des lois discrètes dont les fonctions de masse ont des expressions simples... Alors, pourquoi donc ne pas avoir opté pour cette interprétation-là ?!

Eh bien, c'est parce qu'en réalité, cela n'aurait pas marché ! Car, certes, « déterminer la densité relative entre deux variables de Bernoulli de paramètres différents est trivial »... à condition de connaitre les paramètres des deux lois de Bernoulli en question ! Or ici, nous ne connaissons *pas* ces paramètres, puisqu'ils ont été définis seulement *implicitement*, comme la probabilité que certaines lois normales multivaluées tombent dans un certain octant. Ainsi, pas de formule explicite pour déterminer la densité relative, et donc pas d'échantillonnage préférentiel possible avec cette interprétation de notre cadre formel... !

Nous voyons donc que, parmi les trois choix qui pouvaient sembler naturels pour interpréter concrètement le théorème (EP), le seul choix fonctionnel que nous avions pour déterminer la variable « $\boldsymbol{\omega}$ » était celui que nous avons effectivement fait : les deux autres n'auraient pas marché !

## Étude de cas : Le championnat de basket

Toujours autour de la question de déterminer le choix approprié pour '$\Omega$', regardons maintenant ce qu'il en est dans le cadre du basket. Dans notre implémentation de l'échantillonnage préférentiel, nous avons considéré que la variable aléatoire fondamentale était constituée par le résultat (au sens de « identité du vainqueur ») de tous les matchs du championnat. Ce résultat peut être considéré comme à valeurs dans $\Omega \coloneqq \{0, 1\}^E$, où $E$ est l'ensemble des matchs, de cardinal 120 : c'est cet $\Omega$ qui joue le rôle de notre univers dans le cadre de l'échantillonnage préférentiel. Ce contexte est effectivement adapté à la méthode d'échentillonnage préférentiel : en effet, quelle que soit la loi d'échantillonnage utilisée, l'univers probabiliste est le même, et le fait que Nancy remporte le titre ou pas (ou le nombre de lauréats entre qui elle doit partager le titre) se déduit de l'éventualité $\omega$ de la même façon dans les deux cas ; et d'autre part, on sait calculer la densité relative entre $P$ et $Q$, comme nous l'avons montré dans l'annexe A.1, en utilisant que, dans les deux cas, l'éventualité $\boldsymbol{\omega}$ s'obtient comme une suite de v.a. de Bernoulli indépendantes de paramètres explicites, ce qui permet d'obtenir la densité globale par produit des densités individuelles.

De même que dans le cas de la centrale nucléaire, on peut se demander si

d'autres choix n'auraient pas été également possibles ici. Une première observation est que l'univers $\Omega$ "naïf" qui intervient à la base de notre simulation standard, à supposer que nous ayons obtenu nos v.a. de Bernoulli par la méthode de la fonction quantile, n'est pas $\{0,1\}^E$ lui-même, mais $[0,1]^E$. Cependant, la façon dont nous transformons notre tirage dans $[0,1]^E$ en v.a. de Bernoulli, puis en résultat final du championnat, dépend de la loi d'échantillonnage (via les paramètres des lois de Bernoulli), donc cela n'aurait pas pu convenir pour l'échantillonnage préférentiel.

Nous aurions pu aussi nous demander si l'on ne pouvait pas utiliser des variables intermédiaires "plus avancées" pour notre calcul de densité relative : par exemple, on aurait pu considérer comme éventualité de base le nombre de victoires totales marquées par chaque équipe (auquel cas l'univers probabiliste aurait été quelque chose comme $[\![0,15]\!]^{16}$)... Du point de vue de la formule (EQ), cela ne pose pas de souci : en effet, on a bien le même univers probabiliste sous les deux lois d'échantillonnage, et la façon de déduire l'identité de la vainqueure dépend bien des nombres de victoires de la même façon dans les deux cas. Mais c'est au niveau du calcul de la densité relative qu'on aurait eu un souci, car on n'a pas de façon de déterminer explicitement la densité de la loi sur les nombres totaux de victoires, dont l'expression est très compliquée... Bref ; à nouveau, nous n'avions en réalité guère le choix pour déterminer notre éventualité de base en vue de l'échantillonnage préférentiel.

*Remarque* (FB). Finalement, dans les deux exemples ci-dessus, il n'y avait qu'un seul choix réellement possible pour définir l'univers probabiliste et ses éventualités de base. Ce n'est néanmoins pas toujours le cas : par exemple, imaginons que note simulation consiste à considérer une v.a. $X$ de loi $\mathrm{Expon}^{\mathrm{le}}(\lambda)$, et que nous nous intéressions plus spécifiquement à l'indicatrice $Y := \mathbf{1}_{X \geqslant T}$ que $X$ dépasse une certain seuil $T$[*] ; et dans la loi d'échantillonnage préférentiel, nous conservons la même valeur de $T$ mais changeons la valeur de $\lambda$ en $\lambda'$. Il y a alors *deux* choix pour le rôle de l'éventualité élémentaire : on peut faire jouer ce rôle, soit à $X$, soit à $Y$! En effet, d'une part $Y$ se déduit bien de la même façon de $X$ dans les deux cas (et aussi, trivialement, de $Y$) ; et d'autre part, on a une formule explicite pour la densité de $X$ (que ce soit sous la loi de base ou sous la loi d'échentillonnage préférentiel) aussi bien que pour celle de $Y$! (qui est une Bernoulli($\mathrm{e}^{-\lambda}$) dans le cas de base, resp. une Bernoulli($\mathrm{e}^{-\lambda'}$) sous l'échantillonnage préférentiel). Donc les deux choix sont valables.

Dans ce cas, la règle à suivre est de systématiquement privilégier, comme éventualité élémentaire, la v.a. qui intervient en *fin* de chaine, i. e. elle qui est une fonction des autres ! En l'occurrence, ce serait donc $Y$ qu'il est meilleur d'utiliser comme éventualité élémentaire.

La raison de ce choix est qu'utiliser la v.a. située en fin de chaine est en fait équivalent à utiliser la v.a. de début de chaine en y rajoutant une technique de *conditionnement* : or le conditionnement se traduit (presque) toujours par un gain d'efficacité appréciable (comme nous le verrons en § 5.1) et doit donc systématiquement être privilégié ! ♧

[*]. Indicatrice qu'on suppose utilisée à son tour dans d'autres calculs aboutissant finalement à la fonction dont on souhaite calculer l'espérance.

## 4.3 Calcul des densités relatives dans les cas compliqués

À ce stade, pour calculer une densité relative dans le cadre d'un échantillonnage préférentiel, vous devriez être tentés de vouloir toujours procéder de la façon suivante :

1. On détermine l'espace $\Omega$ sur lequel portent $P$ et $Q$, en priant très fort pour que ce soit un espace discret ou (un sous-ensemble d')un espace de la forme $\mathbb{R}^d$ ;
2. On détermine, sur cet espace, les densités relatives de $P$ et $Q$ par rapport à une même mesure de référence (la mesure de comptage ou la mesure de Lebesgue, selon le cas) ;
3. On fait le quotient de ces deux densités pour trouver $\mathrm{d}P/\mathrm{d}Q$.

Le problème de cette approche, c'est qu'elle ne fonctionne que si $\Omega$ est discret ou de la forme $\mathbb{R}^d$ ! Dans les deux exemple que nous avons traités ci-dessus, c'était effectivement le cas, mais ce ne le sera pas toujours... En outre, dans le cas du championnat de basket, nous avons pu nous apercevoir que les formules obtenues pour les densités de $P$ et $Q$ étaient assez lourdes, alors qu'ensuite elles se simplifiaient ensuite nettement quand on prenait leur quotient : cela suggère qu'on est passé à côté de quelque chose qui aurait pu être plus pratique...

Nous allons donc présenter ci-dessous une approche à la fois nettement plus commode et nettement plus général pour calculer les densités ☺

### Principe général

La méthode de calcul des densités relatives que nous allons présenter s'applique au cadre, très général en pratique, où l'éventualité $\omega$ que nous simulons peut être vue comme une *suite* (de longueur finie) de sous-éventualités, où, *conditionnellement à la valeur dans sous-éventualités précédentes*, chacune des sous-éventualités suit une loi "simple" sur un espace "simple".

Un exemple vous permettra de mieux comprendre à quoi cela peut ressembler en pratique :

*Exemple* (FC). Considérons la situation de simulation suivante. Des paramètres $\lambda$ et $\pi$ ayant été fixés au préalable, on tire successivement des variables aléatoires indépendantes $X_0, X_1, X_2, \ldots$, avec $X_i$ suivant la loi $\mathrm{Expon}^{\mathrm{le}}\big((i+1)\lambda\big)$, jusqu'à ce que la somme des variables tirées atteigne ou dépasse 1. On note alors $N$ le nombre de variables tirées ; et on tire ensuite une v.a. entière $J$ selon la loi $\mathrm{Géom}^{\mathrm{que}}(\pi)\wedge(J-1)$.

Cela donne le code suivant (les paramètres `LAMBDA` et `PI` étant supposés prédéfinis), où j'ai fait appel aux fonctions de simulation `exponential` et `geometric` préimplémentées dans le module `numpy.random`, afin de nous focaliser sur la structure de simulation en tant que telle :

```
from numpy.random import exponential, geometric

les_x = list() # Contiendra la liste des X_i simulés
somme_x = 0.0 # Somme des X_i
i = 0 # Indice considéré
while True:
    x = exponential(1 / ((i + 1) * LAMBDA)) # Simulation de X_i
    les_x.append(x) # Stockage dans la liste
```

```
    somme_x += x # Mise à jour de la somme
    # On regarde si la somme atteint ou dépasse 1
    if somme_x >= 1.0:
        break
    else:
        i += 1 # On passe à l'indice suivant

n = len(les_x) # Nombre de X_i simulées
# Tirage de J...
j = geometric(PI) - 1
# Troncature éventuelle
if j >= n:
    j = n - 1

tuple(les_x + [j])
```

La dernière ligne de ce code sert simplement à afficher, vu comme un tuple, l'ensemble des variables aléatoires successives que nous avons simulées (à savoir : $X_0, X_1, \dots, X_{N-1}, J$). Bien entendu, en pratique, le but ne serait pas d'*afficher* ces simulations, mais de s'en servir pour fournir, par exemple, la valeur de $X_J$ (qu'on renverrait par « `les_x[j]` ». Cependant, dans tous les cas, on pourrait dire que l'aléa *fondamental* simulé par notre programme est constitué par le tuple $(X_0, \dots, X_{N-1}, J)$, et que la valeur $X_J$ finalement renvoyée est une *fonction* déterministe de cet aléa fondamental.

Dans cet exemple, appelons $\boldsymbol{\omega} =: (X_0, \dots, X_{N-1}, J)$ le tuple que nous avons simulé au niveau fondamental. Manifestement, la loi de la variable aléatoire $\boldsymbol{\omega}$ est particulièrement compliquée : notamment, à une même position du tuple, on peut avoir soit un réel, soit un entier (selon que la somme des valeurs précédentes est inférieure à 1 ou pas), et la longueur elle-même du tuple varie selon les valeurs tirées... ! Néanmoins, au niveau de notre code, nous remarquons que, *au moment* de tirer chaque élément supplémentaire de la simulation fondamentale (autrement dit, *sachant* ce qui a *déjà* été tiré jusque-là), la loi de l'élément tiré suivant est très simple : c'est soit une loi exponentielle, soit une loi géométrique tronquée ! ♧

Nous allons formaliser cela plus rigoureusement. Ne vous laissez pas dérouter par la complexité du formalisme : il s'agit ni plus ni moins que d'une traduction (dans un cadre plus général) de l'exemple que nous venons de voir ! ☺

**Définition (FD)** (Espace récursivement simple)**.** On dit qu'un espace $\Omega$ est *récursivement simple*[†] lorsque ses éventualités $\omega$ s'écrivent comme une suite de composantes $(\omega_i)_{i\in\mathbb{N}}$, et que, notant $\Omega_{<k}$ l'ensemble des débuts de taille $k$ possibles d'éléments de $\Omega$ (i. e., $\Omega_{<k} \coloneqq \{(\omega_i)_{0\leqslant i<k} \mid \omega \in \Omega\}$), et notant, pour tout $\vec{\omega}_{<k} \in \Omega_{<k}$ :

$$\mathscr{E}_k(\vec{\omega}_{<k}) \coloneqq \{\omega_k \mid (\vec{\omega}_{<k}, \omega_k) \in \Omega_{k+1}\} \tag{FE}$$

(autrement dit, $\mathscr{E}_k(\vec{\omega}_{<k})$ est l'ensemble des valeurs par lesquelles ont peut étendre $\vec{\omega}_{<k}$ en un élément de $\Omega_{k+1}$), tous les $\mathscr{E}_k(\vec{\omega}_{<k})$ sont soit des espaces discrets, soit des (sous-ensembles d')espaces de la forme $\mathbb{R}^d$ ($d$ pouvant dépendre de $\vec{\omega}_{<k}$).

En outre, nous envisageaons la possibilité que certaines éléments de $\Omega$ ne contiennent en fait qu'un nombre *fini* de composantes. Pour ce faire, nous introduisons une valeur spéciale '$\partial$' qui servira à dire « la suite des composantes est

[†]. Ne pas utiliser cette nomenclature en-dehors du cadre de ce cours : il s'agit d'un terme ad hoc que j'ai créé uniquement pour les besoins spécifiques de la proposition (FK).

terminée », et on assimilera une suite finie $(\omega_0, \dots, \omega_{n-1})$ à la suite (indexée par $\mathbb{N}$) $(\omega_0, \dots, \omega_{n-1}, \partial, \partial, \dots)$ qu'on obtient en lui adjoignant une infinité de valeurs $\partial$. Bien entendu, cela sous-entend que, une fois qu'une composante vaut $\partial$, toutes les composantes suivantes vaudront $\partial$ également : autrement dit, que $\mathscr{E}_k(\vec{\omega}_{<k}) = \{\partial\}$ dès lors que $\omega_{k-1} = \partial$.

Pour $\omega \in \Omega$, la longueur de $\omega$, à valeurs dans $\mathbb{N} \sqcup \{\infty\}$, sera notée $\mathrm{len}(\omega)$. (Dans la version avec le symbole spécial $\partial$, cela correspond à l'infimum des $k$ tels que $\omega_k = \partial$). ♡

Sur les espaces récursivement simples, on peut définir des distributions de probabilité que nous appellerons *lois par densités récursives*

**Définition (FF)** (Loi par densités récursives)**.** Une distribution de probabilité $P$ sur un espace récursivement simple $\Omega$ est dite *par densités récursives*[‡] lorsque, avec les notations ci-dessus, on a pour tout $k \in \mathbb{N}$, pour tout $\vec{\omega}_{<k} \in \Omega_{<k}$, que la loi conditionnelle

$$\mathrm{Loi}^{\omega \sim P}(\omega_k \mid \vec{\omega}_{<k} = \vec{\omega}_{<k}) \tag{FG}$$

est à densité par rapport à la mesure de référence sur $\mathscr{E}_k(\vec{\omega}_{<k})$ (par « mesure de référence », j'entends : mesure de comptage si $\mathscr{E}_k$ est discret, resp. mesure de Lebesgue si c'est un ouvert de $\mathbb{R}^d$). Cette densité sera alors notée $f_k^{(\omega_{<k})}(\bullet) : \mathscr{E}_k(\vec{\omega}_{<k}) \to \mathbb{R}_+$.

En outre, dans le cadre de ce cours, nous ajouterons l'hypothèse que $P$ ne charge que les suites de longueur finie :

$$P(\{\omega \in \Omega \mid \ell(\omega) = \infty\}) = 0. \tag{FH}$$

♡

*Remarque* (FI)*.* Les densités récursives $f_k^{(\omega_{<k})}(\bullet)$ permettent en fait de caractériser entièrement la loi $P$. ♧

Ce formalisme étant posé, nous pouons enfin en venir à la proposition sur le calcul des densités relatives. En voici d'abord une version informelle :

**Principe (FJ)** (Calcul d'une densité relative dans le cas par densités successives)**.** *Supposons que $P$ et $Q$ soient toutes les deux des mesures de probabilité par densités récursives, définies sur le* même *espace $\Omega$, mais avec des densités récursives différentes selon le cas. Alors, lorsqu'on considère une valeur $\omega \in \Omega$ (de longueur finie), la densité en $\omega$ de $P$ par rapport à $Q$ s'obtient en considérant le produit des densités successives entre les lois de probabilité « simples » successivement utilisées pour simuler les différentes composantes de $\omega$.* ◇ !

Une version plus rigoureuse de cet énoncé est la suivante :

**Théorème (FK).** *Supposons que $P$ et $Q$ soient toutes les deux des mesures de probabilité par densités récursives, définies sur le* même *espace $\Omega$, mais avec des densités récursives différentes selon le cas, notées resp. $f_k^{\omega_{<k}}(\bullet)$ et $g_k^{\omega_{<k}}(\bullet)$. Supposons en outre qu'on ait $g_k^{(\omega_{<k})} \neq 0$ partout où $f_k^{(\omega_{<k})} \neq 0$. Alors $P$ admet une densité par rapport à $Q$, qui s'exprime comme le produit suivant : pour tout $\omega \in \Omega$ (de longueur finie),*

$$\frac{P(\mathrm{d}\omega)}{Q(\mathrm{d}\omega)} = \prod_{k=0}^{\mathrm{len}(\omega)-1} \frac{f_k^{(\vec{\omega}_{<k})}(\omega_k)}{g_k^{(\vec{\omega}_{<k})}(\omega_k)}. \tag{FL}$$

◇

[‡]. Encore une notation ad hoc, NdA.

*Remarque* (FM). À cause de sa forme en produit, on ne va jamais écrire explicitement la formule pour la densité relative entre $\boldsymbol{P}$ et $\boldsymbol{Q}$ : on calculera simplement le produit à mesure que la simulation de $\boldsymbol{\omega}$ progressera ! ☺ ♧

Afin de mieux comprendre le schmilblick, illustrons le théorème ci-dessus en reprenant la situation de l'exemple (FC) :

*Exemple* (FN). Nous définissons $\boldsymbol{P}$ comme dans l'exemple (FC) ; et nous considérons aussi une loi $\boldsymbol{Q}$ qui a la même forme que $\boldsymbol{P}$, mais avec des valeurs différentes pour les paramètres, les paramètres de $\boldsymbol{Q}$ étant notés $\lambda'$ et $\pi'$. Remarquons déjà que, effectivement, $\boldsymbol{P}$ et $\boldsymbol{Q}$ portent bien sur le même espace récursivement simple $\boldsymbol{\Omega}$ : en particulier, dans les deux cas c'est bien le fait de dépasser 1 qui compte pour arrêter de tirer des variables continues, et dans les deux cas la variable discrète est bien à valeurs dans l'ensemble des indices des variables déjà tirées.

Maintenant, en ce qui concerne le quotient $f_i^{(\vec{\omega}_{<i})}(\omega_i) \,/\, g_i^{(\vec{\omega}_{<i})}(\omega_i)$, on distingue deux cas :

— Lorsque (sachant ce qui précède) on est en train de tirer une variable continue, $\boldsymbol{\omega}_i$ suit la loi $\mathrm{Expon}^{\mathrm{le}}\big((i+1)\lambda\big)$ sous $\boldsymbol{P}$, resp. la loi $\mathrm{Expon}^{\mathrm{le}}\big((i+1)\lambda'\big)$ sous $\boldsymbol{Q}$. Or les densités respectives de ces deux lois sont $x \mapsto \mathbf{1}_{x>0}\lambda \mathrm{e}^{-(i+1)\lambda x}$ et $x \mapsto \mathbf{1}_{x>0}\lambda' \mathrm{e}^{-(i+1)\lambda' x}$ (on remarquera que, comme exigé, la seconde de ces fonctions est non nulle partout où la première l'est), donc dans ce cas,

$$\frac{f_i^{(\vec{\omega}_{<i})}(\omega_i)}{g_i^{(\vec{\omega}_{<i})}(\omega_i)} = \frac{\lambda}{\lambda'} \mathrm{e}^{(i+1)\cdot(\lambda'-\lambda)\omega_i}. \tag{FO}$$

— Lorsqu'on est en train de tirer la variable discrète, il faut considérer le quotient des fonctions de masse des lois géométriques tronquées :

$$\frac{f_n^{(\vec{\omega}_{<n})}(\omega_n)}{g_n^{(\vec{\omega}_{<n})}(\omega_n)} = \frac{\mathbb{P}\big(\mathrm{Géom}^{\mathrm{que}}(\pi) \wedge (n-1) = \omega_n\big)}{\mathbb{P}\big(\mathrm{Géom}^{\mathrm{que}}(\pi') \wedge (n-1) = \omega_n\big)} = \begin{cases} \frac{\pi}{\pi'}\big(\frac{1-\pi}{1-\pi'}\big)^{\omega_n} & \text{pour } \omega_n < n-1\,; \\ \big(\frac{1-\pi}{1-\pi'}\big)^{n-1} & \text{pour } \omega_i = n-1. \end{cases} \tag{FP}$$

Voici le code qui met ce calcul en pratique : ici on simule toujours $\boldsymbol{\omega}$ sous la loi $\boldsymbol{P}$, mais outre la valeur simulée de $\omega_{\checkmark}$, on renvoie aussi la densité $\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q}$ en $\omega_{\checkmark}$ :

```
from numpy.random import exponential, geometric

# La densité sera calculée par produits successifs :
# initialement le produit est vide, correspondant à une valeur de 1
densite = 1.0
les_x = list()
somme_x = 0.0
i = 0
while True:
    x = exponential(1 / ((i + 1) * LAMBDA))
    # Mise à jour de la densité
    densite *= (LAMBDA / LAMBDAPRIME *
                exp((i + 1) * (LAMBDAPRIME - LAMBDA) * x))
    les_x.append(x)
    somme_x += x
    if somme_x >= 1.0:
```

```
        break
    else:
        i += 1

n = len(les_x)
j = geometric(PI) - 1
if j >= n:
    j = n - 1

# Mise à jour de la densité suite au tirage de J
if j < n - 1:
    densite *= PI / PIPRIME * ((1 - PI) / (1 - PIPRIME)) ** j
else:
    densite *= ((1 - PI) / (1 - PIPRIME)) ** j

print('Simulation :')
print(tuple(les_x + [j]))
print('Densité :')
print(densite)
```

♧

## 4.4 Recherche d'une bonne loi d'échantillonnage préférentiel

La technique d'échantillonnage préférentiel nous laisse le choix de la probabilité d'échantillonnage $\boldsymbol{Q}$ — sous réserve évidemment que nous sachions simuler $\boldsymbol{Q}$ et calculer $\mathrm{d}\boldsymbol{Q}/\mathrm{d}\boldsymbol{P}$. Puisque la technique d'échantillonnage repose sur l'espoir qu'échantillonner selon $\boldsymbol{Q}$ sera plus efficace qu'échantillonner selon $\boldsymbol{P}$, c'est qu'on s'attend à ce que certaines $\boldsymbol{Q}$ soient meilleures que d'autres... Mais lesquelles ? Y a-t-il des heuristiques pour trouver quelles sont les meilleures lois d'échantillonnage $\boldsymbol{Q}$ ? C'est à cette problématique que cette sous-section se propose de répondre.

De manière générale, l'efficacité d'une simulation dépend non seulement de la variance de la quantité dont on calcule l'espérance, mais aussi du cout de calcul par simulation. Cependant, ce dernier est difficile à modéliser mathématiquement, sans compter qu'il est susceptible de dépendre des détails de l'implémentation et même de la machine utilisée... Nous nous limiterons donc ici à essayer d'optimiser l'efficacité *par simulation*, dans l'espoir que cela aidera aussi à améliorer l'efficacité tout court.

**Proposition (FQ).** *L'efficacité par simulation de la technique d'échantillonnage préférentiel est égale à l'inverse de* $\left(\mathbb{E}_{\boldsymbol{P}}(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q} \times f^2) - \mathbb{E}_{\boldsymbol{P}}(f)^2\right)$. ◇

*Démonstration.* L'efficacité par simulation est par définition l'inverse de la variance $\mathrm{Var}_{\boldsymbol{Q}}(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q} \times f)$, qui est égale à $\mathbb{E}_{\boldsymbol{Q}}\left((\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q})^2 \times f^2\right) - \mathbb{E}_{\boldsymbol{Q}}(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q} \times f)^2 = \mathbb{E}_{\boldsymbol{P}}(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q} \times f^2) - \mathbb{E}_{\boldsymbol{P}}(f)^2$ d'après la propriété caractéristique de la densité $\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q}$. ♤

**Théorème (FR).** *L'efficacité par simulation par la méthode d'échantillonnage préférentiel est maximale quand la densité* $\mathrm{d}\boldsymbol{Q}/\mathrm{d}\boldsymbol{P}$ de $\boldsymbol{Q}$ par rapport à $\boldsymbol{P}$ *est proportionnelle à* $|f|$. ◇

*Démonstration.* Notons $\rho \coloneqq \mathrm{d}\boldsymbol{Q}/\mathrm{d}\boldsymbol{P} = (\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q})^{-1}$ (on fera comme si les mesures $\boldsymbol{P}$ et $\boldsymbol{Q}$ étaient équivalentes, ce qui justifie cette expression). D'après la proposition (FQ), notre objectif alors est de minimiser $\mathbb{E}_{\boldsymbol{P}}(\rho^{-1} f^2)$, et ce sous la contrainte que $\boldsymbol{Q}$

soit une mesure de probabilité, c.-à-d. que $\rho \geqslant 0$ et $\mathbb{E}_P(\rho) = 1$. Faisons ici comme si $f$ était non nulle presque-partout ; alors il est clair le $\rho$ optimal vérifiera $\rho^* > 0$ presque-partout, car sinon on aurait $\mathbb{E}_P((\rho^*)^{-1} f^2) = +\infty$ ; de sorte que nous n'avons qu'à considérer la contrainte $\mathbb{E}_P(\rho) = 1$. On applique la méthode du multiplicateur de Lagrange : pour un paramètre $\lambda$ à fixer, on cherche un $\rho^*$ vérifiant $\rho^* \geqslant 0$ et $\mathbb{E}_P(\rho^*) = 1$ tel que $\rho^*$ soit un point critique de la fonctionnelle $\rho \mapsto \mathbb{E}_P(\rho^{-1} f^2) + \lambda\, \mathbb{E}_P(\rho) =: F(\rho)$. On calcule que $(DF(\rho)) \cdot h = \mathbb{E}_P\big((-\rho^{-2} f^2 + \lambda) h\big)$, de sorte qu'un point critique $\rho^*$ doit vérifier $-(\rho^*)^{-2} f^2 + \lambda \equiv 0$, d'où $\rho^* \propto |f|$. ♤

! *Remarque* (FS). En pratique, la probabilité $\boldsymbol{Q}$ optimale, même si elle est simulable, ne peut pas être utilisée car on ne sait pas calculer exactement $\mathrm{d}\boldsymbol{Q}/\mathrm{d}\boldsymbol{P}$. Ce qu'il faut retenir est plutôt qu'on doit privilégier une loi d'échantillonnage $\boldsymbol{Q}$ dont la densité par rapport à $\boldsymbol{P}$ soit « aussi proportionnelle que possible » à $|f|$. En particulier, s'il existe un ensemble restreint d'éventualités $\omega$ pour lesquelles $f$ prend une valeur particulièrement grande, il faudra chercher une loi de simulation $\boldsymbol{Q}$ qui donne plus de poids à ces éventualités-là ! ♧

! *Remarque* (FT). Il y a deux façons de mal "coller" à la probabilité d'échantillonnage optimale : la première est le *suréchantillonnage*, qui consiste à simuler beaucoup trop souvent des zones où la valeur de $|f|$ est plutôt petite ; la seconde, à l'inverse, est le *sous-échantillonnage*, qui consiste à simuler beaucoup trop rarement des zones où la valeur de $|f|$ est plutôt grande. Ces deux défauts n'ont pas du tout la même gravité : *le sous-échantillonnage est beaucoup plus grave que le suréchantillonnage !* On le voit par exemple dans le théorème (FQ), où ce qui est susceptible de faire exploser $\mathbb{E}_P(\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q} \times f^2)$ est que $\mathrm{d}\boldsymbol{P}/\mathrm{d}\boldsymbol{Q}$ soit très grande (autrement dit que $\mathrm{d}\boldsymbol{Q}/\mathrm{d}\boldsymbol{P}$ soit très petite) à un endroit où $f^2$ est très grande également. Dans certains cas, le sous-échantillonnage peut même nous faire perdre le caractère $\mathrm{L}^2$ de la quantité à intégrer. Pire encore : un échantillonnage très marqué, correspondant à l'oubli pratiquement complet d'une zone contribuant à l'espérance de $f$, peut conduire à un estimateur apparemment cohérent... mais en fait faux car il ne tient pas compte de la zone sous-échantillonnée ! Il est donc essentiel, chaque fois qu'on veut appliquer la technique d'échantillonnage préférentiel, de se demander d'abord « ne suis-je pas en train de sous-échantillonner gravement certaines zones ? ». ♧

*Exemple* (FU). Soient $\boldsymbol{P}^*$ et $\boldsymbol{P}_*$ deux mesures de probabilité sur un espace $\Omega$ se décomposant comme une union disjointe $\Omega_0 \cup \Omega_1$. On suppose que $\boldsymbol{P}^*$ (qui sera la mesure *sur*-échantillonnée) donne la masse $\varepsilon^*$ à $\Omega_1$ et $(1 - \varepsilon^*)$ à $\Omega_0$, et que $\boldsymbol{P}_*$ (qui sera la mesure *sous*-échantillonnée) leur donne quant à elle respectivement les masses $\varepsilon_*$ et $(1 - \varepsilon)_*$ ; et on suppose que $\boldsymbol{P}_*$ et $\boldsymbol{P}^*$ ont des densités l'une par rapport à l'autre qui sont constantes sur chaque $\Omega_i$. On prend ici $\varepsilon_* \ll \varepsilon^* \ll 1$, de sorte que les mesures $\boldsymbol{P}^*$ et $\boldsymbol{P}_*$ sont pratiquement identiques sur $\Omega_0$ (la densité relative $\mathrm{d}\boldsymbol{P}^*/\mathrm{d}\boldsymbol{P}_*$ valant $(1 - \varepsilon^*)/(1 - \varepsilon_*) \simeq 1$), mais très différentes sur $\Omega_1$.

Que se passe-t-il alors si nous utilisons la mesure d'échantillonnage $\boldsymbol{P}^*$ là où c'est $\boldsymbol{P}_*$ qui aurait été la plus adaptée ; autrement dit si nous essayons d'utiliser la loi d'échantillonnage préférentiel $\boldsymbol{P}^*$ pour évaluer $\mathbb{E}_{P_*}(1)$ ? L'efficacité par simulation est alors l'inverse de

$$\mathbb{E}_{P_*}(\mathrm{d}\boldsymbol{P}_*/\mathrm{d}\boldsymbol{P}^*) - 1 = (1 - \varepsilon_*) \times \frac{1 - \varepsilon_*}{1 - \varepsilon^*} + \varepsilon_* \times \frac{\varepsilon_*}{\varepsilon^*} - 1 \simeq \varepsilon^*. \qquad (\mathrm{FV})$$

À l'inverse, en échantillonnant avec $\boldsymbol{P}_*$ là où $\boldsymbol{P}^*$ aurait été plus adaptée, on tombe sur une efficacité par simulation inverse de

$$\frac{(1 - \varepsilon^*)^2}{1 - \varepsilon_*} + \frac{(\varepsilon^*)^2}{\varepsilon_*} - 1 \simeq (\varepsilon^*)^2 \,/\, \varepsilon_*, \qquad (\mathrm{FW})$$

ce qui est beaucoup plus grand (et donc beaucoup moins bon) que $\varepsilon^*$ ! ♧

*Remarque* (FX). Historiquement, c'est l'idée d'échantillonnage préférentiel qui a véritablement lancé la méthode de Monte-Carlo, celle-ci ayant conduit à des améliorations spectaculaires du calcul de certaines quantités physiques correspondant à des évènements de probabilités très faibles. ♧

# Chapitre 5

# Méthodes de réduction de la variance sans changement de probabilité

## 5.1 Conditionnement

### Principe général

! **Définition (FY).** Soient $\Omega$ un espace mesurable dont on note $\mathscr{B}$ la tribu naturelle, et $P$ une loi de probabilité sur $\Omega$ ; et soit $f\colon \Omega \to \mathbb{R}$ une fonction telle que $f(P)$ soit d'intégrabilité $\mathrm{L}^2$. Supposons qu'on s'intéresse à évaluer $\mathbb{E}^{\omega\sim P}(f(\omega)) =: \mu$ par méthode de Monte-Carlo.

La *technique de réduction de la variance par conditionnement* repose alors sur le principe suivant. Considérons une certains sous-tribu $\mathscr{A} \subseteq \mathscr{B}$. Sur l'espace probabilité $(\Omega, \mathscr{B}, P)$, la variable aléatoire [*] $\omega \mapsto f(\omega)$ est $\mathrm{L}^2$-intégrable, et donc en particulier $\mathrm{L}^1$ : elle admet donc une espérance conditionnelle par rapport à la sous-tribu $\mathscr{A}$, correspondant à une certaine fonction $\mathscr{A}$-mesurable de $\Omega$ dans $\mathbb{R}$ : notons $\omega \mapsto f^{\mathscr{A}}(\omega)$ cette fonction. On a alors, en vertu des propriétés générales de l'espérance conditionnelle, que $\mathbb{E}^{\omega\sim P}(f^{\mathscr{A}}(\omega)) = \mathbb{E}^{\omega\sim P}(f(\omega)) = \mu$. L'idée de la réduction de la variance par conditionnement consiste alors à estimer $\mu$ *en tant qu'espérance de* $f^{\mathscr{A}}(P)$, càd. qu'on va simuler des réalisations i.i.d. $\omega_0, \dots, \omega_{n-1} \sim P$ et estimer $\mu$ par

$$\hat{\mu}^{\mathrm{cond}} := n^{-1} \sum_{i=0}^{n-1} f^{\mathscr{A}}(\omega_i). \tag{FZ}$$

♡

! *Remarque* (GA). Bien entendu, pour que cette technique de conditionnement soit applicable, il y a besoin qu'on soit en mesure de *calculer effectivement*, de façon numériquement "exacte" [†], l'espérance conditionnelle $f^{\mathscr{A}}(\omega)$ pour une valeur $\omega \in \Omega$ donnée ! ♣

[*]. Ici, une fois n'est pas coutume, je parle de variable aléatoire au sens *technique* du terme, càd. en tant que fonction $\mathscr{B}$-mesurable définie sur $\Omega$ !

[†]. J'entends par là que le calcul de $f^{\mathscr{A}}(\omega)$ ne doit pas présenter d'erreur autre que celle due aux arrondis machine : en particulier, ce calcul ne devra pas s'appuyer lui-même sur une méthode de Monte-Carlo, qui ne serait pas une méthode « exacte » !

## Le conditionnement en pratique

La définition (FY) étant à première vue assez abstraite, un exemple va nous aider à mieux la comprendre :

*Exemple* (GB). Reprenons l'exemple du Keno que nous avions vu dans l'exemple (BV) [‡]. De la façon dont nous avons procédé dans notre code, l'aléa de chaque simulation consiste en la concaténation de vingt étapes successives : on tire d'abord le booléen qui dit si le numéro coché pour 1 point est, puis on tire le booléen qui dit si le numéro coché pour 2 points est sorti, etc.

En fait, du point de vue abstrait, déterminer l'aléa par étapes successives consiste à se placer du point de vue d'une tribu de plus en plus fine. Voyons cela plus en détail. Soit $\Omega \coloneqq \{\texttt{VRAI}, \texttt{FAUX}\}^{[\![1,20]\!]}$ l'univers (qui est ici discret) dans lequel vit notre aléa, une éventualité de cet aléa étant vue comme un 20-uplet $(\omega_1, \dots, \omega_{20}) =: \vec{\omega}_{[\![1,20]\!]}$, où $\omega_1$ dit si le numéro coché pour 1 point est sorti, $\omega_2$ dit si le numéro coché pour 2 points est sorti, etc. Plaçons-nous, au cours de la simulation de la réalisation $\vec{\omega}_{[\![1,20]\!]\checkmark}$ de notre aléa, au moment où nous venons d'en déterminer les $k$ premières composantes de $\omega_{1\checkmark}, \dots, \omega_{k\checkmark}$, les composantes $\omega_{(k+1)\checkmark}$ à $\omega_{20\checkmark}$ n'ayant pas encore été simulées. « Connaitre les $k$ premières composantes de $\vec{\omega}_{[\![1,20]\!]\checkmark}$ », c'est la même chose que « être capable, pour tout $A \in \mathscr{F}_k$, de dire si on a ou non $\{\vec{\omega}_{[\![1,20]\!]\checkmark} \in A\}$ », où $\mathscr{F}_k$ est la sous-tribu suivante de $\mathscr{P}(\Omega)$ :

$$\mathscr{F}_k \coloneqq \left\{ \bar{A} \times \{\texttt{VRAI}, \texttt{FAUX}\}^{[\![k,20]\!]} \;\middle|\; \bar{A} \subseteq \{\texttt{VRAI}, \texttt{FAUX}\}^{[\![1,k]\!]} \right\}. \tag{GC}$$

On peut donc considérer, en termes informels, que « tirer les $k$ premières composantes de $\boldsymbol{\omega}$, c'est déterminer $\boldsymbol{\omega}$ relativement à la sous-tribu $\mathscr{F}_k$ ».

Ici, la sous-tribu que nous allons considérer est $\mathscr{F}_{19}$, qui donne l'information relative aux numéros cochés pour 1 à 19 points, mais pas celle relative au numéro coché pour 20 points. Notant $gain(\omega)$ le gain correspondant à une éventualité $\omega$, nous nous posons alors la question : à quoi correspond, dans le contexte de notre simulation, l'espérance conditionnelle $\mathbb{E}(gain(\boldsymbol{\omega}) \mid \mathscr{F}_{19})$ ? Il ne serait sans doute pas évident de tout écrire dans le formalisme rigoureux des probabilités et des tribus ; néanmoins on comprend très bien ce qui se passe : au bout des 19 premières étapes de simulation, on connait le 19-uplet $(\omega_{1\checkmark}, \dots, \omega_{19\checkmark})$ qui dit quels numéros cochés entre 1 et 19 sont sortis ; de sorte qu'on peut dire combien de ces numéros sont sortis (valeur que nous noterons $q_{19\checkmark}$), et quel est le score dû à ces numéros (valeur que nous noterons $m_{19\checkmark}$) [§]. Nous savons alors que, conditionnellement à l'information sur les résultats des 19 premiers numéros cochés, il y a une probabilité $(20-q_{19\checkmark})/61$ que le vingtième numéro sorte, auquel cas on aura un gain valant $gain(m_{19\checkmark} + 20)$, et une probabilité $(41 + q_{19\checkmark}) / 61$ qu'il ne sorte pas, auquel cas le gain sera de $gain(m_{19\checkmark})$. Il est alors immédiat de déterminer l'espérance pour une telle situation à seulement deux issues possibles : en fin de compte, nous avons donc déterminé que

$$\mathbb{E}(gain(\vec{\boldsymbol{\omega}}_{[\![1,20]\!]}) \mid \vec{\boldsymbol{\omega}}_{[\![1,19]\!]} = \vec{\omega}_{[\![1,19]\!]\checkmark}) = \frac{(20 - q_{19\checkmark}) \times gain(m_{19\checkmark} + 20) + (41 + q_{19\checkmark}) \times gain(m_{19\checkmark})}{61}. \tag{GD}$$

[‡]. Ici, je supposerai déjà implémentée l'amélioration de la méthode de simulation aléatoire évoquée dans l'exemple (EM).

[§]. En termes formels, on a resp. $q_{19\checkmark} \coloneqq \sum_{k=1}^{19} \mathbf{1}_{\omega_{k\checkmark}}$ et $m_{19\checkmark} \coloneqq \sum_{k=1}^{19} \mathbf{1}_{\omega_{k\checkmark}} k$.

Le membre de droite de (GD) correspond à ce que, dans le formalisme de la définition (FY), nous avions noté "$f^{\mathscr{A}}(\omega_{\checkmark})$" [où "$f(\bullet)$" correspond à la fonction qui dit quel gain on obtient selon quels numéros entre 1 et 20 points sont sortis, et $\mathscr{A}$ correspond à $\mathscr{F}_{19}$] : c'est (la réalisation de) l'espérance conditionnelle de notre gain sachant l'information relative à la sous-tribu $\mathscr{F}_{19}$. Comme prévu, cette variable aléatoire "$f^{\mathscr{A}}(\boldsymbol{\omega})$" est bien $\mathscr{F}_{19}$-mesurable, puisqu'on n'a besoin que de $q_{19\checkmark}$ et $m_{19\checkmark}$ pour déterminer sa valeur, et que celles-ci sont connues dès lors qu'on a l'information relative à $\mathscr{F}_{19}$ ☺

En fin de compte, appliquer la réduction de la variance par conditionnement relativement à la tribu $\mathscr{F}_{19}$ nous amènera à simuler des valeurs "$f^{\mathscr{A}}(\boldsymbol{\omega})$" de la façon suivante :

1. Simuler quels sont les numéros sortis parmis ceux cochés de 1 à 19 points (inutile de regarder si le numéro coché à 20 points est sorti) ;
2. Calculer les valeurs correspondantes de $q_{19\checkmark}$ et $m_{19\checkmark}$ ;
3. Utiliser la formule (GD) pour en déduire la réalisation correspondant de '$f^{\mathscr{A}}(\omega_{\checkmark})$'.

♣

*Remarque* (GE). Nous voyons donc que, dans cet exemple du Keno, nous avons commencer à simuler notre aléa de Monte-Carlo normalement, mais que, une fois l'information relative à $\mathscr{F}_{19}$ connue, nous avons arrêté notre simulation aléatoire, dans la mesure où nous connaissions *déjà* l'espérance *exacte* sachant ce que nous avions déjà simulé : dès lors, autant utiliser directement ce résultat exact, vu que celui-ci sera impacté par l'aléa que si nous avions poursuivi le tirage aléatoire jusqu'au bout, et nous permettra ainsi, en toute logique, d'obtenir un résultat plus précis ! (Et de fait, c'est effectivement le cas : cela sera confirmé par le théorème (GG) un peu plus loin).

En l'occurrence, on peut donc résumer la philosophie du conditionnement ainsi : ne pas utiliser Monte-Carlo quand on a accès à un résultat exact ! ♣

*Remarque* (GF). Cet exemple du Keno participe de la situation la plus fréquente dans lequel on est amené à utiliser le conditionnement, à savoir : lorsque notre simulation produit l'aléa récursivement à partir de lois plus simples (comme expliqué dans la § sec:densites-relatives.cas-complique relative au calcul des densités relatives dans un tel cas), on peut procéder en interrompant la procédure de simulation récursive à une certaine phase, et utiliser le conditionnement par la tribu décrivant l'information obtenue à l'issue de cette phase-là. (Sous réserve, bien entendu, qu'on sache calculer exactement l'espérance conditionnelle de notre fonction). Il peut certes exister quelques autres situations dans lesquelles des formes de conditionnement plus originales peuvent faire sens ; et nous en verrons peut-être certaines en exercices : néanmoins, c'est avant tout le cas présenté dans la remarque (GF) qui est le plus fréquent en pratique, et qui est celui dans lequel la technique de conditionnement est à la fois la plus naturelle et la plus immédiate à mettre en œuvre : c'est donc ce cas que vous devez savoir maitriser en priorité ! ☺ ♣

### Intérêt du conditionnement

Une propriété très sympathique de la réduction de la variance par conditionnement est que la variance de la variable conditionnée est *toujours* inférieure (ou égale) à celle de la variance conditionnée, et que donc, lorsqu'on est en mesure

d'appliquer le conditionnement, cela va *toujours* se traduire par un gain d'efficacité par simulation :

**Théorème (GG).** *L'efficacité par simulation de l'estimateur conditionné est toujours meilleure que celle de l'estimateur non conditionné.* ◇ !

*Démonstration.* L'efficacité par simulation de l'estimateur non conditionné est l'inverse de la variance de $f$, tandis que celle de l'estimateur conditionné est l'inverse de la variance de $f^{\mathscr{B}}$.

Par l'inégalité de Jensen, $\mathbb{E}((f^{\mathscr{B}})^2) \leqslant \mathbb{E}(f^2)$, et comme en outre on a $\mathbb{E}(f^{\mathscr{B}}) = \mathbb{E}(f)$, il s'ensuit que $\mathrm{Var}(f^{\mathscr{B}}) = \mathbb{E}((f^{\mathscr{B}})^2) - \mathbb{E}(f^{\mathscr{B}})^2 \leqslant \mathbb{E}(f^2) - \mathbb{E}(f)^2 = \mathrm{Var}(f)$, ce qui prouve que l'efficacité par simulation est effectivement meilleure pour $f^{\mathscr{B}}$ que pour $f$. ♤

En outre, en général, non seulement on va gagner sur l'efficacité par simulation, mais on va *aussi* gagner sur le cout de calcul par simulation, en vertu de l'argument suivant :

*Remarque* (GH). Lorsqu'on est dans le cas expliqué à la remarque (GF), où le conditionnement consiste à s'interrompre en cours de route d'une simulation récursive, alors la partie « simulation » de chaque passage dans la boucle de Monte-Carlo sera un peu plus *simple* que dans le cas conditionné ! Dans ces conditions, le cout de calcul par simulation sera (normalement [¶]) plus *faible* dans l'algorithme avec conditionnement que dans l'algorithme naïf !

Sachant que, comme nous vu juste au-dessus, l'efficacité par simulation est quant à elle *toujours* améaliorée par le conditionnement, on peut en conclure (et en retenir) que, en pratique, il ne faut jamais hésiter à implémenter une technique de réduction de variance par conditionnement quand on en voit la possibilité ! Cela est, au demeurant, parfaitement cohérent avec la philosophie exposée dans la remarque (GE) *supra* : il n'y a pas lieu de faire du Monte-Carlo quand on dispose d'une méthode de calcul exacte ! ♡ ♧

## Complément : Bien se repérer dans la notion d'espérance conditionnelle

À ce stade, j'ai dit ce qu'il y avait besoin de savoir sur la réduction de la variance par conditionnement dans le cadre de ce cours. Il m'a néanmoins semblé utile d'ajouter ci-dessous quelques remarques autour de la notion technique d'espérance conditionnelle — qui, comme vous l'avez vu, est au cœur de cette méthode. En effet, il se trouve que la notion d'espérance conditionnelle est particulièrement subtile à traiter dans le cadre général de la théorie des probabilités (notamment dans le cas où on considère des lois de probabilité diffuses), ce qui peut rendre difficile de faire la connexion entre la formalisme général de l'espérance conditionnelle et nos intuitions d'ingénieur : j'espère donc que les quelques remarques à venir vous rendront cette connexion plus facile ! ☺

*Remarque* (GI). Il faut garder à l'esprit qu'une espérance conditionnelle, disons $\mathbb{E}(f \mid \mathscr{A})$ (pour $f$ une v.a. réelle et $\mathscr{A}$ une sous-tribu), est un objet "hybride" :

[¶]. J'ai écrit « normalement », car la phrase peut devenir fausse dans les cas où, au sein de la boucle de Monte-Carlo, la partie « évaluer $f^{\mathscr{A}}(\omega)$ » se met à devenir si complexe (par rapport à l'évaluation de $f(\omega)$ dans l'algorithme non conditionné) que cela compenserait le gain de cout dû à la simplification de la partie « simulation de $\omega$ » : cependant, dans la plupart des codes de Monte-Carlo, c'est bien la partie « simulation » qui est responsable de la majorité du cout de calcul, et non pas la partie « évaluation de la fonction de l'aléa » : de sorte que gagner au niveau de la simulation se traduit « normalement » par une diminution du cout de chaque passage dans la boucle ☺

— D'un côté, c'est une *variable aléatoire*, qui décrit "la meilleure approximation de $f$ qui soit $\mathscr{A}$-mesurable". En tant que variable aléatoire, on peut donc parler de sa loi, de son espérance, etc. Cet aspect « variable aléatoire » est rendu plus clair par la notation "$f^{\mathscr{A}}$" : avec une telle notation, quand on écrit quelque chose comme "$\mathrm{Var}(f^{\mathscr{A}})$", on comprend qu'on parle de la variance de la variable aléatoire $f^{\mathscr{A}}$, ce qui est passablement plus clair qu'une expression comme $\mathrm{Var}\big(\mathbb{E}(f \mid \mathscr{A})\big)$, me semble-t-il.
— Mais d'un autre côté, quand on raisonne *conditionnellement à la tribu* $\mathscr{A}$ (voir à ce sujet la remarque (GJ) ci-après), la notion d'« espérance conditionnelle » correspond simplement à la notion d'espérance, et décrit donc une *constante* ! Cette fois-ci, la notation classique "$\mathbb{E}(f \mid \mathscr{A})$" est la plus adaptée à rendre cette idée : par exemple, on comprend très facilement l'inégalité conditionnelle de Jensen en l'écrivant ($\varphi(\bullet)$ désignant une fonction convexe) sous la forme « $\mathbb{E}(\varphi(f) \mid \mathscr{A}) \geqslant \varphi\big(\mathbb{E}(f \mid \mathscr{A})\big)$ » !

Il y a donc besoin de jongler entre ces deux visions lorsqu'on traite d'espérance conditionnelle, ce qui n'est pas particulièrement intuitif... ♧

*Remarque* (GJ). La locution « espérance conditionnelle » semble suggérer qu'il s'agit de l'espérance par rapport à une certaine loi. De fait, c'est bien le cas : pour $X$ une variable aléatoire sur $\Omega$ (à valeurs dans un espace quelconque $\mathscr{X}$) et $\mathscr{A}$ une sous-tribu de $\mathscr{P}(\Omega)$, on peut définir la *loi conditionnelle de $X$ sachant $\mathscr{A}$* de la façon suivante : il s'agit d'une application de $\Omega$ dans l'ensemble $\mathscr{M}_1(\mathscr{X})$ des *lois de probabilités* sur $\mathscr{X}$, qui, en tant qu'application de $\Omega$ dans $\mathscr{M}_1(\mathscr{X})$, est $\mathscr{A}$-mesurable, et qui est telle que, pour n'importe quelle fonction (bornée) $f\colon \mathscr{X} \to \mathbb{R}$, l'espérance conditionnelle de $f(X)$ (qui est une fonction de $\Omega$ dans $\mathbb{R}$) coïncide, en chaque valeur $\omega$, avec l'espérance de $f\big(\mathrm{Loi}(X \mid \mathscr{A})\big)$, où ici « $\mathrm{Loi}(X \mid \mathscr{A})$ » doit être vue comme la loi de probabilité sur $\mathscr{X}$ qui est l'image, en $\omega$, de la fonction « loi conditionnelle ».

Comme vous le voyez, la définition est passablement compliquée [‖], ce qui explique que, en pratique, on se limite à la notion d'espérance conditionnelle dans les cours de M1. Néanmoins, pour peu qu'on en comprenne au moins l'intuition (confer remarque (GK) ci-après), la notion de loi conditionnelle s'avère extrêmement puissante : elle permet en effet de parler, par ricochet, de densité conditionnelle, de variance conditionnelle, de quantiles conditionnels, de façon complètement naturelle ; et ces concepts peuvent s'avérer fort utiles dans de nombreux cas ! ◡̈ ♧

*Remarque* (GK). Dans cette section, j'ai parlé à plusieurs reprises de l'idée de « raisonner conditionnellement à la sous-tribu $\mathscr{A}$ », en soulignant qu'elle éclairait particulièrement bien notre intuition, notamment vis-à-vis de la notion de loi conditionnelle (confer remarque (GJ)). Je vais expliquer plus en détail ce qu'on entend par là dans la cas où $\Omega$ est discret, ou tout peut être rendu parfaitement rigoureux.

Lorsque $\Omega$ est dénombrable, la tribu naturelle de $\Omega$ est l'ensemble $\mathscr{P}(\Omega)$ tout entier des parties de $\Omega$, et une sous-tribu $\mathscr{A}$ de cette tribu est simplement une sous-ensemble de $\mathscr{P}(\Omega)$ vérifiant certaines propriétés de stabilité, notamment celle d'être stable par intersection dénombrable. Or, dans un tel cas, on peut définir la notion d'« être équivalent relativement à $\mathscr{A}$ » de la façon suivante : deux éventualités $\omega$, $\omega' \in \Omega$ sont équivalentes relativement à $\mathscr{A}$ si et seulement si, pour tout $A \in \mathscr{A}$, soit $\omega$ et $\omega'$ appartiennes toutes les deux à $A$, soit elles sont toutes les deux en-dehors de $A$. Grâce à cette notion d'équivalence, on peut créer ce qu'on appelle une *partition* de $\Omega$ relativement à la tribu $\mathscr{A}$ : il s'agit de découper $\Omega$ en une famille de parties

[‖]. Sans parler que j'ai caché certaines subtilités sous le tapis... ☿

non vides $(\mathfrak{A}_i)_{i\in I}$, telles que tout élément de $\Omega$ appartienne à une et une seule des $\mathfrak{A}_i$, et telle que deux éléments appartiennent au même $\mathfrak{A}_i$ si et seulement si ils sont équivalents relativement à $\mathscr{A}$. Et j'affirme que, dans le cas où $\Omega$ est discret, un sous-ensemble $B \subseteq \Omega$ appartient à $\mathscr{A}$ si et seulement si il peut s'écrire commune l'union de certains $\mathfrak{A}_i$. Ainsi, « considérer une sous-tribu de $\mathscr{P}(\Omega)$ », c'est la même chose que « considérer une partition de $\Omega$ ».

Dans ce cas, lorsqu'on se place en un certain $\omega \in \Omega$, l'idée de « raisonner conditionnellement à $\mathscr{A}$ » signifie qu'on raisonne conditionnellement au fait d'être dans $\mathfrak{A}(\omega)$, où $\mathfrak{A}(\omega)$ est le bloc de la partition $(\mathfrak{A}_i)_{i\in I}$ qui contient $\omega$ (ou, ce qui est équivalent, $\mathfrak{A}$ est l'ensemble des éléments de $\Omega$ qui sont équivalents à $\omega$ relativement à $\mathscr{A}$). Et « conditionner par rapport à la tribu $\mathscr{A}$ », cela signifie simplement : « au niveau de $\omega$, conditionner par l'évènement $\mathfrak{A}(\omega)$ ».

Par exemple, si $f$ est une fonction bornée de $\Omega$ dans $\mathbb{R}$, on sait bien que, pour $A$ un évènement, $\mathbb{E}(f \mid A)$ n'est autre que l'espérance de la loi conditionnellement à la réalisation de $A$ :

$$\mathbb{E}(f \mid A) = \frac{\sum_{\omega\in A} f(\omega)\,\mathbb{P}(\omega)}{\sum_{\omega\in A} \mathbb{P}(\omega)}. \tag{GL}$$

Eh bien, l'espérance conditionnelle par rapport à la tribu $\mathscr{A}$, c'est la même chose, sauf qu'on a une *fonction* de $\Omega$ dans $\mathbb{R}$ (que je noterai $f^{\mathscr{A}}$), et que la valeur de cette fonction en $\omega$ est obtenue en remplaçant, dans la formule (GL), '$A$' par "$\mathfrak{A}(\omega)$" :

$$f^{\mathscr{A}}(\omega) = \frac{\sum_{\omega'\sim_{\mathscr{A}}\omega} f(\omega')\,\mathbb{P}(\omega')}{\sum_{\omega'\sim_{\mathscr{A}}\omega} \mathbb{P}(\omega')}. \tag{GM}$$

Dans les situations où l'univers $\Omega$ n'est pas discret, cette vision de choses n'est malheureusement plus valide formellement, et on est alors obligé de passer par des définitions bien plus indirectes ; néanmoins, l'intuition reste la même ! ☺ ♧

## 5.2 Couplage [**]

**Théorème (GN).** *Soient $(\Omega_1, \mathbb{P}_1), (\Omega_2, \mathbb{P}_2)$ des espaces probabilisés (munis de leurs tribus standard) ; soient $f_1 : \Omega_1 \to \mathbb{R} \in \mathrm{L}^1(\mathbb{P}_1), f_2 : \Omega_2 \to \mathbb{R} \in \mathrm{L}^1(\mathbb{P}_2)$. Supposons qu'il existe un espace probabilisé $(\Omega, \mathbb{P})$ et des applications $\pi_1 : \Omega \to \Omega_1$ et $\pi_2 : \Omega \to \Omega_2$ telles que $\pi_{1*}\mathbb{P} = \mathbb{P}_1$ et $\pi_{2*}\mathbb{P} = \mathbb{P}_2$. Alors : (notant resp. $\mathbb{E}_1, \mathbb{E}_2, \mathbb{E}$ les espérances relatives aux probabilités $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}$),*

$$\mathbb{E}_2(f_2) - \mathbb{E}_1(f_1) = \mathbb{E}(f_2 \circ \pi_2 - f_1 \circ \pi_1). \tag{GO}$$

♢

*Démonstration.* Trivial à partir de la linéarité de l'espérance et de la définition de l'espérance. ♤

**Définition (GP).** Lorsque, pour estimer une différence d'espérances par la méthode de Monte-Carlo, on commence par modifier les calculs faits en appliquant le théorème (GN), on dit qu'on utilise la *méthode de couplage*. ♡ !

**Théorème (GQ).** *Supposons que $f_1$ soit dans $\mathrm{L}^2(\mathbb{P}_1)$, resp. $f_2$ dans $\mathrm{L}^2(\mathbb{P}_2)$, et que sous la loi $\mathbb{P}$, les variables aléatoires $f_1 \circ \pi_1$ et $f_2 \circ \pi_2$ soient positivement* !

[**]. En anglais : *Common random numbers.*

*corrélées*[††]. *Alors l'estimateur de Monte-Carlo pour* $(f_1 \circ \pi_1 - f_2 \circ \pi_2)$ *sous* $\mathbb{P}$, *avec* $n$ *simulations, a une variance (effective) moins grande que la différence entre l'estimateur de Monte-Carlo pour* $f_1$ *sous* $\mathbb{P}_1$ *et l'estimateur de Monte-Carlo pour* $f_2$ *sous* $\mathbb{P}_2$, *chacun avec* $n$ *simulations.* ◇

*Démonstration.* Notons $\hat{\mu}_1$ l'estimateur de Monte-Carlo (naïf) de $\mathbb{E}_1(f_1)$, $\hat{\mu}_2$ l'estimateur de Monte-Carlo de $\mathbb{E}_2(f_2)$, de sorte que $\hat{\mu}_1 - \hat{\mu}_2$, où $\hat{\mu}_1$ et $\hat{\mu}_2$ sont indépendants, correspond à l'estimateur sans couplage de $(\mathbb{E}_1(f_1) - \mathbb{E}_2(f_2))$ ; et notons $\hat{\mu}$ l'estimateur de $\mathbb{E}(f_1 \circ \pi_1 - f_2 \circ \pi_2)$, c.-à-d. l'estimateur avec couplage. Il s'agit donc de comparer les variances de $(\hat{\mu}_1 - \hat{\mu}_2)$ d'une part, et de $\hat{\mu}$ d'autre part. L'estimateur du second est tout simplement :

$$\mathrm{Var}(\hat{\mu}) = n^{-1} \mathrm{Var}(f_1 \circ \pi_1 - f_2 \circ \pi_2); \tag{GR}$$

et comme $f_1 \circ \pi_1$ et $f_2 \circ \pi_2$ sont positivement corrélées, en développant la variance, on en déduit que

$$\mathrm{Var}(\hat{\mu}) \leqslant n^{-1}(\mathrm{Var}(f_1 \circ \pi_1) + \mathrm{Var}(f_2 \circ \pi_2)). \tag{GS}$$

Pour l'estimateur non couplé ; dans la mesure où celui-ci s'écrit comme la différence de deux estimateurs *indépendants*, sa variance est la somme des variances individuelles de ces estimateurs, autrement dit :

$$\mathrm{Var}(\hat{\mu}_1 - \hat{\mu}_2) = n^{-1} \mathrm{Var}_1(f_1) + n^{-1} \mathrm{Var}_2(f_2). \tag{GT}$$

Or la loi de $f_1$ sous $\mathbb{P}_1$ est aussi la loi de $f_1 \circ \pi_1$ sous $\mathbb{P}$, et la loi de $f_2$ sous $\mathbb{P}_2$ est aussi la loi de $f_2 \circ \pi_2$ sous $\mathbb{P}$ ; de sorte que

$$\mathrm{Var}(\hat{\mu}_1 - \hat{\mu}_2) = n^{-1}(\mathrm{Var}(f_1 \circ \pi_1) + \mathrm{Var}(f_2 \circ \pi_2)). \tag{GU}$$

Au final, nous en déduisons bien que la variance de l'estimateur couplé est plus petite. ♤

*Remarque* (GV). Dans le théorème (GQ), le résultat s'exprime sous la forme d'une amélioration de l'efficacité *par simulation* ; alors que, comme nous l'avons dit et redit, ce qui est intéressant est de savoir si l'efficacité *tout court* est améliorée. Néanmoins, dans les situations pratiques auxquelles nous serons confrontés, le cout de calcul par simulation est *plus faible* dans la situation couplée que dans la situation non couplée ! En effet, pour la plupart des couplages naturels, simuler $\mathbb{P}$ puis $\pi_1$ n'est guère plus long que de simuler $\mathbb{P}_1$ directement (idem concernant la simulation de $\mathbb{P}_2$ via $\mathbb{P}$ puis $\pi_2$). Du coup, si nous négligeons le cout des étapes "techniques" (calcul de la somme de simulations, de la somme des carrés, de la moyenne, etc. ; et dans le cas présent, calcul de la différence entre les deux fonctions) [‡‡], nous voyons que le cout d'une étape de calcul est, pour la situation couplée, correspond à

— Simulation de $\mathbb{P}$ ;
— Calcul de $\pi_1$ ;
— Calcul de $\pi_2$ ;
— Calcul de $f_1$ ;
— Calcul de $f_2$ ;

[††]. Rappelons que, pour $f_1, f_2 \in \mathrm{L}^2(\mathbb{P})$, dire que des variables sont *positivement corrélées* signifie que $\mathrm{Cov}(f_1, f_2) \geqslant 0$. On peut encore donner du sens à ce concept même dans un cadre non $\mathrm{L}^2$, par exemple : « on dit que $f_1$ et $f_2$ sont positivement corrélées si, lorsque $(f_1, f_2)$ et $(f_1', f_2')$ sont des réalisations indépendantes de la loi jointe du couple de variables aléatoires $(f_1, f_2)$, la variable aléatoire $(f_1 - f_1')(f_2 - f_2')$ est d'espérance positive (la notion d'espérance positive pour une variable aléatoire pouvant être définie dès lors que celle-ci est de partie négative intégrable) ». Avec une telle définition, si $f_2$ est une fonction croissante de $f_1$, $f_1$ et $f_2$ seront toujours positivement corrélées, même en cas de défaut d'intégrabilité.

[‡‡]. En pratique, ces étapes techniques étant très simples par rapport aux étapes de simulation et d'évaluation des fonctions à intégrer, elles ne prennent effectivement qu'une part extrêmement mineure du cout de calcul total.

et pour la situation non couplée, à
— Simulation de $\mathbb{P}_1$, *égale*[*] Simulation de $\mathbb{P}$ *plus* Simulation de $\pi_1$ ;
— Calcul de $f_1$ ;
— Simulation de $\mathbb{P}_2$, *égale* Simulation de $\mathbb{P}$ *plus* Simulation de $\pi_1$.
Le calcul couplé est ainsi (modulo les approximations simplificatrices que nous avons faites) strictement plus court que le calcul couplé — on gagne à peu près le temps de simulation de $\mathbb{P}$ à chaque étape.

Il convient aussi d'observer que, dans les situations pratiques, la possibilité d'appliquer une forme de stratification (de type « à postériori ») (cf. § 5.3) pour l'évaluation naïve de $(\mathbb{E}_1(f_1) - \mathbb{E}_2(f_2))$ n'offre aucune amélioration par rapport à la méthode proposée ici consistant à procéder au même nombre de simulations pour chaque espérance individuelle ; car le temps de simulation de $f_1$ sous $\mathbb{P}_1$ et $f_2$ sous $\mathbb{P}_2$ sera quasiment identique, et $\mathrm{Var}_1(f_1)$ et $\mathrm{Var}_2(f_2)$ seront très proches ; de sorte que l'allocation optimale des calculs correspond bien à faire faire à peu près autant de simulations pour chaque espérance. ♧

*Remarque* (GW). Contrairement à la plupart des autres techniques de réduction de la variance, la possibilité de procéder par couplage ou pas pour améliorer l'efficacité d'une estimation apparait en général de façon assez flagrante. *Les situations qui se prêtent bien au couplage correspondent en effet à celles où on doit estimer la différence entre deux situations qui sont essentiellement les mêmes, ne différant que par un paramètre mineur*, comme dans l'exemple (GX) ci-dessous. Dans ce cas, la façon judicieuse de procéder au couplage est d'utiliser le couplage *naturel* qui existe entre les deux situations ; où par « couplage naturel », j'entends qu'on utilise la *même* source d'aléa d'une situation à l'autre, les paramètres différant n'intervenant qu'aussi discrètement que possible (là encore, cf. exemple (GX)). Dès lors, $f_1 \circ \pi_1$ et $f_2 \circ \pi_2$ se "ressembleront" beaucoup, ce qui fait que la corrélation positive (et même une *forte* corrélation positive) sera quasi-automatiquement assurée. ♧ !

*Exemple* (GX) (Le yahtzee de la mort). Un groupe d'explorateurs a été capturé par une tribu de sauvages. Ceux-ci décident que le sort des explorateurs sera tranché par les dieux de la forêt *via* le jeu du yahtzee. À savoir, les explorateurs disposent de 5 dés équilibrés, et lancent ceux-ci une première fois ; puis, à deux reprises, il ont le droit de relancer tout ou partie (ou aucun) des dés, en choisissant ceux qu'ils veulent relancer en fonction du résultat obtenu. Les primitifs, qui attribuent aux dieux les résultats des dés, libèreront les explorateurs si le total des dés final atteint ou dépasse 24, et les exécuteront sinon. Le jeu ne devant avoir lieu que le lendemain matin, les explorateurs (qui pour leur part croient plus aux probabilités qu'au jugement des dieux) se concertent sur la stratégie optimale à suivre. Or, deux propositions s'opposent au sujet du meilleur choix des dés à relancer : l'une due à Charlotte et l'autre due à Xénia (voir la § A.6 pour les détails). Comment savoir quelle stratégie est la meilleure ? Les explorateurs ayant justement 5 dés sur eux, le plus simple est de profiter de la nuit pour appliquer la méthode de Monte-Carlo "à la main" et évaluer les probabilités de succès respectives avec chaque stratégie.

Voilà à quoi pourrait ressembler le code qui évalue la probabilité de victoire pour la stratégie de, disons, Charlotte :

```
# La fonction "probaC" évalue la probabilité de succès en suivant la
# stratégie de Charlotte.
def probaC():
```

[*]. L'égalité s'entend ici au sujet du cout de calcul.

```
    NSIM = 1200000
    Nsucc = 0
    for i in range(NSIM):
        Nsucc = Nsucc + strategieC()
    p = Nsucc / NSIM
    var_f = p - p * p
    ect_estr = sqrt(var_f / NSIM)
    print('Probabilité de succès pour la stratégie de Charlotte :',
          '{:f} ± {:f}.'.format(p, 1.96 * ect_estr))
    return
```

La stratégie de Charlotte, quant à elle, est simulée par le code `strategieC`. Noter qu'en réalité c'est dans la fonction auxiliaire `choixC` que la stratégie de Charlotte est décrite ; mais nous renvoyons celle-ci à *Arche* dans la mesure où ce code n'a pas d'intérêt du point de vue de la méthode de Monte-Carlo.

```
# La fonction "strategieC" fait une simulation du jeu en suivant la
# stratégie de Charlotte. Elle renvoie True en cas de succes, et False en
# cas d'échec.
def strategieC():
    global NDES, NFACES, SEUIL
    # La variable "des" est un 5-vecteur qui stocke les valeurs des dés devant
    # le joueur. On commence par le premier lancer.
    des = [ceil(NFACES * random()) for j in range(NDES)]
    # On va relancer deux fois les dés, d'où la boucle ci-dessous.
    for i in range(1, NLANCERS):
        # La fonction auxiliaire "choixC" est au centre de la stratégie de
        # Charlotte: elle dit, sachant à quel stade du jeu on en est, pour un
        # ensemble de dés fixés, quels sont les dés que Charlotte choisit de
        # relancer ou pas.
        choix = choixC(i, des)
        # On relance les dés que Charlotte veut relancer.
        for j in range(NDES):
            if choix[j]:
                des[j] = ceil(NFACES * random())
    return sum(des) >= SEUIL
```

Bien entendu, on définirait similairement des fonctions `probaX` et `strategieX` (et `choixX`). Ce qui est intéressant, c'est de voir comment on pourrait coupler ces deux stratégies. Voici un couplage naturel : on imagine qu'il y a une même source d'aléa quelle que soit la stratégie, qui réside dans le résultat du premier lancer de dés, puis dans le résultat du premier dé éventuellement relancé, le résultat du second dé éventuellement relancé, etc. Dans le programme ci-dessous, cette source d'aléa est appelée "source". Notons que nous faisons le choix de ne pas remplir cette source à l'avance, mais seulement en fonction des résultats dont on a besoin. le programme est le suivant :

```
# La fonction "diffprobaCX" évalue la différence des probabilités de succès
# selon qu'on suit la stratégie de Charlotte ou celle de Xénia.
def diffprobaCX():
    NSIM = 120
    total_f = 0
    total_f2 = 0
    for i in range(NSIM):
        # Toute la technique de couplage sera contenu dans la fonction de
        # simulation couplée "diffstrategieCX"
```

```
        f = diffstrategieCX()
        total_f += f
        total_f2 += f ** 2
    moy = total_f / NSIM
    var_f = total_f2 / NSIM - moy * moy
    ect_estr = sqrt(var_f / NSIM)
    print('Différence des probabilités de succès :',
            '{:f} ± {:f}.'.format(moy, 1.96 * ect_estr))
    return

# La fonction "diffstrategieCX" renvoie, dans une situation couplée, +1
# quand la stratégie de Charlotte a réussi mais pas celle de Xénia, -1
# quand c'est l'inverse, et 0 sinon.
def diffstrategieCX():
    # On alloue directement la mémoire pour "source", sachant qu'on aura besoin
    # d'au plus 15 valeurs.
    source = [None for j in range(NLANCERS * NDES)]
    # On commence par regarder ce que donne la stratégie de Charlotte. Les
    # tirages de dés que nous serons amenés à faire seront stockés dans
    # "source".
    des = [None for j in range(NDES)]
    for j in range(NDES):
        des[j] = ceil(NFACES * random())
    # Ce lancer de dés initial constitue évidemment les 5 premières valeurs de
    # la source.
    for j in range(NDES):
        source[j] = des[j]
    # La variable "donnees_puisees" nous indique combien on a déjà tiré de dés
    # dans "source".
    donnees_puisees = NDES
    # On passe aux dés à relancer.
    for i in range(1, NLANCERS):
        choix = choixC(i, des)
        for j in range(NDES):
            if choix[j]:
                # "r" est le résultat du dé supplémentaire tiré.
                r = ceil(NFACES * random())
                des[j] = r
                # On met ce résultat supplémentaire dans la source de données.
                source[donnees_puisees] = r
                donnees_puisees += 1
    # On regarde si l'objectif est atteint et on stocke le résultat dans
    # "resultatC"
    total = 0
    for j in range(NDES):
        total += des[j]
    resultatC = total >= 24
    # On stocke aussi le nombre total de donnees écrites, dans
    # "donnees_ecrites"
    donnees_ecrites = donnees_puisees
    # Maintenant on regarde ce que donne la stratégie de Xénia. Toute l'astuce
    # est qu'on utilisera la source de données déjà utilisée par Charlotte,
    # autant que possible!
    for j in range(NDES):
        des[j] = source[j]
```

```
    donnees_puisees = NDES
    for i in range(1, NLANCERS):
        choix = choixX(i, des)
        for j in range(NDES):
            if choix[j]:
                # Observez comment on va forcer le couplage en allant lire dans
                # "source" quand c'est possible.
                if donnees_puisees < donnees_ecrites:
                    r = source[donnees_puisees]
                    donnees_puisees += 1
                else:
                    r = ceil(NFACES * random())
                des[j] = r
    total = 0
    for j in range(NDES):
        total += des[j]
    resultatX = total >= 24
    return resultatC - resultatX
```

Voyons maitenant en quoi le couplage a amélioré l'efficacité du calcul. Disons pour fixer les idées que la nuit laisse suffisamment de temps aux explorateurs pour procéder à 210 essais pour chaque stratégie. S'ils testent déparément la taux de succès pour les stratégies de Charlotte et de Xénia, on a les résultats suivants :

```
>>> probaC(); probaX()
Probabilité de succès pour la stratégie de Charlotte : 0.552381 ± 0.067254.
Probabilité de succès pour la stratégie de Xénia : 0.514286 ± 0.067599.
```

La différence entre les résultats des deux stratégies n'est pas significative... Si on agrégeait ces deux résultats (obtenus indépendamment) en un unique intervalle de confiance, on obtiendrait que l'avantage de performance de l'idée de Charlotte sur celle de Xénia vaut $0{,}038\,1 \pm 0{,}095\,4$ [†], ce qui ne nous permet pas de nous assurer de sa positivité.

À l'inverse, s'ils utilisent la stratégie de couplage avec 210 essais (ce qui demande en fait *moins* de lancers de dés que de faire 210 essais pour chaque, puisque certains lancers de dés seront utilisés pour les deux stratégies à la fois—même si, en contrepartie, l'algorithme est plus compliqué), ils obtiennent

```
>>> diffprobaCX()
Différence des probabilités de succès : 0.047619 ± 0.036774.
```

On voit que les fluctuations sur l'estimateur sont près de 3 fois plus petites que dans le cas précédent : on a ainsi gagné un facteur 7 environ en efficacité ! En

[†]. Note importante. Puisqu'on a obtenu des estimations de la probabilité de Charlotte et de celle de Xénia de façons *indépendantes*, on contrôle l'erreur sur la différence des performances en disant que la *variance* de l'estimateur de la différence est la somme des *variances* des estimateurs individuels. Dans la mesure où la caractère centré et gaussien des estimateurs des performances de Charlotte et Xénia se transmet à celui de la différence, et que la demi-largeur de l'intervalle de confiance à 5 % est proportionnelle à la racine carrée de la variance, on obtient les fluctuations de l'estimateur différence en faisant la somme *quadratique* des fluctuations individuelles (la somme quadratique de deux nombres positifs $a$ et $b$, que nous noterons ici $a +_2 b$, est le nombre positif tel que $(a +_2 b)^2 = a^2 + b^2$), plutôt que la somme simple. Cela permet un gain substantiel par rapport à l'intervalle de confiance que nous aurions obtenu si nous n'avions pas pu prendre en compte l'indépendance et la gaussianité, puisqu'on aurait alors dû faire la somme simple des fluctuations, et que par-dessus le marché l'intervalle de confiance obtenu n'aurait été à priori valable qu'à un risque de 10 % !

l'occurrence, cela nous permet d'avoir la quasi-certitude que c'est bien la stratégie de Charlotte qui est la plus efficace [‡]. ♣

## 5.3 Stratification

### Stratification avec échantillonnage uniforme

**Théorème (GY).** *Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo, et que l'espace $\Omega$ sur lequel est définie la probabilité $\mathbb{P}$ est partitionné en différentes « strates » $A_1, ..., A_K$ dont les probabilités respectives (supposées toutes non nulles) sont connues exactement, notées $p_k$. Réalisons $N$ simulations indépendantes de $\mathbb{P}$ et notons $n_k$ le nombre de ce simulations qui tombent respectivement dans $A_k$ ; alors l'*estimateur stratifié

$$\hat{\mu}^{\text{strat}} := \sum_{k=1}^{K} p_k n_k^{-1} \sum_{\omega_i \in A_k} f(\omega_i) \tag{GZ}$$

*a une meilleure efficacité (par simulation) que l'estimateur "simple"* (**??**). ♢

*Démonstration.* Notons $\mathbb{P}_k$ la loi conditionnelle de $\mathbb{P}$ sous $A_k$ (c.-à-d. $\mathbb{P}_k(B) := p_k^{-1}\,\mathbb{P}(B \cap A_k)$), et $m_k := \mathbb{E}_k(f)$. On a alors $\mathbb{P} = \sum_k p_k\,\mathbb{P}_k$, d'où $\mathbb{E}(f) = \sum p_k m_k$. Or on peut réécrire (GZ) comme $\hat{\mu}_{\text{strat}} = \sum_k p_k \hat{\mu}_k$, où $\hat{\mu}_k$ est l'estimateur de Monte-Carlo "ordinaire" de $m_k$, réalisé avec un nombre de simulations égal à $n_k$. Puisque $N$ est très grand, par la loi des grands nombres on a $n_k \sim p_k N$ ; dans la suite, nous ferons comme si on avait exactement $n_k = p_k N$, ce qui ne changera pas le calcul de la valeur de l'efficacité. L'estimateur $\hat{\mu}_k$ a une variance égale à $n_k^{-1} \operatorname{Var}_{\mathbb{P}_k}(f)$, et les différents $\hat{\mu}_k$ sont indépendants, donc la variance globale de $\hat{\mu}^{\text{strat}}$ vaut

$$\sum_{k=1}^{K} p_k^2 n_k^{-1} \operatorname{Var}_{\mathbb{P}_k}(f) = N^{-1} \sum_{k=1}^{K} p_k \operatorname{Var}_{\mathbb{P}_k}(f), \tag{HA}$$

d'où une efficacité par simulation égale à celle qu'aurait une variable de variance $\sum_k p_k \operatorname{Var}_{\mathbb{P}_k}(f)$ : cette quantité est appelée la *variance intrastrates*.

Montrons que la variance globale $\operatorname{Var}(f)$ est plus grande que la variance intrastrates, ce qui prouvera le théorème. On a

$$\begin{aligned} \operatorname{Var}(f) = \mathbb{E}(f^2) - \mathbb{E}(f)^2 &= \sum_k p_k\,\mathbb{E}_k(f^2) - \Big(\sum_k p_k\,\mathbb{E}_k(f)\Big)^2 \\ &= \sum_k p_k \operatorname{Var}_{\mathbb{P}_k}(f) + \Big(\sum_k p_k\,\mathbb{E}_k(f)^2 - \Big(\sum_k p_k\,\mathbb{E}_k(f)\Big)^2\Big). \end{aligned} \tag{HB}$$

Le premier terme de cette somme correspond à la variance intrastrates, tandis que le terme entre crochets est positif par l'inégalité de Cauchy–Schwarz (vu que $\sum_k p_k = 1$). ♤

[‡]. Attention à bien comprendre l'interprétation des résultats. On peut se demander à quoi cela peut bien servir d'avoir réduit la variance, puisque dans les deux cas on aurait choisi la stratgéie de Charlotte ! Ce serait là oublier que l'estimateur de la différence entre les stratégies de Charlotte et Xénia est en fait *aléatoire*, et que le signe que nous avons obtenu sur *ces* simulations aurait pu être différent avec des simulations différentes. En réalité, la vraie différence de performance vaut environ 0,016 en faveur de Charlotte : les estimateurs respectifs de performance sont donc à peu près, dans le cas de deux évaluations indépendantes, une loi normale de moyenne 0,016 et d'écart-type 0,049, et dans le cas d'une évaluation couplée, une loi normale de moyenne 0,016 et d'écart-type 0,019. Dans le premier cas, on a à peu près 37 % de risque que le signe de l'estimateur ne soit pas le bon ; tandis que dans le second, ce risque descend à 20 % : c'est en cela que la réduction de la variance nous permet de prendre une décision plus pertinente !

*Exemple* (HC) (La Lorraine indépendante ?). Un référendum va être organisé en Lorraine, pour ou contre l'indépendance de la région. Un institut de sondage essaye de prédire les résultats. Nous ferons les hypothèses suivantes :

— L'institut de sondage dispose d'un mécanisme qui lui permet de tirer au hasard un électeur lorrain de façon exactement uniforme.

— Tous les électeurs interrogés répondent, et répondent honnêtement [§].

— Les électeurs ne changeront pas d'avis d'ici le scrutin définitif.

Dans ce cas, faire un sondage est exactement équivalent à évaluer la différence entre la proportion d'électeurs qui voteront « oui » et celle de « non » par la méthode de Monte-Carlo, ce qui est doit ainsi effectivement prédire le résultat du référendum.

Ayant interrogé 1 000 électeurs, notre institut de sondage a obtenu les résultats suivants :

| | Sondés | Pour | Contre | Blanc | Bilan |
|---|---|---|---|---|---|
| Global | 1 000 | 457 (45,7 %) | 467 (46,7 %) | 76 (7,6 %) | −10 (−1,0 %) |

D'après ces résultats, la méthode de Monte-Carlo lui permettra d'annoncer comme estimateur une victoire du « non » par une avance de 1,0 % des inscrits. Cependant, si on calcule sa marge d'erreur, on trouve que celle-ci est de ±6,0 %, ce qui signifie qu'il reste encore une très grande incertitude sur le résultat...

Mais il se trouve que, en plus de demander leur avis aux sondés, les sondeurs leur ont demandé de quel département de la Lorraine ils étaient : Meurthe-et-Moselle (*54*), Meuse (*55*), Moselle (*57*) ou Vosges (*88*). Et les résultats détaillés montrent une nette disparité d'un département à l'autre :

| D$^{pt}$ | Sondés | Pour | Contre | Blanc | Bilan |
|---|---|---|---|---|---|
| *54* | 284 | 35 (12,3 %) | 220 (77,5 %) | 29 (10,2 %) | −185 (−65,1 %) |
| *55* | 95 | 36 (37,9 %) | 52 (54,7 %) | 7 ( 7,4 %) | − 16 (−16,9 %) |
| *57* | 418 | 373 (89,2 %) | 16 ( 3,8 %) | 29 ( 6,9 %) | +357 (+85.4 %) |
| *88* | 203 | 13 ( 6,4 %) | 179 (88,2 %) | 11 ( 5,4 %) | −166 (−81,8 %) |

Évidemment, sans information supplémentaire, de telles données détaillées ne permettent pas d'affiner l'estimation en quoi que ce soit. Mais information supplémentaire il y a ! Car notre institut de sondage connait grâce au recensement le nombre exact d'électeurs de chaque département : il y a exactement 30 % des électeurs inscrits dans le *54*, 10 % dans le *55*, 40 % dans le *57* et 20 % dans le *88*. L'institut peut donc considérer qu'il a en fait procédé à quatre sondages partiels indépendants, chacun de ces sondages partiels lui donnant une estimation de la contribution du département concerné au résultat global (nous faisons figurer également les marges d'erreur) :

| D$^{pt}$ | Résultat loc. (%) | Prop. d'électeurs | Contribution (%) |
|---|---|---|---|
| *54* | −65,1 ± 8,0 | 30 % | −19,5 ± 2,4 |
| *55* | −16,8 ± 19,1 | 10 % | − 1,7 ± 1,9 |
| *57* | +85,4 ± 4,3 | 40 % | +34,2 ± 1,7 |
| *88* | −81,8 ± 7,2 | 20 % | −16,4 ± 1,4 |

[§]. Ici nous ne considérons pas la possiblité d'abstention : d'un point de vue pratique, cela peut correspondre à une situation où le vote est obligatoire, ou où les abstentions sont intégrées aux votes blancs.

En totalisant les estimations pour chaque contribution, l'institut obtient alors un estimateur pour le bilan global de $-3{,}4 \pm 3{,}8$ %. En ce qui concerne la marge d'erreur, on l'a obtenue ainsi : étant donné que les estimateurs des différentes contributions sont indépendants, il faut ajouter leurs *variances* (et non pas naïvement les intervalles eux-mêmes) pour obtenir la variance de l'estimateur somme ; et puisque la largeur de l'intervalle de confiance est proportionnelle à l'écart-type, cela donne une largeur pour l'intervalle global qui est la racine de la somme des carrés des largeurs, soit 3,8 % (ce qui est nettement plus petite que la somme des largeurs, qui serait 7,5 %).

On constate donc deux choses :

— L'estimateur prenant en compte la stratification par département n'est pas le même que l'estimateur "brut" ! En l'occurrence, au lieu d'une victoire étriquée du « non », il prédit une victoire du « non » par une avance plus ample [¶].

— La précision de l'estimateur stratifié est sensiblement meilleure que celle de l'estimateur brut : la marge d'erreur est en effet passée de $\pm 6{,}0$ % à $\pm 3{,}8$, ce qui représente une efficacité deux fois et demie supérieure ! L'utilisation intelligente des données sur les départements nous a donc permis de largement améliorer notre estimation [‖].

En l'occurrence, le combinaison des deux phénomènes change très fortement la conclusion du sondage : alors qu'avant, il ne pouvait que donner un léger avantage au « non » (le niveau de risque pour la victoire du « oui » étant de 37 %), à présent il y a une très forte conviction que c'est bien le « non » qui va gagner ! (le niveau de risque pour la victoire du « oui » étant tombé à 4 %). ♧

## Allocation interstrates optimale

**Théorème (HD).** *Supposons qu'on cherche à évaluer par la méthode de Monte-Carlo une quantité de la forme* $\mathbb{E}(f_1) + \cdots + \mathbb{E}(f_K)$ *en estimant séparément les différents* $\mathbb{E}(f_k)$. *La façon optimale de diviser le cout entre les différents calculs consiste alors à consacrer au calcul de* $\mathbb{E}(f_k)$ *un cout proportionnel à* $\mathit{Eff}_k^{-1/2}$, *où* $\mathit{Eff}_k$ *désigne l'efficacité du calcul de* $\mathbb{E}(f_k)$. ◇

[¶]. Attention : Ici il se trouve que l'estimateur raffiné du score global "amplifie" la prédiction de l'estimateur simple ; mais cela est totalement *fortuit* ! (En fait, les deux estimateurs sont sans biais, donc l'estimateur raffiné aurait très bien pu amoindrir, voire renverser, la conclusion de l'estimateur simple...). Le fait que le « non » sorte renforcé de la prise en compte de l'information sur les départements vient en fait de ce que, lorsqu'on regarde les départements de provenance des sondés, on s'aperçoit qu'on a "sur-sondé" le *57* (on y a en effet interrogé 418 personnes, là où une représentation rigoureusement proportionnelle aurait dû conduire à 400 sondés), qui est un département extrêmement favorable au « oui », et "sous-sondé" le *54*, qui est un département globalement très au « non ». De ce fait, on se doute bien qu'on a obtenu trop de « oui » et trop peu de « non » par rapport à ce qu'on aurait "dû" recevoir... Mais bien sûr, le fait que tel ou tel département ait été sur- ou sous-sondé résultait uniquement de l'aléa dû au tirage au sort des personnes sondées, et ces fluctuations auraient donc tout aussi bien pu avoir lieu dans l'autre sens...

[‖]. Attention : quand je dis qu'on a « amélioré notre estimation », je parle *uniquement* de la réduction de la marge d'erreur de l'estimation, et pas du fait que l'estimateur en tant que tel amplifie les résultats de l'estimateur précédent : puisque cette amplification est purement fortuite, comme nous l'avons fait remarquer à la note précédente. En fait, il se pourrait même que dans certains cas, la prise en compte des données de stratification fasse que l'institut de sondage réalise qu'il aura *plus* de mal à prédire le vainqueur qu'il ne l'aurait cru de prime abord, parce que l'estimateur en lui-même s'est très fortement rapproché de 0 avec la nouvelle méthode... Ce qui reste vrai dans tous les cas, en revanche, c'est que la prise en compte de la stratification a permis de *resserrer* la marge d'erreur sur l'estimation du score !

*Démonstration.* Notons $Z$ le cout total de calcul, et $q_k$ la proportion de $Z$ qu'on consacre au calcul de $\mathbb{E}(f_k)$. Le cout de calcul total consacré à l'estimation de $\mathbb{E}(f)$ est alors $q_k Z$, et la variance de l'estimateur obtenu est donc de $(Eff_k q_k Z)^{-1}$. Comme les estimateurs des différents $\mathbb{E}(f_k)$ sont indépendants, leurs variances s'ajoutent, de sorte que la variance globale pour l'estimateur de $\mathbb{E}(f_1)+\cdots+\mathbb{E}(f_K)$ vaut $\sum_k (Eff_k q_k)^{-1} \times Z^{-1}$, ce qui correspond à une efficacité globale de

$$\Big(\sum_{k=1}^{K} (Eff_k q_k)^{-1}\Big)^{-1}. \qquad (\mathsf{HE})$$

Il faut donc maximiser (HE), ou encore minimiser $\sum_k Eff_k^{-1} q_k^{-1}$, sous la contrainte que $\sum_k q_k = 1$. On va utiliser la méthode du multiplicateur de Lagrange : on introduit un paramètre $\lambda$ et on cherche une valeur critique de $f(q_1, \dots, q_K) := \sum_k Eff_k^{-1} q_k^{-1} + \lambda \sum_k q_k$, qui vérifie la contrainte. Vu que $\partial f / \partial q_k = -Eff_k^{-1} q_k^{-2} + \lambda$, il y a un unique point critique pour $q_k = \lambda^{-1/2} Eff_k^{-1/2}$ $\forall k$, ce qui montre (sans même avoir à calculer $\lambda$) que les $q_k$ optimaux sont proportionnels aux $Eff_k^{-1/2}$. ♤

*Exemple* (HF) (La Lorraine indépendante, suite). Reprenons l'exemple de notre sondage sur l'indépendance de la Lorraine. Un institut concurrent de notre premier institut de sondage veut procéder à sa propre estimation. Pour faire mieux que le premier institut, il désire arriver à une précision meilleure pour un investissement financier égal. Nous supposerons ici que le cout du sondage est directement proportionnel au nombre d'individus sondés, et ce, indépendamment de leur département, de sorte que l'efficacité du sondage relatif à un département donné sera directement proportionnelle à l'inverse de la variance par habitant pour ce département. Nous supposons également qu'il existe pour l'institut un moyen de sélectionner un habitant au hasard de n'importe quel département de notre choix.

En analysant les données du sondage du premier institut, le second institut calcule les facteurs de cout suivants [**] pour chaque département [††] :

| Dép$^{t}$ | Fact. rés. loc. | Poids | Fact. contrib. glob. | Allocation | Normalisée |
|---|---|---|---|---|---|
| *54* | 0,474 | 30 % | $4{,}26 \times 10^{-2}$ | 0,206 | 35,2 % |
| *55* | 0,898 | 10 % | $0{,}90 \times 10^{-2}$ | 0,095 | 16,2 % |
| *57* | 0,201 | 40 % | $3{,}22 \times 10^{-2}$ | 0,179 | 30,6 % |
| *88* | 0,277 | 20 % | $1{,}11 \times 10^{-2}$ | 0,105 | 18,0 % |

♧

On en déduit que, pour optimiser sa variance, notre second institut devra consacrer une proportion 0,354 de son cout (soit, en l'occurrence, du nombre de sondés) sur le *54*, ce qui correspond, pour un total de 1 000 sondés, à interroger 354 *54*-iens ; et de même, 162 *55*-iens, 306 *57*-iens et 180 *88*-iens. Si nous comparons avec ce qu'aurait donné un échantillon uniforme (300 *54*-iens, 100 *55*-iens, 400 *55*-iens et 200 *88*-iens), on s'aperçoit qu'il va falloir notamment interroger moins de *57*-iens (on va même en interroger moins que des *54*-iens, alors qu'ils sont plus nombreux dans l'absolu !) et plus de *55*-iens. Cela est lié au fait que le facteur de cout pour le résultat local du *57* (où les résultats étaient très tranchés) était très faible, de

[**]. L'unité de mesure pour le facteur de cout étant le nombre de personnes à sonder par unité dans la contribution au bilan

[††]. Les colonnes du tableau correspondent respectivement aux données suivantes : numéro du département ; facteur de cout si on ne s'intéresse qu'à évaluer le résultat dans le département ; poids du département dans la population de la région ; facteur de cout pour la contribution du département au résultat global ; nombre auquel l'allocation des couts pour le département doit être proportionnelle ; proportion du cout total allouée au département.

sorte qu'il n'y avait pas besoin d'interroger beaucoup d'électeurs pour obtenir une bonne précision, alors qu'à l'inverse le facteur de cout du *55* (où les résultats étaient beaucoup plus équilibrés) était élevé, de sorte qu'il faut y interroger beaucoup de personnes pour obtenir un résultat assez fin.

Ayant décidé d'utiliser ces valeurs, notre institut de sondage obtient les résultats suivants :

| D$^{\text{pt}}$ | Sondés | Pour | Contre | Blanc | Bilan |
|---|---|---|---|---|---|
| *54* | 352 | 54 (15,3 %) | 264 (75,0 %) | 34 (9,7 %) | −210 (−59,7 %) |
| *55* | 162 | 70 (43,2 %) | 88 (54,3 %) | 4 (2,5 %) | − 18 (−11,1 %) |
| *57* | 306 | 273 (89,2 %) | 9 ( 2,9 %) | 24 (7,8 %) | +264 (+86.3 %) |
| *88* | 180 | 12 ( 6,7 %) | 157 (87,2 %) | 11 (6,1 %) | −145 (−80,6 %) |

| D$^{\text{pt}}$ | Résultat loc. (%) | Prop. d'électeurs | Contribution (%) |
|---|---|---|---|
| *54* | −59,7 ± 7,7 | 30 % | −17,9 ± 2,3 |
| *55* | −11,1 ± 15,1 | 10 % | − 1,1 ± 1,5 |
| *57* | +86,3 ± 4,7 | 40 % | +34,5 ± 1,9 |
| *88* | −80,6 ± 7,9 | 20 % | −16,1 ± 1,1 |

Cela lui donne finalement un estimateur du bilan global de $-0{,}6 \pm 3{,}7$ % : par rapport aux fluctuations de $\pm 3{,}8$ % du premier institut, notre second institut a ainsi amélioré son efficacité d'environ 6 % [‡‡] — ce qui, dans un contexte de concurrence féroce, peut s'avérer tout-à-fait intéressant !

*Remarque* (HG). La situation du sondage présentée dans cet exemple est un cas un peu particulier, puisque c'est en substance la « même » fonction dont on évalue l'espérance dans chaque strate. Alors que le théorème (HD) nous montre qu'en réalité la problématique d'allocation optimale se pose dès lors qu'on a à calculer une somme d'espérances, sans qu'il y ait à avoir quelque lien que ce soit entre elles ! Ainsi, l'idée que chaque strate (en l'occurrence, les départements) aurait un « poids » naturel dans l'estimateur global n'est plus valable dans le cas général, pas plus que ne l'est la notion de « résultat local » : dans la situation générale, seule la contribution au résultat global a un sens. ♧

## 5.4 Variables de contrôle

### Version rudimentaire

**Définition (HH).** Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo, et qu'on dispose d'une variable aléatoire $g$ de classe $\mathrm{L}^2$ sur $\Omega$ dont on connaisse l'espérance exactement. Dire qu'on utilise $g$ comme *variable de contrôle* pour évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo signifie, dans sa version "rudimentaire", qu'on s'appuie sur l'écriture « $\mathbb{E}(f) = \mathbb{E}(f-g) + \mathbb{E}(g)$ », autrement dit qu'on considère l'estimateur

$$\hat{\mu}^g := N^{-1} \sum_{i=1}^{N} (f(\omega_i) - g(\omega_i)) + \mathbb{E}(g). \qquad \text{(HI)}$$

♡

!

**Proposition (HJ).** *L'efficacité par simulation de la méthode de Monte-Carlo utilisant la variable de contrôle $g$ (au sens rudimentaire) est l'inverse de la variance de $(f-g)$.* ◇ !

*Démonstration.* Évident. ♤

*Exemple* (HK) (Mathématiques financières). En finance, certains contrats permettent de vendre un actif à la valeur maximale que son cours prendra sur une certaine période. Dans ce cadre, il est important d'être capable d'évaluer l'espérance du maximum ou du minimum d'une trajectoire ; cela motive l'exemple simplifié que nous présentons maintenant.

On considère une marche aléatoire de 60 pas sur $\mathbb{R}$ issue de 0 dont les incréments sont de loi Normale(1) ; autrement dit, pour $(S_j)_{1\leqslant j\leqslant 60}$ des v.a. i.i.d. Normale(1), on considère la suite finie $(X_i)_{0\leqslant i\leqslant 60}$ définie par $X_i = \sum_{j=1}^{i} S_j$. On définit la variable aléatoire $X_*$ comme la plus grande valeur atteinte par la suite aléatoire $(X_i)_i$, c'est-à-dire que pour tout $\omega$, on pose $X_*(\omega) \coloneqq \max_{0\leqslant i\leqslant 60} X_i(\omega)$. Notre but est d'évaluer $\mathbb{E}(X_*)$.

En première approximation, si on imagine que le comportement que $(X_i)_i$ au cours du temps est à peu près linéaire, on peut considérer que $X_* \simeq \max\{X_{60}, 0\} =: Y$. Or on sait calculer exactement $\mathbb{E}(Y)$ : en effet, $Y \sim$ Normale(60), donc, notant $\sigma \coloneqq \sqrt{60}$,

$$\mathbb{E}(Y) = \int_{-\infty}^{\infty} \max\{x,0\} \frac{\mathrm{e}^{-x^2/2\sigma^2}}{\sqrt{2\pi}\sigma}\,\mathrm{d}x = \frac{1}{\sqrt{2\pi}\sigma}\int_0^{\infty} x\mathrm{e}^{-x^2/2\sigma^2}\mathrm{d}x$$
$$= \frac{1}{\sqrt{2\pi}\sigma}\Big(-\sigma^2\mathrm{e}^{-x^2/2\sigma^2}\Big)_0^{\infty} = \frac{\sigma}{\sqrt{2\pi}} = 3{,}090\,193... \quad \text{(HL)}$$

Cela suggère d'utiliser $X_{60} \wedge 0 =: Y$ comme variable de contrôle pour $\mathbb{E}(X_*)$.

Les codes sur *Arche* montrent comment implémenter l'utilisation de cette variable de contrôle. Sans variable de contrôle, on obtient le résultat suivant :

```
>>> topchrono = process_time(); mathfi_naif(100000); \
... duree = process_time() - topchrono; \
... print('Temps écoulé : {:.2f} s.'.format(duree))
Estimation de E(X_*) : 5.608389 ± 0.028868.
Temps écoulé : 4.33 s.
```

Et avec l'utilisation rudimentaire de la variable de contrôle $Y$ :

```
>>> topchrono = process_time(); mathfi_ctrl0(100000); \
... duree = process_time() - topchrono; \
... print('Temps écoulé : {:.2f} s.'.format(duree))
Estimation de E(X_*) : 5.620559 ± 0.014314.
Temps écoulé : 4.39 s.
```

On voit donc que, pour un temps de calcul à peu près égal, l'incertitude a été réduite d'un peu plus d'un facteur 2, soit une amélioration de l'efficacité d'un facteur 4. ♧

[‡‡]. Ci-dessus les valeurs arrondies des marges d'erreurs à resp. ±3,7 % et ±3,8 % ne permettent pas d'évaluer précisément de gain d'efficacité : le chiffre de 6 % donné ci-dessus a été calculé à partir de valeurs non arrondies.

### Version paramétrée

Si on dispose d'une variable de contrôle $g$, on peut aussi se servir de $\lambda g$ comme variable de contrôle pour n'importe quel $\lambda \in \mathbb{R}$, puisqu'on connait alors $\mathbb{E}(\lambda g) = \lambda\,\mathbb{E}(g)$. Cela suggère donc de choisir la valeur la plus judicieuse possible pour $\lambda$.

**Théorème (HM).** *La valeur de $\lambda$ qui minimise* $\mathrm{Var}(f - \lambda g)$ *est*

$$\lambda^* = \mathrm{Cov}(f, g)\,\mathrm{Var}(g)^{-1}. \tag{HN}$$

*Notant $r \coloneqq \mathrm{Cov}(f,g) \,/\, \mathrm{Var}(f)^{1/2}\,\mathrm{Var}(g)^{1/2}$ le coefficient de corrélation entre $f$ et $g$, l'efficacité est alors améliorée d'un facteur $(1-r^2)^{-1}$ : cette technique est donc d'autant plus efficace que $|r|$ est très proche de 1.* ♢

*Démonstration.* On développe par bilinéarité $\mathrm{Var}(f-\lambda g) = \mathrm{Var}(f) - 2\lambda\,\mathrm{Cov}(f,g) + \lambda^2\,\mathrm{Var}(g)$. Il s'agit est un polynôme de degré 2 en $\lambda$ dont la minimisation est élémentaire : le minimum de $\mathrm{Var}(f-\lambda g)$ est atteint en $\lambda^*$ et vaut $\mathrm{Var}(f) - \mathrm{Var}(g)^{-1}\,\mathrm{Cov}(f,g)^2 = (1-r^2)\,\mathrm{Var}(f)$, d'où les résultats annoncés. ♤

*Exemple* (HO). Si nous reprenons l'exemple des mathématiques financières avec toujours la variable de contrôle $Y$, mais cette fois-ci dans sa version paramétrée, nous arrivons au programme `mathfi_ctrl1` que vous trouverez sur *Arche*. L'exécution donne :

```
>>> topchrono = process_time(); mathfi_ctrl1(100000); \
... duree = process_time() - topchrono; \
... print('Temps écoulé : {:.2f} s.'.format(duree))
Estimation de E(X_*) : 5.620256 ± 0.014040.
Temps écoulé : 4.42 s.
```

En l'occurrence on ne gagne presque rien par rapport à la version simple de la variable de contrôle, parce qu'il se trouve que la $\lambda$ optimal vaut à peu près 0,9, de sort que la version simple (correspondant à $\lambda = 1$) était déjà pratiquement optimale. ♧

### Variables de contrôle multiples

On peut étendre l'idée du paragraphe précédent au cas de plusieurs variables de contrôle $g_1, \dots, g_K$, en appliquant (HI) avec $\lambda_1 g_1 + \cdots + \lambda_K g_K$, $(\lambda_1, \dots, \lambda_K)$ étant un jeu de $K$ paramètres à optimiser.

**Théorème (HP).** *La valeur de $(\lambda_1, \dots, \lambda_K)^\mathsf{T} =: \vec{\lambda}$ qui minimise* $\mathrm{Var}(f - \lambda_1 g_1 - \cdots - \lambda_K g_K)$ *est donnée par*

$$\vec{\lambda}^* = \boldsymbol{\Sigma}_g^{-1}\,\vec{R}_{fg}, \tag{HQ}$$

*où $\vec{R}_{fg}$ est le vecteur $\big(\mathrm{Cov}(f,g_1), \dots, \mathrm{Cov}(f,g_K)\big)^\mathsf{T}$ et $\boldsymbol{\Sigma}_g$ est la matrice de covariance (supposée non dégénérée) de $(g_1, \dots, g_K)^\mathsf{T}$.* ♢

*Démonstration.* Par bilinéarité,

$$\mathrm{Var}(f - \lambda_1 g_1 - \cdots - \lambda_K g_K) = \mathrm{Var}(f) - 2\vec{\lambda}^\mathsf{T}\vec{R}_{fg} + \vec{\lambda}^\mathsf{T}\boldsymbol{\Sigma}_g\vec{\lambda}, \tag{HR}$$

où nous rappelons que $\boldsymbol{\Sigma}_g$, en tant que matrice de covariance, est symétrique et positive (définie). Ce polynôme de degré 2 en $\lambda$ se réduit en

$$\mathrm{Var}(f) + \|\boldsymbol{\Sigma}_g^{1/2}\vec{\lambda} - \boldsymbol{\Sigma}_g^{-1/2}\vec{R}_{fg}\|^2 - \vec{R}_{fg}^\mathsf{T}\boldsymbol{\Sigma}_g^{-1}\vec{R}_{fg}, \tag{HS}$$

où $\|\vec{v}\|^2$ désigne la norme $\vec{v}^\mathsf{T}\vec{v}$ d'un vecteur $\vec{v}$ de $\mathrm{L}^2(\mathbb{R}^K)$. Le seul terme de (HS) dépendant de $\vec{\lambda}$ étant alors le carré de la norme de $\boldsymbol{\Sigma}_g^{1/2}\vec{\lambda} - \boldsymbol{\Sigma}_g^{-1/2}\vec{R}_{fg}$, l'expression est minimale quand ce vecteur s'annule, ce qui donne bien $\vec{\lambda}^* = \boldsymbol{\Sigma}_g^{-1}\vec{R}_{fg}$. ♤

*Exemple* (HT). Reprenons encore une fois l'exemple des mathématiques financières. Je prends trois variables de contrôle : $X_{20} \wedge 0 \coloneqq Y_1$, $X_{40} \wedge 0 \coloneqq Y_2$ et $X_{60} \wedge 0 \coloneqq Y_3$, qui ont pour espérances respectives $\sqrt{20\,/\,2\pi}$, $\sqrt{40\,/\,2\pi}$ et $\sqrt{60\,/\,2\pi}$. Cela conduit au programme `mathfi_ctrl3` sur *Arche*, dans lequel nous avons utilisé le formalisme vectoriel offert par le module `numpy.matrix` pour traiter $Y_1$, $Y_2$ et $Y_3$ avec une seule ligne de code à chaque fois. L'exécution donne :

```
>>> topchrono = process_time(); mathfi_ctrl3(100000); \
... duree = process_time() - topchrono; \
... print('Temps écoulé : {:.2f} s.'.format(duree))
Estimation de E(X_*) : 5.616246 ± 0.010067.
Temps écoulé : 7.50 s.
```

On a donc encore gagné un facteur 2 sur l'efficacité par simulation par rapport à l'utilisation d'une seule variable de contrôle. ♧

## 5.5 Variables antithétiques

**Théorème (HU).** *Supposons que nous cherchons à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo. Notant $\Omega$ l'espace sur lequel vivent nos simulations, supposons que nous disposions d'une application non triviale $\gamma\colon \Omega \to \Omega$ qui préserve la mesure $\mathbb{P}$ (c.-à-d. que la mesure-image de $\mathbb{P}$ par $\gamma$ est encore $\mathbb{P}$), et supposons que simuler $\big(f(\omega), f(\gamma(\omega))\big)$ coute deux fois plus cher que simuler $f(\omega)$. Alors, si les variables aléatoires $f$ et $f \circ \gamma$ sont négativement corrélées, l'estimateur*

$$\hat{\mu}^{\text{anti}} \coloneqq (2N)^{-1} \sum_{i=1}^{N} \big(f(\omega_i) + f(\gamma(\omega_i))\big) \tag{HV}$$

*est plus efficace que l'estimateur de Monte-Carlo "simple".* ♢

> *Démonstration.* L'efficacité de cette méthode par simulation est l'inverse de la variance de $\frac{1}{2}\big(f(\omega_i)+f(\gamma(\omega_i))\big)$, laquelle se développe en $\frac{1}{4}\big(\mathrm{Var}(f)+\mathrm{Var}(f\circ\gamma)+\mathrm{Cov}(f, f\circ\gamma)\big) = \frac{1}{2}\,\mathrm{Var}(f)+\frac{1}{2}\,\mathrm{Cov}(f, f\circ\gamma) \leqslant \frac{1}{2}\,\mathrm{Var}(f)$ en utilisant successivement que $f\circ\gamma$ a la même loi que $f$ et l'hypothèse de corrélation négative entre $f$ et $f\circ\gamma$. Ainsi l'efficacité par simulation est au moins 2 fois meilleure pour la méthode antithétique ; comme en outre le coût par simulation est au plus 2 fois plus important, au final on obtient bien que l'efficacité tout court est meilleure avec la méthode antithétique. ♤

*Exemple* (HW). Nous prenons encore une fois l'exemple des mathématiques financières. Nous allons adjoindre à la simulation de notre processus financier une simulation "antithétique" où la trajectoire du processus financier sera symétrique par rapport à 0 ! Il est clair que la trajectoire symétrique a bien la même loi que la trajectoire originale, de sorte que celle-ci correspond toujours à un échantillonnage sous la loi de probabilité qui nous intéresse. En outre, on peut raisonnablement supposer que la corrélation entre les valeurs de $X_*$ pour des deux trajectoires sera négative, puisque quand $X_*$ est grand pour la première trajectoire, c'est que celle-ci a eu tendance à prendre de grandes valeurs, et donc la trajectoire symétrique a eu tendance à prendre de petites valeurs, et donc $X_*$ est petit pour la trajectoire symétrique. Voir l'implémentation `mathfi_anti` sur *Arche*, qui donne à l'exécution :

```
>>> topchrono = process_time(); mathfi_anti(100000); \
... duree = process_time() - topchrono; \
```

```
... print('Temps écoulé : {:.2f} s.'.format(duree))
Estimation de E(X_*) : 5.61616 ± 0.01621
Temps écoulé : 2.30 s.
```

On voit qu'on a gagné par rapport à l'algorithme de base, non seulement en termes de variance (d'un facteur plus de 3), mais aussi sur la vitesse par simulation, ce qui s'explique par le fait que nous avons eu à effectuer deux fois moins de tirages aléatoires, chaque tirage étant effectué deux fois. ♧

## Deuxième partie

# Autres méthodes de Monte-Carlo

# Chapitre 6

# Chaines de Markov pour Monte-Carlo

## 6.1 Rappels et compléments sur les chaines de Markov

Dans ce chapitre, nous allons beaucoup parler de chaines de Markov et de leurs lois d'équilibre. Rappelons (ou donnons) ici les définitions et propriétés afférentes à ces concepts : d'autant que, dans notre contexte focalisé sur la simulation, nous utiliserons un formalisme légèrement différent de celui auquel vous êtes habitués.

**Définition (HX)** (Noyau markovien)**.** Un espace (mesurable) $\Omega$ étant donné, un *noyau markovien* sur $\Omega$ est la donnée d'une application [*] $k : \Omega \to \mathscr{M}_1(\Omega)$ allant de $\Omega$ dans l'ensemble des lois de probabilité sur $\Omega$. Intuitivement, $k(x)$ dit comment sera distribuée l'étape suivante de notre chaine de Markov lorsqu'on fait un pas depuis $x$. !

En pratique, pour décrire la masse que la loi de probabilité $k(x)$ donne à la zone $\mathrm{d}y$, on ne notera pas $\big(k(x)\big)(\mathrm{d}y)$, mais $k(x, \mathrm{d}y)$. (On voit alors $k$ comme une application de $\Omega \times \mathscr{P}(\Omega)$ dans $[0, 1]$). ♡

*Remarque* (HY). Lorsque l'espace $\Omega$ est discret, la donnée d'un noyau markovien est équivalente à celle d'une *matrice de transition* $\big(\big(p(x, y)\big)\big)_{x,y\in\Omega}$ avec $p(x, y) \geqslant 0$ pour tous $x, y$ et $\sum_{y\in\Omega} p(x, y) = 1$ : dans ce cas, la loi $k(x)$ correspond à

$$k(x) = \sum_{y\in\Omega} p(x, y)\delta_y, \tag{HZ}$$

ce qui est encore équivalent à dire que $p(x, y) = k(x, \{y\})$.

Néanmoins, le formalisme de la matrice de transition n'est pas adapté aux cas où l'espace $\Omega$ est continu, ce qui sera la situation la plus fréquente dans ce chapitre. ♧

**Définition (IA)** (Chaine de Markov)**.** Un noyau markovien $k$ sur un espace $\Omega$ et une loi de probabilité $\mu_0$ sur ce même espace étant donnés, la *chaine de Markov* associée est (la loi de) le processus aléatoire $(X_t)_{t\in\mathbb{N}}$ à valeurs dans $\Omega^{\mathbb{N}}$ caractérisé par les propriétés suivantes :

— $\mathrm{Loi}(X_0) = \mu_0$ ;

[*]. Il faudrait aussi ajouter certaines conditions de régularité pour que la définition soit rigoureusement exploitable ; mais nous omettrons ces aspects dans le cadre de ce cours.

— Pour tous $x_0, \dots, x_t \in \Omega$,

$$\mathrm{Loi}(X_{t+1} \mid X_0 = x_0 \text{ et } \dots \text{ et } X_t = x_t) = k(x_t). \tag{IB}$$

Cette définition donné alors un moyen immédiat de simuler (un fragment initial de) la chaine de Markov (sous réserve qu'on sache simuler les lois $k(x)$ ainsi que $\mu_0$) : on simule d'abord $X_0$, puis on simule récursivement $X_1, X_2, \dots$ selon les lois respectives $k(x_{0\checkmark}), k(x_{1\checkmark}), \dots$, où les éléments aléatoires intervenant dans le tirage de la loi $k(x_{t\checkmark})$ sont pris, à chaque fois, de façon complètement indépendante de tout ce qui précède. ♡

*Exemple* (IC). Par exemple, le code suivant simule et affiche les 20 premiers pas de la chaine de Markov sur $]-2,2[$ pour laquelle la loi initiale $\mu_0$ est $\mathrm{Unif}^{\mathrm{me}}(-2,2)$ et où le noyau de transition est défini par $k(x) \coloneqq \mathrm{Unif}^{\mathrm{me}}(x/2-1, x/2+1)$ :

```
TMAX = 20 # Nombre de pas
t = 0 # Temps courant
x = -2 + 4 * random() # Valeur courante de la chaine de Markov
print(f'{x:.4f}', end='') # Affichage de la valeur
while t < TMAX: # Boucle de construction pas à pas
    # Pas suivant de la chaine de Markov
    t += 1
    x = x / 2 - 1 + 2 * random()
    print(', ' + f'{x:.4f}', end='')
print('\n')
```

L'exécution donne :

```
1.0318, 0.3571, -0.3036, -0.1293, -0.2548, 0.4402, -0.1733, -0.1334,
0.1000, 0.8662, 0.4425, -0.2151, 0.4041, 0.4388, -0.2796, 0.6797,
1.3054, 1.2731, 1.4409, 0.3407, 0.6300
```

♧

**Définition (ID).** Une loi de probabilité $P$ sur $\Omega$ est qualifiée de *loi d'équilibre* pour le noyau markovien $k(\bullet)$ lorsque la chaine de Markov de noyau $k$ partant de la loi $P$, notée ici $(X_t)_{t\in\mathbb{N}}$, vérifie $\mathrm{Loi}(X_1) = P$ (et donc $\mathrm{Loi}(X_t) = P$ pour tout $t$). Cela se traduit par le fait que le noyau vérifie, pour tout voisinage infinitésimal élémentaire $\mathrm{d}y$ de tout $y \in \Omega$ :

$$P(\mathrm{d}y) = \int_{x\in\Omega} P(\mathrm{d}x)k(x, \mathrm{d}y). \tag{IE}$$

♡

La loi d'équilibre, lorsqu'elle existe, est unique — du moins si notre chaine permet réellement d'explorer tout $\Omega$ :

**Théorème (IF).** *Si un noyau markovien a une loi d'équilibre, alors, sous des hypothèses de type « irréductibilité »* [†]*, cette loi d'équilibre est unique.* ◇

[†]. Rappelons que l'irréductibilité signifie, en substance, pour pour tous $x$ et $y$ de $\Omega$, il existe un moyen d'aller de $x$ à $y$ en suivant un nombre fini de pas compatibles avec le noyau. Formaliser rigoureusement cette notion dans le cadre d'un espace continu serait assez technique ; mais en pratique, dans les exemples considérés, il sera toujours intuitivement clair de voir si on a irréductibilité ou non.

Tous les noyaux markoviens n'admettent pas de loi d'équilibre : par exemple, une marche aléatoire biaisée part vers l'infini presque-surement et ne saurait donc pas posséder un « équilibre »... Cependant, sous réserve qu'il y ait quelque chose qui tende à "empêcher le processus de partir trop loin", l'existence d'une loi d'équilibre est assurée :

**Théorème (IG).** *Supposons qu'il existe une fonction $L(\bullet) : \Omega \to \mathbb{R}_+$ qui vérifie les conditions suivantes :*

- *Pour tout $M < \infty$, l'ensemble $\{x \in \Omega \mid L(x) \leqslant M\}$ est contenu dans un compact*[‡]*. [Cette condition exprime l'idée que, pour que $x$ s'éloigne à l'infini, il y a besoin que $L(x)$ diverge vers l'infini] ;*
- *Il existe des constantes $a < 1$ et $b < \infty$ telles que, pour tout $x$, notant $(X_t)_{t\in\mathbb{N}}$ une chaine de Markov de noyau $k$ :*

$$\mathbb{E}(L(X_{t+1}) \mid X_t = x) \leqslant aL(x) + b. \tag{IH}$$

*[Cette condition exprime l'idée que, lorsque $L(x)$ devient grand, le pas suivant de la chaine de Markov aura tendance, en moyenne, à avoir une valeur de $L(\bullet)$ plus petite].*

*Alors il existe une loi d'équilibre pour le noyau markovien $k$.*

*(Dans un tel cas, on qualifie $L(\bullet)$ de* fonction de Liapounov[§] *pour le noyau $k$).* ♢

*Remarque* (II). L'existence d'une fonction de Liapounov est une méthode très générale pour s'assurer qu'une chaine de Markov possède une loi d'équilibre[¶]. En pratique, on peut utiliser cet argument de façon intuitive : s'il existe une "force" qui tend à ramener notre chaine de Markov (en moyenne) plus près d'une certaine "zone centrale" dès lors qu'on s'en éloigne trop, alors il y aura une loi d'équilibre. ♧

Lorsqu'on dispose d'une mesure d'équilibre (supposée ici unique), on peut considérer ce qu'on appelle la *chaine de Markov stationnaire* associée au noyau markovien $k(\bullet)$ : il s'agit tout simplement de la chaine de Markov pour laquelle $X_0$ part de la loi $P_{\text{éq}}$. Dans ce cas, il résulte de la définition du concept de mesure d'équilibre que la loi de la chaine de Markov est invariante lorsqu'on décale l'origine du temps (d'où l'appellation « stationnaire ») :

**Définition (IJ).** Un noyau markovien $k(\bullet)$ possédant une loi d'équilibre $P_{\text{éq}}$ étant donné, on appelle *chaine de Markov stationnaire* (associé au noyau markovien $k(\bullet)$ et à la loi d'équilibre $P_{\text{éq}}$) le chaine de Markov sur $\Omega$ issue de $\text{Loi}(X_0) = P_{\text{éq}}$ et obéissant au noyau $k(\bullet)$. ♡

**Proposition (IK).** *Si $(X_t)_{t\in\mathbb{N}}$ est une chaine de Markov stationnaire, alors le processus décalé d'une unité de temps, à savoir $(X_{t+1})_{t\in\mathbb{N}}$, possède la même loi (en tant que loi sur $\Omega^{\mathbb{N}}$) que le processus de départ.* ♢

*Démonstration.* Par définition de la proriété de Markov, le processus $(X_{t+1})_{t\in\mathbb{N}}$ est lui-même une chaine de Markov de noyau $k$, et puisque $X_1$ suit la loi $P_{\text{éq}}$ (du fait que $X_0$ suit la loi $P_{\text{éq}}$ et que $P_{\text{éq}}$ est une loi d'équilibre), cette chaine de Markov est issue de $P_{\text{éq}}$, comme la chaine origianle : les deux chaines ont donc la même loi ! ♤

*Remarque* (IL). Le processus $(X_{t+1})_t$ pouvant même être défini pour tout $t \in [\![-1, \infty[\![$, on peut considérer la loi de notre chaine de Markov $(X_t)_{t\in\mathbb{N}}$ comme issue d'un processus indexé par $[\![-1, \infty[\![$. Mais la loi de ce processus peut elle-même être vue comme la restriction de la loi d'un processus indexé par $[\![-2, \infty[\![$, lui-même obtenu par restriction d'un processus indexé par $[\![-3, \infty[\![$, ... ; et à la limite, par lemme de classe

[‡]. Dans le cas où $\Omega$ est discret, cela signifie que cet ensemble est fini.

[§]. En anglais : « *Lyapunov function* ».

[¶]. Attention, la condition n'est pas nécessaire : on peut posséder une loi d'équilibre sans admettre pour autant de fonctionnelle de Liapounov.

monotone, on peut même construire la loi d'un processus $(X_t)_{t\in\mathbb{Z}}$ indexé par tous les temps positifs *et négatifs*, loi qui sera invariante par translation, et qui redonnera notre chaine de Markov stationnaire quand on ne considèrera que les temps positifs ! Cette notion de processus stationnaire indexé par $\mathbb{Z}$ [‖] est très utile dans un certain nombre d'applications, même si nous n'en aurons pas l'usage dans le cadre de ce cours ☊ ♧

Un point essentiel avec les lois d'équilibre est que, sur le long terme, presque-surement, la trajectoire de notre chaine de Markov va finir par explorer tout l'espace $\Omega$ en passant, dans les différentes zones de $\Omega$, un temps proportionnel à la loi d'équilibre :

! **Théorème (IM).** *Supposons un noyau markovien $k$ (vérifiant quelques conditions techniques gentilles et) qui possède une propriété d'irréductibilité et admet une loi d'équilibre (unique en vertu de l'hypothèse précédente), notée $P$. Alors, si $(X_t)_{t\in\mathbb{N}}$ est une chaine de Markov de noyau $k$ (peu importe sa mesure de départ), on a presque surement que la mesure empirique des $T$ premiers pas de la chaine tend vers $P$ lorsque le temps vers l'infini : autrement dit, si on réalise cette chaine de Markov, on obtiendra presque-surement une réalisation $(x_{t\checkmark})_t$ telle que*

$$\frac{1}{T}\sum_{t=0}^{T-1}\delta_{x_{t\checkmark}} \overset{T\to\infty}{\to} P \tag{IN}$$

*(où la convergence considérée sur les mesures de probabilité est la convergence faible).* ♢

! **Corolaire (IO).** *Sous les mêmes hypothèses, si $f(\bullet)\colon \Omega \to \mathbb{R}$ est une fonction continue et bornée, alors on a*

$$\frac{1}{T}\sum_{t=0}^{T-1} f(X_t) \overset{T\to\infty}{\to} \mathbb{E}(f(P)) \quad \textit{presque-surement.} \tag{IP}$$

*En fait, on peut montrer que dans ce cas, le résultat est vrai pour toute fonction de $\mathrm{L}^1(P)$. On dit alors que la chaine de Markov vérifie la propriété d'*ergodicité. ♢

La mesure d'équilibre d'une chaine de Markov joue donc un rôle central dans son comportement à long terme. Comme nous l'avons dit, la mesure d'équilibre associée à une chaine de Markov est (normalement) unique ; cependant, il est en général très difficile de déterminer une formule pour celle-ci, quand bien même on diposerait d'une formule simple pour le noyau markovien ! Il existe néanmoins un exception qui sera très importante pour nous, qui est la situation dite de *réversibilité* :

! **Définition (IQ).** Soit $k(\bullet)$ un noyau Markovien sur $\Omega$ et $\mu$ une mesure non identiquement nulle (mais pas forcément de probabilité, ni même de masse totale finie !) sur $\Omega$. On dit que $\mu$ est une *mesure réversible* pour le noyau Markovien si elle vérifie la propriété suivante :

$$\forall x, y\,[**] \in \Omega \qquad \mu(\mathrm{d}x)k(x,\mathrm{d}y) = \mu(\mathrm{d}y)k(y,\mathrm{d}x). \tag{IR}$$

♡

[‖]. Attention : C'est *parce qu'*on considère le processus *stationnaire* qu'on peut étendre le temps à $\mathbb{Z}$ tout entier ; pour une chaine de Markov non stationnaire, cette extension à $\mathbb{Z}$ n'est pas possible !

[**]. Techniquement parlant, la quantification a aussi lieu sur le choix des voisinages infinitésimaux $\mathrm{d}x$ et $\mathrm{d}y$ de resp. $x$ et $y$ ; cependant en pratique il suffit que cela marche sur un voisinage infinitésimal arbitraire pour que cela marche en fait pour tous les voisinages.

*Remarque* (IS). Intuitivement, la condition de réversibilité se comprend en imaginant qu'on a distribué un très grand nombre de particules (indiscernables) sur $\Omega$ selon la mesure $\mu$ [étant entendu que $\mu(\bullet)$ mesure des nombres de "moles" : autrement dit, le nombre de particules dans la région mésoscopique $\mathrm{d}x$ vaut $\mu(\mathrm{d}x)N_{\mathrm{A}}$, où le "nombre d'Avogadro" $N_{\mathrm{A}}$ est une constante choisie suffisamment grande pour que $\mu(\mathrm{d}x)N_{\mathrm{A}}$ soit beaucoup plus grand que 1]. Dans ce cas, si on demande à chaque particule, indépendamment, de se déplacer selon le noyau markovien au point où elle est située (autrement dit, si une particule est située en $x$, elle va choisir sa nouvelle position suivant selon la loi $k(x)$), alors la proportion des particules situées dans $\mathrm{d}x$ qui va sauter vers $\mathrm{d}y$ sera $k(x,\mathrm{d}y)$ [††], de sorte que le nombre total de particules sautant de $\mathrm{d}x$ vers $\mathrm{d}y$ sera égal $\mu(\mathrm{d}x)k(x,\mathrm{d}y)$ ; et symétriquement, le nombre de particules sautant de $\mathrm{d}y$ vers $\mathrm{d}x$ sera égal à $\mu(\mathrm{d}y)k(y,\mathrm{d}x)$. La condition de réversibilité signifie ainsi que les particules qui sautent de $\mathrm{d}x$ vers $\mathrm{d}y$ sont exactement compensées par la particules qui sautent de $\mathrm{d}y$ vers $\mathrm{d}x$, de sorte qu'à l'issue de notre ré-organisation des particules par application du noyau markovien, la distribution des particules n'aura pas changé ! Et non seulement elle n'aura pas changé, mais on ne verra pas de changement non plus si on "renverse la vidéo" : en effet, quand on renverse la vidéo, les particules qu'on voit sauter de $\mathrm{d}x$ vers $\mathrm{d}y$ dans la vidéo renversée correspondent aux particules qu'on avait vu sauter de $\mathrm{d}y$ vers $\mathrm{d}x$ dans le sens normal du temps ; or ce nombre est égal à celui des particules qui sautent de $\mathrm{d}x$ vers $\mathrm{d}y$ dans le sens normal du temps, de sorte qu'on ne voit pas de différence ! C'est de cette propriété d'invariance par renversement du temps que provient la nomenclature « réversible ». ♣

*Remarque* (IT). Il est facile de vérifier, à partir de la définition (IQ), que toutes les mesures réversibles (s'il y en a) sont proportionnelles les unes aux autres. (Sachant qu'on a sous-entendu qu'on ne considérait que des mesures *localement finies*, autrement dit, qui peuvent éventuellement être de masse *globalement* finie, mais qui donnent une messe finie aux ensembles bornés (relativement à une métrique adéquate)). ♣

Il est, pour le coup, facile de vérifier la condition de réversibilité, car contrairement à la condition d'équilibre générale, la formule (IR) ne requiert pas d'intégrer sur $\Omega$ ! Or, lorsqu'on a trouvé une mesure réversible, cela nous donne automatiquement la mesure d'équilibre :

**Théorème (IU).** *(Nous supposons ici que le noyau markovien vérifie les propriétés d'irréductibilité assurant l'unicité d'une éventuelle mesure d'équilibre). Si notre noyau markovien possède une mesure réversible $\mu_{\text{rév}}$ de masse totale finie, alors sa mesure d'équilibre est donnée par*

$$\mu_{\text{éq}}(\mathrm{d}x) = \frac{\mu_{\text{rév}}(\mathrm{d}x)}{\mu_{\text{rév}}(\Omega)} \ : \tag{IV}$$

*autrement dit, $\mu_{\text{éq}}$ est égale à $\mu_{\text{rév}}$ à constante multiplicative près, cette constante étant fixée par la condition que $\mu_{\text{éq}}$ doit être une loi de probabilité.* ◇ !

*Remarque* (IW). On a aussi que, dans le cas où $\mu_{\text{rév}}(\Omega)$ est de masse totale infinie, il n'existe pas de mesure d'équilibre pour le noyau markovien. ♣

[††]. En fait, techniquement, $k(x,\mathrm{d}y)$ désigne la *probabilité* que chacune de ces particules saute vers $\mathrm{d}y$ ; cependant, si notre nombre d'Avogadro a été choisi suffisamment grand, la loi des grands nombres assurera qu'on ne "voit" pas les fluctuations aléatoires, de sorte que la proportion de particules qui feront effectivement ce saut correspondra bien à la probabilité.

## 6.2 Monte-Carlo en contexte non indépendant

### Motivation

Jusqu'ici dans ce cours, lorsque nous nous sommes intéressés à la méthode de Monte-Carlo pour évaluer une espérance $\mathbb{E}(f(P))$ (pour $P$ une loi de probabilité sur un certain $\Omega$, et $f$ une fonction de $\Omega$ dans $\mathbb{R}$), nous avons considéré que nous réalisations notre estimation en procédant à des tirages de $P$ *indépendants*. Cependant, nous allons voir que dans certains cas, cette contrainte peut s'avérer pénible en pratique. En effet, la méthode de Monte-Carlo requiert que le nombre $n$ de tirages de $P$ auxquels on procède soit *grand* [penser à $n = 10^6$], de sorte qu'il est important d'avoir une méthode permettant d'obtenir ces $n$ tirages pour un cout de calcul raisonnable. Or, il se trouve que pour certaines lois $P$, la méthode la plus efficace pour obtenir un grand nombre de tirages fournit des échantillons présentant une certaine dépendance... !

Bien entendu, quelle que soit la loi $P$ qu'on sait simuler, il est toujours possible de procéder à $n$ tirages de manière indépendante : il suffit pour ce faire de simuler un tirage, puis d'effacer complètement notre mémoire, de procéder à un nouveau tirage avec un aléa indépendant, et ainsi de suite $n$ fois. Ce faisant, le cout de calcul $t(n)$ requis pour procéder à ces $n$ tirages est égal à $nt(1)$. Le point ici est que, dans certaines situations, le cout $t(1)$ par simulation sera très grand [penser, par exemple, à un temps de calcul de 20 s pour obtenir chaque simulation !], ce qui rendrait la méthode de Monte-Carlo prohibitive... Néanmoins, dans ces situations-là, il arrive fréquemment qu'on dispose d'une méthode fournissant une *série* de $n$ simulations de $P$ à cout $t(1) + \alpha(n-1)$, où la constante $\alpha$, qui représente le cout par simulation *supplémentaire*, sera beaucoup plus petite que $t(1)$ ! Mais cela se paye alors par le fait qu'on aura une certaine *dépendance* entre nos simulations...

Concrètement, ces séries de simulations vont s'obtenir à partir d'une *chaine de Markov* sur $\Omega$ dont $P$ sera la *mesure d'équilibre* (d'où les rappels de la section précédente). L'idée centrale est que, pour certaines lois de probabilité $P$, il est difficile de simuler "directement" $P$, mais assez facile de trouver une chaine de Markov qui admette $P$ pour loi d'équilibre, avec qui plus est de bonnes propriétés de mélange. Dès lors, si nous partons d'une mesure de départ $\mu_0$ arbitraire (typiquement, un Dirac en un point arbitraire), on peut lancer une simulation de la chaine de Markov issue de $\mu_0$ sur un nombre de pas $T_{\text{in}}$ soit suffisamment grand (penser à $T_{\text{in}} = 10^6$) pour que, en pratique, la loi de $X_{T_{\text{in}}}$ soit complètement indistinguable de $P$! On voit alors que simuler cette unique variable $X_{T_{\text{in}}}$ requiert un cout de calcul important ; mais qu'en revanche, sitôt celle-ci obtenue, il suffit d'ajouter $K$ pas à notre chaine de Markov — ce qui est rapide — pour obtenir $k$ variables supplémentaires $X_{T_{\text{in}}+1}, \dots, X_{T_{\text{in}}+K}$ qui seront elles aussi de loi $P$... mais qui seront corrélées à $X_{T_{\text{in}}}$ (et corrélées les unes aux autres), en particulier lorsque les indices de temps correspondants sont proches à l'échelle du temps de mélange de la chaine !

Cette dépendance entre simulations semble à première vue ruiner tout espoir d'appliquer la méthode de Monte-Carlo pour calculer une espérance. Mais n'oublions pas que, en pratique, nous avons supposé que notre chaine de Markov possédait de bonnes propriétés de mélange... Cela va nous assurer que, si notre nombre de simulations est nettement plus grand que le temps de mélange, on aura quand même convergence de la moyenne empirique des simulations vers l'espérance recherchée, et même, d'une façon où on saura déterminer un intervalle de confiance.

L'efficacité par simulation sera alors inférieure à l'inverse de la variance de notre v.a. (à cause des corrélations) ; mais l'efficacité *tout court* n'en sera pas moins très supérieure à ce qu'on aurait obtenu en tentant d'obtenir des simulations rigoureusement indépendantes... !

## Principe général de la méthode

Pour estimer une espérance par Monte-Carlo à l'aide d'une chaine de Markov, le principe est le suivant :

!!

*Principe* (IX) (Méthode de Monte-Carlo par chaine de Markov). Soit $P$ une loi de probabilité sur un $\Omega$ pour laquelle on connait un noyau markovien $k$ ayant $P$ pour loi d'équilibre, qu'on sait simuler ; et soit $f : \Omega \to \mathbb{R}$ une fonction qu'on sait calculer, telle que $f(P)$ soit d'intégrabilité $\mathrm{L}^1$. Alors la méthode de Monte-Carlo pour déterminer $\mathbb{E}(f(P))$ à partir du noyau $k$ consiste à choisir une mesure de départ $\mu_0$ arbitraire, un entier $T_{\text{in}}$ appelé *durée de préchauffage*[‡‡] et un entier $n$ correspondant à un nombre de simulations ; à simuler $(T_{\text{in}} + n)$ pas de la chaine de Markov de noyau $k$ issue de $\mu_0$ (nous noterons $x_{0\checkmark}, \dots, x_{(T_{\text{in}}+n-1)\checkmark}$ les valeurs obtenues dans cette simulation), et à estimer $E(f(P))$ par

$$\frac{1}{n}\sum_{i=0}^{n-1} f(x_{(T_{\text{in}}+i)\checkmark}). \qquad \text{(IY)}$$

♧

Sous réserve que le noyau markovien soit ergodique, cette méthode fournit effectivement un estimateur convergent — et l'introduction de la phase de préchauffage permet d'améliorer sa qualité :

**Théorème (IZ).** *Supposons que le noyau markovien soit ergodique. Alors, à choix fixés de $\mu_0$ et de $T_{\text{in}}$, l'estimateur de Monte-Carlo (dont la réalisation est) donné par* (IY) *est (fortement) convergent.*

*En outre, si $T_{\text{in}}$ est choisie nettement plus grande que le temps de mélange typique de la chaine*[∗]*, l'estimateur sera sans biais en pratique (au sens où son biais sera beaucoup plus petit que le niveau de précision auquel on considèrera nos résultats).* ◇

## Intervalles de confiance

Savoir qu'on a un estimateur convergent dans le cas d'une simulation par chaine de Markov est une bonne chose ; mais, de même que pour l'application des méthodes de Monte-Carlo aux calculs d'espérance, nous aimerions être capables d'associer à nos estimations des intervalles de confiance (asymptotiques). Nous allons expliquer ici comment faire.

Sans surprise, une première condition pour déterminer de tels intervalles de confiance est que $f(P)$ soit d'intégrabilité $\mathrm{L}^2$ : en effet, il n'y a pas de raison qu'on puisse faire mieux dans le cas dépendant que dans le cas indépendant !

En outre, de même que la condition $\mathrm{L}^1$ n'est, en situation de dépendance, pas suffisante à elle seule pour assurer la convergence de l'estimateur [car il faut aussi de l'ergodicité], il faut aussi renforcer la condition $\mathrm{L}^2$ pour obtenir des intervalles

[‡‡]. En anglais : *burn-in time.*

[∗]. En réalité, je passe un certain nombre de conditions sous le tapis :

de confiance normaux : et le renforcement requis est alors plus fort que la simple ergodicité. L'idée est que, non seulement il faut que la chaine de Markov finisse par exeplorer tout $\Omega$ (ce qu'assure l'ergodicité), mais aussi qu'elle "oublie" suffisamment vite d'où elle est partie pour que deux valeurs correspondant à des temps suffisamment éloignés soient, en pratique, "indépendants". On parle, dans le jargon, de « condition de mélange » : l'idée étant que, si on "touille" la chaine de Markov suffisamment longtemps, au bout d'un moment les valeurs issues d'un même point de départ (mais ayant évolué indépendamment) vont se retrouvée "bien mélangées" !

Il existe une foultitude de conditions de mélange, dont la description nous éloignerait largment du cadre de ce cours... Mentionnons tout de même, juste pour la culture, une condition classique [†] qui, à la fois, est suffisante (quoique non nécessaire) pour faire fonctionner le cas $\mathrm{L}^2$, et se trouve effectivement vérifiée dans un grand nombre de cas intéressants (quoique pas tous) :

**Définition (JA).** On dit qu'un noyau Markovien $k(\bullet)$ sur une chaine de Markov $\Omega$, admettant une loi d'équilibre $\mu_{\text{éq}}$, est $\rho^*$-*mélangeant* [‡] lorsqu'il existe une constante $R < 1$ telle que, pour la chaine de Markov stationnaire (autrement dit, issue de $\mathrm{Loi}(X_0) = \mu_{\text{éq}}$), on ait :

$$\forall f, g : \Omega \to \mathbb{R} \qquad |\mathrm{Cov}(f(X_0), g(X_1))| \leqslant R \,\mathrm{Var}^{1/2}(f(X_0)) \,\mathrm{Var}^{1/2}(g(X_1)) \qquad \text{(JB)}$$

(où il est sous-entendu qu'on ne considère que les cas où $f(X_0)$ et $g(X_1)$ sont d'intégrabilité $\mathrm{L}^2$) : autrement dit, l'inégalité de Cauchy-Schwartz entre une v.a. $X_0$-mesurable et une v.a. $X_1$-mesurable peut toujours être améliorée par un facteur $R$, indépendamment du choix de $f(\bullet)$ et de $g(\bullet)$. ♡

Lorsqu'on a une condition de mélange suffisamment forte (par exemple le $\rho^*$-mélange évoqué ci-dessus), alors les fluctuations de l'estimateur sont asymptotiquement normales :

**Théorème (JC).** *Soit $\Omega$ un espace sur lequel on se donne un noyau markovien $k(\bullet)$ ergodique* [§] *; soit $P_{\text{éq}}$ supposé suffisamment mélangeant ; et soit $f : \Omega \to \mathbb{R}$ de classe $\mathrm{L}^2$ (et supposée non constante). Alors l'estimateur de Monte-Carlo $\hat{\mu}^{(n)}$ de $\mathbb{E}^{\omega \sim P_{\text{éq}}}(f(\omega)) =: \mu$ fourni par la formule (IY) a des fluctuations asymptotiquement normales : à savoir, il existe une constante $\sigma_k(f) \in ]0, \infty[$ telle que*

$$\mathrm{Loi}\left( \frac{\hat{\mu}^{(n)} - \mu}{\sigma_k(f) n^{-1/2}} \right) \overset{n\to\infty}{\to} \mathrm{Normale}(0, 1). \qquad \text{(JD)}$$

♢

*Remarque* (JE). On notera que $T_{\text{in}}$ n'intervient pas dans ce théorème : de fait, il est facile de constater que le préchauffage ne fait que modifier la somme définissant $\hat{\mu}^{(n)}$ par une constante aléatoire, de sorte que, à l'échelle considéré, il disparait dans l'asymptotique ! ♧

On peut même préciser la valeur de la constante $\sigma_k(f)$ :

**Proposition (JF).** *La constante $\sigma_k(f)$ qui apparait dans le théorème (JC) est donnée par*

$$\sigma_k(f)^2 = \sum_{t\in\mathbb{Z}} \mathrm{Cov}(f(X_s), f(X_{s+t})) = \mathrm{Var}^{\omega\sim P_{\text{éq}}}(f(\omega)) + 2 \sum_{t=1}^{\infty} \mathrm{Cov}(f(X_0), f(X_t)), \qquad \text{(JG)}$$

[†]. Incidemment, l'auteur du présent polycopié a, dans le passé, écrit quelques travaux de recherche au sujet de cette condition de mélange ! ☺

[‡]. Ici, '$\rho^*$' n'est pas un objet mathématique : c'est juste une partie du mot « $\rho^*$-mélange » ! De même qu'on parle de « $p$-valeurs » ou de « $\sigma$-additivité », par exemple.

[§]. Dans ce contexte, « ergodique » doit juste être compris comme « possédant une loi d'équilibre ».

*où $(X_u)_{u\in\mathbb{Z}}$ désigne la chaine de Markov stationnaire (i.e., la chaine de Markov issue de $\mathrm{Loi}(X_0) = P_{\text{éq}}$, qui est invariante par translation est qu'on peut alors indexer par $t \in \mathbb{Z}$), et $s \in \mathbb{Z}$ est un temps de référence arbitraire.* ◇

Cependant, cette expression "théorique" de $\sigma_k(f)$ n'est guère utile en pratique pour l'estimer : mieux vaut pour ce faire utiliser le théorème **??** ci-dessous.

*Remarque* (JH). Cette convergence en $n^{-1/2}$, permet alors de définir une notion d'efficacité de la même façon que nous avions fait pour comparer les méthodes d'estimation d'une espérance dans le cas indépendant. On peut ainsi comparer l'efficacité entre une méthode de Monte-Carlo par chaine de Markov et une méthode par simulations indépendantes, ou entre deux méthodes de Monte-Carlo par chaines de Markov [¶]. ♧

Avoir cette convergence normale est une bonne chose, mais encore faudrait-il être capable d'estimer convenablement $\sigma_k(f)$ pour en tirer des intervalles de confiance ! C'est ce que nous permet de faire le théorème suivant, dont la logique sera expliquée juste après :

**Théorème (JI)** (Intervalles de confiance pour les méthodes MCMC)**.** *Soit $d \in [\![2, \infty[\![$ un entier fixé. Pour $n$ de la forme $dm$, écrivons notre estimateur $\hat{\mu}^{(n)}$ comme la moyenne de $d$ estimateurs $\hat{\mu}_r^{(n')}$ pour $r \in [\![0, d[\![$, où $\hat{\mu}_r^{(m)}$ est l'estimateur de Monte-Carlo où on aurait utilisé $m$ simulations* avec un temps de préchauffage de $T_{\text{in}} + rm$, *autrement dit, en faisant la moyenne des simulations $T_{\text{in}} + rm$ à $T_{\text{in}} + (r+1)m - 1$ :*

$$\hat{\mu}^{(dm)} \underset{\text{déf}}{=} \frac{1}{d}\sum_{r=0}^{d-1} \hat{\mu}_r^{(m)}, \text{ où} \qquad \hat{\mu}_r^{(m)} := \frac{1}{m}\sum_{i=rm}^{(r+1)m-1} X_{T_{\text{in}}+i}. \tag{JJ}$$

*Alors, notant $\hat{\sigma}_d^{(n)}$ la quantité*

$$\hat{\sigma}_d^{(n)} = m^{1/2}\,\mathrm{var}_{\text{emp}}^{1/2}(\hat{\mu}_r^{(m)})_{r\in[\![0,d[\![}, \tag{JK}$$

*on obtient un intervalle de confiance de niveau de risque $\alpha$ pour $\mu$ en considérant*

$$[\hat{\mu}^{(n)} \pm q_\alpha n^{-1/2}\hat{\sigma}_d^{(n)}], \tag{JL}$$

*où $q_\alpha$ désigne le seuil tel que $\mathbb{P}(|\mathrm{T}_{\text{St}}(d-1)| > \alpha) = \alpha$, sachant que la loi de probabilité $\mathrm{T}_{\text{St}}(d-1)$ se réfère à la « loi $t$ de Student à $(d-1)$ degrés de liberté ».* ◇

*Remarque* (JM). Expliquons la logique du théorème (JI). L'idée est de se dire que, asymptotiquement :

- D'une part, les $\hat{\mu}_r^{(m)}$ se comporteront approximativement comme des lois $\mathrm{Normale}(\mu, m^{-1/2}\sigma_k(f))$ : c'est une conséquence directe du théorème (JC), puisque les $\hat{\mu}_r^{(m)}$ sont des estimateurs de Monte-Carlo associés au noyau markovien $k(\bullet)$, et qu'on considère un régime où $m$ tend vers l'infini !
- D'autre part, les $\hat{\mu}_r^{(m)}$ associés à des valeurs différentes de $r$ deviennent (asymptotiquement) *indépendants* entre eux. Cela est lié au fait que la chaine est mélangeante : informellement, cette propriété de mélange fait que la dépendance entre $X_t$ et $X_{t+u}$ devient négligeable dès lors que $u$ dépasse une certaine "portée maximale" $u_{\text{port}}$. Or, lorsque $m$ devient beaucoup plus grand

[¶]. Ou même pour jauger la performance d'une méthode MCMC incluant une technique de réduction de la variance : car de fait, il existe des méthodes de réduction de la variance pour les méthodes MCMC, même si le cours n'abordera pas ce sujet ☺

que $u_{\text{port}}$, la contribution à $\hat{\boldsymbol{\mu}}_r^{(m)}$ des indices de temps qui sont à distance moins de $u_{\text{port}}$ des indices de temps utilisés pour les autres $\hat{\boldsymbol{\mu}}_{r'}^{(m)}$ devient négligeable, d'où l'indépendance asymptotique.

Dès lors, notre estimateur de Monte-Carlo $\hat{\boldsymbol{\mu}}^{(n)}$ devient la moyenne de $d$ variables Normale$(\mu, m^{-1/2}\sigma_k(f))$ indépendantes. Dans ce cas, la moyenne de ces $d$ variables normales indépendantes constitue bien un estimateur de leur espérance commune, dont l'écart-type sera $d^{-1/2} \times m^{-1/2}\sigma_k(f) = n^{-1/2}\sigma_k(f)$, comme prévu. Pour estimer cet écart-type, on pourrait approcher $m^{-1/2}\sigma_k(f)$ par l'écart-type empirique des $\hat{\boldsymbol{\mu}}_r^{(m)}$, ce qui revient à estimer $\sigma_k(f)$ par $\hat{\boldsymbol{\sigma}}_d^{(n)}$ (ce qui est l'explication de la formule (JK)). Simplement, comme nous avons utilisé un nombre $d$ fixé de v.a. normales pour estimer leur espérance commune, nous n'avons pas de convergence asymptotique de l'estimateur $\hat{\boldsymbol{\sigma}}_d^{(n)}$ vers $\sigma_k(f)$ lorsque $m$ tend vers l'infini ! C'est pour cela que nous devons utiliser une formule qui nous donne un intervalle de confiance non asymptotique pour estimer l'espérance d'un ensemble de $d$ v.a.i.i.d. normales, laquelle fait intervenir la loi de Student. ♧

L'intervalle de confiance fourni par le théorème (JI) n'est pas tout à fait optimal, même asymptotiquement, à cause du fait que l'estimateur intermédiaire $\hat{\boldsymbol{\sigma}}_d^{(n)}$ utilisé dans cette procédure n'est pas convergent : en exploitant un peu plus habilement les simulations permettrait, on pourrait en réalité trouver un estimateur convergent de $\sigma_k(f)$, qui permettrait alors de trouver un intervalle de confiance plus précis. Néanmoins, si $d$ a été choisi suffisamment grand, la différence est infime en pratique :

**Proposition (JN).** *Si on a pris $d \geqslant 25$, alors dans tous les cas pratiques considérés, la marge d'erreur de l'intervalle de confiance fournie par la formule* (JL) *s'écartera de moins de* 10 % *(en valeur relative) de la marge d'erreur $\tilde{q}_\alpha \sigma_k(f)$ qu'aurait fournie un "oracle" ayant accès à la véritable valeur de $\sigma_k(f)$, où $q_\alpha$ désigne le quantile relatif à la loi normale standard, caractérisé par $\mathbb{P}\big(|\text{Normale}(0,1)| > \tilde{q}_\alpha\big) = \alpha$.*

*Par conséquent, on pourra en pratique appliquer l'intervalle de confiance* (JL) *avec $d = 25$ sans se poser trop de questions sur la finesse de cet intervalle : et si on a vraiment beaucoup de simulations, on pourra même monter à $d = 50$, voire $d = 100$, stade auquel ce sera pratiquement comme si on avait utilisé un intervalle de confiance oracle !* ♢

*Remarque* (JO). Attention ! En pratique, plus on choisit $d$ grand, plus l'intervalle de confiance (JL) est asymptotiquement précis... Mais plus la convergence met de temps à être atteinte ! En pratique, pour pouvoir utiliser convenablement l'estimateur (JL), il faut que $m$ soit nettement plus grand que le "temps de mélange typique" de la chaine de Markov [un concept que je n'ai pas encore défini à ce stade, NdA !], ce qui demande donc que $n$ doit plus grand que $d$ fois ce temps typique : d'où le risque à utiliser un $d$ trop grand tant que $n$ reste modéré. Il est donc préférable de commencer plus modestement avec (par exemple) $d = 10$, et, si les simulations semblent témoigner que $n / d$ est nettement plus grand que le temps de mélange [ce que vous ne savez pas encore comment faire à ce stade, NdA], alors on pourra envisager des valeurs de $d$ plus élevées. ♧

## 6.3 Chaine de Metropolis-Hastings

Les chaines de Markov de Metropolis-Hastings sont des façons de fabriquer un noyau markovien qui ait une loi d'équilibre prescrite, lorsqu'on ne saurait pas échantillonner directement cette loi de façon efficace. Les situations où on recourra à cette méthode sont celles où on est en train de considérer une loi de probabilité $P$, définie sur un espace $\Omega$, qui :

— Possède une densité par rapport à une mesure de référence simple $\mu_{\text{réf}}$ sur $\Omega$ (typiquement, une mesure de Lebesgue ou de comptage) ;

— Cette densité est connue "explicitement" (au sens où on sait la calculer numériquement, de façon efficace et précise, en n'importe quel point de $\Omega$) *à constante multiplicative près*. Dans la suite, nous noterons par $f(\omega)$ la valeur (à constante multiplicative près) de la densité en $\omega$.

C'est une situation qui se rencontre réellement "dans la vraie vie", notamment pour les deux usages suivants :

— Postériore d'une analyse statistique bayésienne (cas le plus fréquent en pratique) ;

— Loi de distribution au sein d'un ensemble de micro-états en physique statistique (que nous utiliserons dans le cadre du recuit simulé au chapitre **??**).

En pratique, nous noterons que, sous réserve que notre densité (à constante près) soit bornée explicitement, on dispose déjà de la méthode de rejet qui permet d'échantillonner (sans passer par une chaine de Markov, qui plus est) la loi de probabilité considérée. Néanmoins, pour qu'une méthode de rejet soit efficace, il y a besoin qu'on n'ait pas besoin de faire "trop" d'essais avant d'accepter une simulation : typiquement, une méthode de rejet où on n'accepterait qu'un essai sur $10^9$ serait totalement inopérante en pratique ! ⁚

Or, il se trouve que, lorsque la dimension de l'espace $\Omega$ augmente, le taux d'acceptation des méthodes de rejet qu'on observe *en pratique* siminue dramatiquement : en fait, ce taux a tendance à être exponentiellement faible en la dimension [‖]. Par conséquent, dès lors qu'on est en train de vouloir échantillonner une loi décrite par une densité en dimension 100 [**], on n'a aucune chance de s'en sortir avec une simulation par rejet ! À l'inverse, les méthodes de Metropolis-Hastings sont nettement moins sensibles aux problèmes de dimensionnalité, et sont donc à privilégier dans un tel cadre.

Décrivons donc ce fameux noyau de Metropolis-Hastings ! On a besoin d'un outil auxiliaire : un noyau markovien $k_{\text{réf}}$ pour lequel la mesure de référence $\mu_{\text{réf}}$ *réversible* (par juste invariante, attention !). En général, vu que la mesure de référence est très simple, on peut très facilement proposer un tel noyau markovien de référence. Avant le poursuivre, je mentionne comment on peut fabriquer un noyau markovien réversible dans le cas de la mesure de Lebesgue sur $\mathbb{R}^d$, qui est particulièrement fréquent :

**Lemme (JP).** *Soit $Q$ n'importe quelle loi de probabilité* symétrique *sur $\mathbb{R}^d$, autrement dit telle que, pour $V$ une v.a. suivant la loi $Q$, $-V$ suit également la loi*

[‖]. On pourrait donner divers arguments tendant à "démontrer" ce fait : je le ferai peut-être un jour en annexe dans une version ultérieure de ce cours ⁛

[**]. Et cela peut réellement arriver, si on est amené à introduire un modèle bayésien possédant de nombreux paramètres avec des interactions compliquées : c'est même particulièrement fréquent en médecine ou en écologie, par exemple !

$\boldsymbol{Q}$ [††]. *Alors la chaine de Markov où l'on passe de $X_t$ à $X_{t+1}$ en ajoutant un pas de loi $X$ (autrement dit, le noyau markovien décrit par $k(x, \mathrm{d}y) = \mathbb{P}(x + Q \in \mathrm{d}y)$) est réversible par rapport à la mesure de Lebesgues sur $\mathbb{R}^d$.* ◇

*Remarque* (JQ). Attention ! Il est important que $Q$ soit indépendant du point de départ ! Une chaine de Markov telle que $\mathrm{Loi}(X_{t+1} - x \mid X_t = x)$ soit symétrique pour tout $x$ n'aura pas, en général, la propriété d'invariance pour la mesure de Lebesgue ! ♧

Bref ; supposons donc donné nore noyau $k_{\text{réf}}$ ayant $\mu_{\text{réf}}$ pour mesure réversible. Alors l'algorithme de Metropolis-Hastings consiste à considérer le noyau markovien suivant :

**Définition (JR)** (Noyau de Metropolis-Hastings)**.** Le noyau de Metropolis-Hastings est défini par

$$k_{\text{mh}}(x, \mathrm{d}y) = \begin{cases} k_{\text{réf}}(x, \mathrm{d}y) & \text{si } y \neq x \text{ et } f(y) \geqslant f(x) ; \\ f(y) \,/\, f(x) \times k_{\text{réf}}(x, \mathrm{d}y) & \text{si } y \neq x \text{ et } f(y) < f(x) ; \\ 1 - \int_{y \in \Omega \setminus \{x\}} \big( (k(y) \,/\, k(x)) \wedge 1 \big) k_{\text{réf}}(x, \mathrm{d}y) & \text{si } y = x. \end{cases} \tag{JS}$$

♡

Bien que le dernier cas de la définition du noyau de Metropolis-Hastings ait l'air très compliqué, il n'est que le complément à 1 des deux cas précédents ; et de la sorte, il est en fait très facile de simuler un pas de ce noyau :

! **Procédure (JT)** (Simulation d'un pas de Metropolis-Hastings)**.** Un point $x \in \Omega$ étant donné, on simule la loi donnée par le noyau $k(x)$ (autrement dit, la loi de la prochaine étape de la chaine de Metropolis-Hastings) de la façon suivante :

1. On tire un candidat $y_{\text{cand}}$ selon le noyau de référence $k_{\text{réf}}(x)$ (autrement dit, c'est un tirage ce que donnerait la chaine de Markov pour le noyau de référence).
2. Si $f(y_{\text{cand}}) \geqslant f(x)$, on renvoie $y_{\text{cand}}$.
3. Sinon, on calcule le ratio $f(y_{\text{cand}}) \,/\, f(x) =: p$ (qui est nécessairement dans $[0, 1[$, et donc assimilable à une probabilité). Puis on procède à ce qui suit.
4. On tire une loi de Bernoulli $\boldsymbol{B}$ de paramètre $p$. [Typiquement, on le fera à partir d'une loi $\mathrm{Unif}^{\text{me}}(0, 1)$].
5. Si $\boldsymbol{B}$ vaut VRAI (ce qui arrive avec probabilité $p$), on renvoie $y_{\text{cand}}$ ;
6. Si, à l'inverse, $\boldsymbol{B}$ vaut FAUX, on renvoie $x$. ♡

*Remarque* (JU). Cet algorithme est assez intuitif à comprendre : en effet, il exprime le fait qu'on cherche à échantillonner une mesure qui amplifie la mesure de référence d'un facteur $f$ : par rapport à $\mu_{\text{réf}}$, notre mesure $\boldsymbol{P}$ va donc d'autant plus privilégier telle ou telle zone de $\Omega$ que $f(\bullet)$ y est grande. Si la transition que nous aurions proposée pour $\mu_{\text{réf}}$ nous aurait conduit à *augementer* la valeur de $f$, alors elle est en quelque sorte "encore mieux adaptée" à $\boldsymbol{P}$ qu'elle ne l'était à $\mu_{\text{réf}}$, donc nous

[††]. Par exemple, entre autres, $\boldsymbol{Q}$ peut être la loi de n'importe quel vecteur gaussien centré, ou encore, la loi consistant à tirer uniformément un vecteur de base parmi $\{\vec{e}_1, \dots, \vec{e}_d\}$, puis à multiplier ce vecteur par un nombre tiré uniformément entre $-1$ et $+1$.

reprenons cette transition systématiquement ! Si, en revanche, la transition proposer aurait fait diminuer $f$, alors nous allons être d'autant plus réticients à la reprendre qu'elle aurait fait dimuner $f$ de beaucoup : et très précisément, nous l'acceptons averc une probabilité qui correspond au *ratio* entre la valeur de $f$ au point où cette transition nous ferait arriver et la valeur de $f$ au point dont nous sommes partis.

Ce qui n'a rien d'intuitif, car contre, c'est, d'une part, prendre *précisément* le ratio des $f$ comme probabilité d'acceptation va bien nous conduire à la mesure d'équilibre désirée (après tout, cela aurait pu être une fonction plus compliquée du ratio, voire quelque chose dépendant aussi de $x$ !) ; et d'autre part, que lorsque notre transition est rejetée, il faut en fait rester sur place [‡‡]. ♧

> *Remarque* (JV). À noter que le fait que $f$ n'intervienne que via le *ratio* $f(y_{\text{cand}})/f(x)$ est satisfaisant, dans la mesure où nous savons que deux fonctions $f$ qui diffèrent par une constante multiplicative décrivent en fait la même loi $\boldsymbol{P}$ : le fait que ce soit le ratio qui intervienne nous dit que, en fait, la chaine de Metropolis-Hastings sera la même dans les deux cas ! Il n'y a donc aucune importance quant au choix de la constante multiplicative dont nous disposons comme degré de liberté pour définir la fonction $f$. ♧

Comme annoncé, la chaine de Metropolis-Hastings a bien $\boldsymbol{P}$ pour mesure d'équilibre ; en fait, c'est même une mesure invariante de la chaine :

**Théorème (JW).** *Avec les notations ci-dessus, $\boldsymbol{P}$ est une loi réversible pour le noyau de Metropolis-Hastings défini par la définition (JR) (ou, si vous préférez, celui simulé à l'aide de la procédure (JT)).* ◇ !

[‡‡]. Bien noter que ce n'est pas comme dans la simulation par rejet, où on refait des essais jusqu'à avoir une acceptation : ici, si notre loi de Bernoulli vaut FAUX, on posera $X_{t+1} = X_t$ !

# Troisième partie

# Application aux processus aléatoires

# Chapitre 7

# Initiation aux processus aléatoires

## 7.1 Généralités sur les processus aléatoires

**Définition (JX).** Pour $E$ un espace métrique, une fonction $f\colon \mathbb{R}_+ \to E$ est dite *càdlàg* (**c**ontinue **à** **d**roite avec **l**imite **à** **g**auche [*]) si $f(t) = \lim_{\substack{u\to t\\ u>t}} f(u)$ pour tout $t \geqslant 0$, et que pour tout $t > 0$, $\lim_{\substack{u\to t\\ u<t}} f(u)$ existe dans $E$ (cette limite étant alors notée $f(t-)$). Dans la suite, nous appellerons *trajectoire à valeurs dans $E$* une fonction càdlàg de $\mathbb{R}_+$ dans $E$. ♡

**Définition (JY).** Pour $E$ un espace métrique séparable, un *processus aléatoire à valeurs dans $E$* est une variable aléatoire qui prend ses valeurs dans l'esapce des trajectoires de $\mathbb{R}_+$ dans $E$ ; autrement dit c'est une trajectoire aléatoire $(Z_t)_{t\geqslant 0}$. ♡ !

*Remarque* (JZ). Pour traiter la trajectoire $(Z_t)_{t\geqslant 0}$ comme une variable aléatoire unique, il faut définir une topologie sur l'espace des trajectoires, ce qui soulève quelques problèmes. On pourrait utiliser la topologie de la convergence simple, pour laquelle la notion de convergence en loi est simplement la convergence au sens des marginales fini-dimensionnelles dont nous parlerons dans la suite du cours. Cependant cette topologie est mauvaise pour faire des probabilités, parce qu'elle n'est pas « de Luzin », ce qui signifie en gros qu'on ne peut pas simuler les trajectoires pour la loi-limite d'une façon compatible avec la topologie. En outre, ce n'est pas non plus une topologie satisfaisante du point de vue analytique : par exemple, le fait d'être une trajectoire continue ne correspond pas à une partie borélienne de cette topologie !

Il se trouve heureusement qu'il existe d'autres topologies présentant de bonnes propriétés probabilistes et analytiques, en particulier la *topologie de Skorokhod* qui est celle qu'on utilise généralement pour parler des processus aléatoires. Mais cette topologie est assez compliquée à décrire et à manipuler... Nous ferons donc le choix dans ce cours de ne pas nous encombrer de ces considérations techniques, mais il faudra garder à l'esprit que les propriétés trajectorielles des processus aléatoires n'ont en fait de sens que pour la topologie de Skorokhod. ♧

**Définition (KA).** Un processus aléatoire $(Z_t)_{t\geqslant 0}$ est qualifié de *markovien (homo-* !

[*]. Incidemment, nonobstant l'origine francophone de l'adjectif « càdlàg », on dit aussi « càdlàg » en anglais ☺

*gène)* quand la prédiction de son futur au-delà d'un instant $t_1$ ne demande que de connaitre la valeur de $Z_{t_1}$ et pas l'ensemble du passé de la trajectoire jusqu'à cet instant, et qu'en outre la loi conditionnelle du futur est invariante par translation en temps. Formellement, cela signifie qu'on a :

$$\mathrm{Loi}((Z_t)_{t>t_1} \mid (Z_t)_{t\leqslant t_1}) = \mathrm{Loi}((Z_t)_{t>t_1} \mid Z_{t_1}) \tag{KB}$$

et

$$\mathrm{Loi}((Z_{t_1+u})_{t\geqslant 0} \mid Z_{t_1} = z) = \mathrm{Loi}((Z_u)_{u\geqslant 0} \mid Z_0 = z). \tag{KC}$$

L'ensemble (KB)-(KC) est appelé « propriété de Markov (faible) ». ♡

! **Définition (KD).** Un temps aléatoire $T$ est qualifié de *temps d'arrêt* quand on peut déterminer le moment de ce temps à partir de la seule connaissance du passé de la trajectoire avant ce moment, sans avoir à savoir ce qui se passe ensuite. Informellement, il s'agit d'une règle pour crier « STOP ! » à un certain moment suivant une règle déterministe dépendant de la trajectoire, *trajectoire qu'on découvre au fur et à mesure.* ♡

*Exemple* (KE).

— Si $Z$ est une processus aléatoire à valeurs dans $\mathbb{R}$, « le premier instant $T$ tel que $|Z_T - Z_{T-1}| \geqslant 3$ » définit un temps d'arrêt.

— En revanche, « l'instant $T \in [0,1]$ auquel $Z_t$ atteint sa plus grande valeur » ne définit *pas* un temps d'arrêt (sauf exception), car on ne peut pas savoir que $Z_T$ sera la plus grande valeur tant qu'on ne connait pas $Z_t$ sur $(T,1]$.

♧

! **Théorème (KF).** *(Sous certaines hypothèses techniques qui seront toujours satisfaites ici), tout processus aléatoire vérifiant la propriété de Markov faible vérifie aussi la* propriété de Markov forte. *Cette propriété signifie que* (KB) *s'applique aussi à un temps d'arrêt : pour tout temps d'arrêt $T$,*

$$\mathrm{Loi}\big((Z_t)_{t>T} \;\big|\; T=\tau \text{ et } (Z_t)_{t\leqslant\tau} = (z_t)_{t\leqslant\tau}\big) = \mathrm{Loi}((Z_{t-\tau})_{t>\tau} \mid Z_0 = z_\tau). \tag{KG}$$

♢

## 7.2 Processus de sauts

### Théorie des processus de sauts

! **Définition (KH).** Un processus de sauts $Z$ est un processus dont les trajectoires sont constantes par morceaux ; autrement dit, tel qu'il existe des instants (aléatoires) $0 < \tau_1 < \tau_2 < \ldots$ [†] et des valeurs (aléatoires) $y_0, y_1, y_2 \ldots \in E$ tels que

$$Z_t = \begin{cases} y_0 & \text{si } 0 \leqslant t < \tau_1 \\ y_1 & \text{si } \tau_1 \leqslant t < \tau_2 \\ y_2 & \text{si } \tau_2 \leqslant t < \tau_3 \\ \text{etc.} \end{cases} \tag{KI}$$

♡

[†]. Nous supposerons aussi que notre processus de sauts a un nombre de sauts localement fini, c.-à-d. que, lorsque $i \to \infty$, les $\tau_i$ tendent vers le supremum du domaine de définition de $Z$.

Si l'on souhaite qu'un tel processus soit markovien, cela impose des contraintes précises sur la loi des $\tau_i$ et des $y_i$. Par exemple, supposons que le processus soit toujours issu du point $y_0$, et demandons-nous à quel instant $\tau_1$ survient le premier saut. Soit $u > 0$ un délai fixé. Quelle est la loi de $\tau_1$ sachant que $\tau_1 > u$ ? Conditionner à l'évènement $\{\tau_1 > u\}$ revient à conditionner par le fait que la trajectoire jusqu'au temps $u$ soit constante à la valeur $y_0$. Mais alors, puisque notre processus est markovien, nous savons que la loi du processus à partir du temps $u$ sera la même que la loi à partir du temps 0 pour le processus issu de $y_0$. Dans le détail,

$$\mathrm{Loi}\big((Z_{u+t})_{t\geqslant 0} \mid (Z_t)_{0\leqslant t\leqslant u} = (z_t)_{0\leqslant t\leqslant u}\big) \overset{\text{p.s.}}{=} \mathrm{Loi}\big((Z_t)_{t\geqslant 0} \mid Z_0 = z_u\big), \tag{KJ}$$

d'où

$$\mathrm{Loi}(\tau_1 - u \mid Z_0 = y_0 \text{ et } \tau_1 > u) = \mathrm{Loi}(\tau_1 \mid Z_0 = y_0), \tag{KK}$$

puisque, conditionnellement à ce que $\tau_1 > u$, le premier instant de saut du processus $(Z_{u+t})_{t\geqslant 0}$ correspond à $u + t = \tau_1$, soit $t = \tau_1 - u$.

On en déduit que pour tous $u, v > 0$, $\mathbb{P}(\tau_1 > u+v) = \mathbb{P}(\tau_1 > u)\,\mathbb{P}(\tau_1 > v)$, et donc que $\tau_1$ doit suivre une loi exponentielle : c'est la classique propriété d'absence de mémoire caractérisant la loi exponentielle, qui fait pendant à l'absence de mémoire des processus markoviens (qui ne décident de leur avenir qu'en fonction de l'endroit où ils sont, pas de là par où ils sont passés).

Avec un peu de travail supplémentaire, on pourrait montrer de même que la valeur de $y_1$ est indépendante de $\tau_1$ (conditionnellement à la valeur de $y_0$), puis que, conditionnellement à la valeur de $y_1$, $(\tau_2 - \tau_1)$ doit suivre une loi exponentielle dont le paramètre ne dépend que de $y_1$, et que la valeur de $y_2$ suit une loi ne dépendant que de $y_1$ et indépendante de $(\tau_2 - \tau_1)$, etc. Finalement, on arrive à la propriété suivante :

**Théorème (KL).** *Un processus de sauts markovien sur un espace polonais $E$ peut être décrit par la donnée :*

- *D'une fonction $\lambda$ de $E$ dans $\mathbb{R}_+$, $\lambda_y$ étant appelée « l'intensité de sauts depuis la valeur $y$ » ;*
- *D'une fonction $\mu$ de $E$ dans les mesures de probabilité sur $E$, $\mu_y$ étant appelée « la mesure d'arrivée des sauts depuis la valeur $y$ ».*

*Pour de telles données des fonctions $\lambda$ et $\mu$ [‡], on peut alors définir la loi du processus $Z$ (issu d'une certaine loi pour sa valeur initiale) ainsi :*

- *Conditionnellement à $y_0$, on tire $\tau_1$ et $y_1$ indépendamment, selon les lois respectives $Expon(\lambda_{y_0})$ et $\mu_{y_0}$ ;*
- *Puis, conditionnellement à $y_0$, $\tau_1$ et $y_1$, on tire $\tau_2$ et $y_2$ indépendamment, selon les lois respectives $\tau_1 + Expon(\lambda_{y_1})$ et $\mu_{y_1}$ ;*
- *Etc.*

◇

*Remarque* (KM). On remarquera que la donnée de la fonction $\mu$ (et de la loi de $y_0$) est équivalente à la donnée d'une *chaine de Markov* sur $E$ : ainsi, notre processus markovien suit des valeurs correspondant à une chaine de Markov, avec des durées entre les sauts qui, conditionnellement à la donnée de cette chaine de Markov, sont indépendantes et suivent des lois exponentielles de paramètres $\lambda_{y_i}$. ♧

**Définition (KN)** (Mesure de densité de sauts)**.** En fait, plutôt que de considérer $\lambda$

[‡]. Il y a en fait aussi à vérifier une condition technique de mesurabilité des fonctions $\lambda$ et $\mu$.

et $\mu$ séparément, on introduit plutôt la mesure $\bar{\mu}(y) \coloneqq \lambda_y \cdot \mu_y$ (qui n'est plus une probabilité en général), qui est appelée la *densité de sauts* depuis le point $y$. (Cette mesure est homogène à l'inverse d'un temps). Formellement, on a alors la relation suivante :

$$\forall y' \neq y \quad \mathrm{d}\mathbb{P}(Z_{t+\mathrm{d}t} = y' \mid Z_t = y) = \mathrm{d}\mu_y(y')\mathrm{d}t, \tag{KO}$$

qui caractérise la fonction $\mu$ et donc le processus de sauts. ♡

*Remarque* (KP). La formule (KO) caractérisant $\bar{\mu}_y$ ne définit $\bar{\mu}_y$ que sur $E \setminus \{y\}$ ; en fait, on peut attribuer une masse arbitraire au point $y$, ce qui revient à ajouter de "faux sauts" lors desquels le processus ne bouge en réalité pas. En général, on choisit donc de définir $\bar{\mu}_y$ de façon que $\bar{\mu}_y(\{y\}) = 0$. ♣

*Remarque* (KQ). La fonction $y \mapsto \lambda_y$ est presque caractérisée uniquement pour notre processus de sauts markovien (sous réserve qu'on impose $\bar{\mu}_y(\{y\}) = 0$) : en fait, $\bar{\mu}_y$ est bien définie pour tous les points $y$ qui seront effectivement visités par notre processus. Du point de vue technique, on peut définir la mesure $\nu$ sur $E$ (ou plus exactement la classe de mesures équivalentes associées à $\nu$) telle que, pour tout ensemble $A$, $\nu(A) > 0$ si et seulement s'il y a une probabilité non nulle que le processus $Z$ visite $A$ ; alors la fonction $\bar{\mu}$ est uniquement définie à un ensemble de $\nu$-mesure nulle près. ♣

### Simulation des processus de sauts

! **Principe (KR).** *La simulation d'un processus de sauts markovien est extrêmement simple, à partir du moment où l'on sait simuler les lois $\mu_y$ : il suffit de suivre la procédure expliquée ci-dessus définissant le processus en termes de $\tau_i$ et de $y_i$... On décrit l'intégralité du processus en gardant en mémoire l'ensemble des $\tau_i$ et des $y_i$ pour les $i$ tels que $\tau_i$ est dans la fenêtre de simulation qui nous intéresse. Si on souhaite tracer notre processus, il faudra toutefois prendre garde à deux petits détails :*

- *Le processus évolue par sauts, en étant* constant *entre ces sauts : il est donc hors de question de relier entre eux les $(\tau_i, y_i)$ par des segments de droites... !! Ce qu'il faut tracer sont les segments allant de $(\tau_i, y_i)$ à $(\tau_{i+1}, y_i)$ d'une part, et éventuellement (selon vos choix graphiques) les sauts allant de $(\tau_{i+1}, y_i)$ à $(\tau_{i+1}, y_{i+1})$.*
- *Si on vous demande de tracer votre processus sur une fenêtre de temps $[0, T]$, il faut prendre garde à ne pas tracer votre courbe jusqu'à $T$, ni en-deçà, ni au-delà... ! En pratique, vous aller trouver des temps de sauts $\tau_{i_{\max}}$ et $\tau_{i_{\max}+1}$ tels que $\tau_{i_{\max}} < T < \tau_{i_{\max}+1}$ : il faudra alors tracer le segment constant allant de $(\tau_{i_{\max}}, y_{i_{\max}})$ à $(T, y_{i_{\max}})$, et s'arrêter là.*

♢

### Processus de comptage de Poisson

**Définition (KS).** Le processus de comptage de Poisson d'intensité $\lambda$ est un processus de $\mathbb{R}_+$ dans $\mathbb{N}$ qui augmente par incréments d'une unité. L'occurrence (ou pas) d'incréments à différents instants est totalement indépendante, et la probabilité qu'un incrément se produise à l'instant $t$ vaut $\lambda \mathrm{d}t$. En d'autres termes, les intervalles de temps entre deux incréments successifs sont indépendants et suivent chacun une loi *Exponentielle*($\lambda$). ♡

*Remarque* (KT). Le processus de comptage de Poisson est la star des processus à sauts, de même que le mouvement brownien sera la star des processus continus. Et de même que ce sera plus le bruit blanc gaussien que le mouvement brownien lui-même qui sera l'objet fondamental, ici c'est l'ensemble des points où un saut se produit qui est le plus intéressant : c'est ce qu'on appelle un *processus ponctuel de Poisson* (d'intensité $\lambda$). ♧

# Chapitre 8

# Processus de type brownien

## 8.1 Mouvement brownien

Dans cette section, nous allons décrire la star de tous les processus aléatoires : le *mouvement brownien*.

### Heuristique et définition

Soit $P$ une loi à valeurs dans $\mathbb{R}^d$ de classe $\mathrm{L}^2$, centrée, avec pour matrice de covariance $\sigma\sigma^\mathsf{T}$. Considérons la marche aléatoire de pas $P$, c.-à-d. la suite des $S_n = \sum_{i=1}^n X_i$ où les $X_i$ sont i.i.d. de loi $P$. On se demande la loi de la trajectoire de cette marche aléatoire devient à grande échelle, c.-à-d. que pour une très grande valeur $N$ on s'intéresse au comportement de la trajectoire aléatoire $(S_{\lfloor tN \rfloor})_{t\geqslant 0}$. Comme $S_{\lfloor tN \rfloor}$ sera de l'ordre de $N^{1/2}$ d'après le théorème-limite central, on s'intéressera plus exactement à $(N^{-1/2} S_{\lfloor tN \rfloor})_{t\geqslant 0}$, abrégé dans la suite en $(B_t^N)_{t\geqslant 0}$.

À $N$ fixé, le processus $B^N$ est un certain processus aléatoire à valeurs dans $\mathbb{R}^d$. Quand $N \to \infty$, nous allons voir que ce processus converge (au sens de la convergence en loi pour les lois sur l'espace des trajectoires) vers une certaine limite, à savoir le mouvement brownien. La définition du mouvement brownien repose sur trois propriétés fondamentales qui sont le pendant de propriétés similaires pour la marche aléatoire $(S_n)_{n\in\mathbb{N}}$ (voir la preuve du théorème (KW)) :

! **Définition (KU).** On appelle *mouvement brownien $d$-dimensionnel de covariance par unité de temps $\sigma\sigma^\mathsf{T}$* le processus $(B_t)_{t\geqslant 0}$ défini de la façon suivante :
- $B_0 = 0$ avec probabilité 1 ;
- Pour $0 = t_0 < t_1 < \cdots < t_k$, les vecteurs $(B_{t_{i+1}} - B_{t_i})$ sont indépendants ;
- Le vecteur $B_{t_1} - B_{t_0}$ est un vecteur gaussien centré de matrice de covariance $(t_1 - t_0)\sigma\sigma^\mathsf{T}$.

On appelle mouvement brownien $d$-dimensionnel *standard* celui pour lequel $\sigma\sigma^\mathsf{T} = \mathbf{I}_d$. ♡

! **Théorème (KV).** *Le mouvement brownien existe, c'est-à-dire qu'on peut bien construire un objet satisfaisant la définition ci-dessus. En outre, cette définition décrit complètement la loi du mouvement brownien (« au sens des marginales fini-dimensionnelles »), c'est-à-dire qu'elle permet de connaitre la loi de $(B_{t_1}, \dots, B_{t_k})$ pour tout $k$-uplet $(t_1, \dots, t_k) \in (\mathbb{R}_+)^k$ (l'ensemble de ces lois constituant ce qu'on appelle les* marginales fini-dimensionnelles *du mouvement brownien), comme l'explicitera la proposition (LH) ci-dessous.* ◇

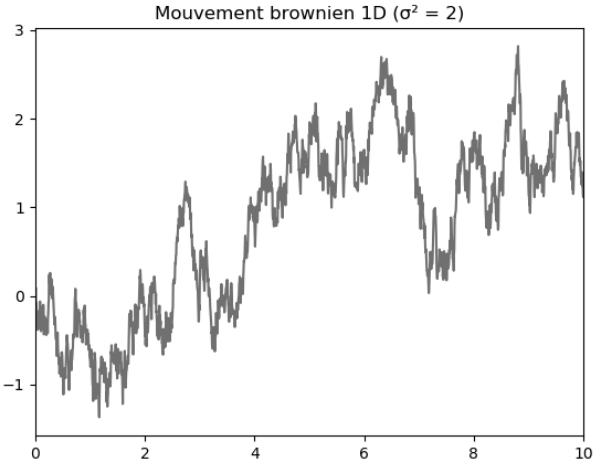


FIGURE 8.1 – Graphe d'un mouvement brownien unidimensionnel sur l'intervalle de temps $[0, 10]$.

**Théorème (KW).** *Quand $N \to \infty$, la loi de la trajectoire $(B_t^N)_{t\geqslant 0}$ converge vers le mouvement brownien (de variance par unité de temps $\sigma\sigma^\mathsf{T}$) « au sens des marginales fini-dimensionnelles », c'est-à-dire que pour tout $k$-uplet $(t_1, \dots, t_k)$, la loi de $(B_{t_1}^N, \dots, B_{t_k}^N)$ (qui est un vecteur aléatoire de l'espace fini-dimensionnel $(\mathbb{R}^d)^k$) converge vers celle de $(B_{t_1}, \dots, B_{t_k})$.* ♢

*Remarque* (KX). Le mouvement brownien est un objet *universel*, au sens où on retombe sur le même objet à la limite *quelle que soit la nature exacte de la loi de $P$*, seule la variance $\sigma\sigma^\mathsf{T}$ de celle-ci intervenant. ♣

*Exemple* (KY).

1. La figure 8.1 montre le graphe d'(un morceau de) la trajectoire d'une réalisation d'un mouvement brownien unidimensionnel. On remarquera le caractère imprévisible de la trajectoire, ainsi que son aspect très découpé.
2. La figure 8.2 montre (un morceau de) la trajectoire d'une réalisation d'un mouvement brownien 2-dimensionnel. (Comme notre dessin est lui-même 2-dimensionnel, cette fois-ci on ne peut pas représenter l'indication de temps : on a juste indiqué les points de départ et d'arrivée par les symboles respectifs '▷' et '□'). On voit que la trajectoire est complètement erratique et embrouillée... Il s'agit ici d'un mouvement brownien *anisotrope*, c.-à-d. que la matrice de covariance par unité de temps n'est pas diagonale. Cela se remarque au fait que la trajectoire "remue" manifestement beaucoup plus dans la direction de la première bissectrice que dans celle de la seconde bissectrice.

♣

## Simulation

### Généralités

*Point* (KZ) (Nécessité de discrétiser le temps). Le mouvement brownien étant un ! |

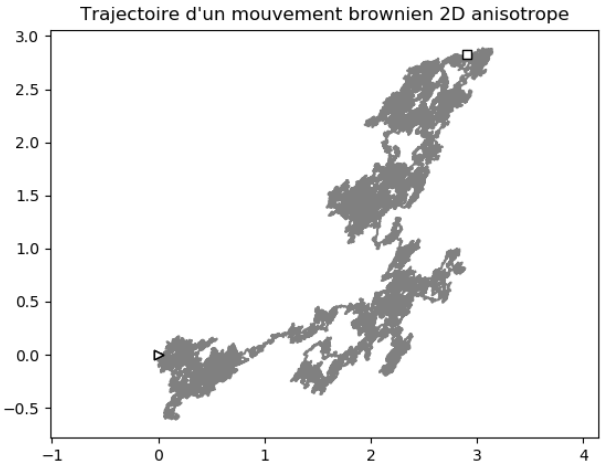


FIGURE 8.2 – Trajectoire d'un mouvement brownien 2-dimensionnel. La matrice de covariance par unité de temps a été choisie de sorte que $\mathrm{d}X_t^1$ et $\mathrm{d}X_t^2$ aient un coefficient de corrélation de $+0{,}5$.

processus à temps continu, simuler complètement sa trajectoire, même sur un intervalle de temps borné, requerrait de calculer sa valeur en un nombre infini d'instants, ce qui est évidemment impossible. On est donc obligé en pratique de *discrétiser* le temps ; c'est-à-dire qu'on ne simule pas la valeur du mouvement brownien en tous les instants, mais seulement en un nombre fini grand d'entre eux, espacés par de tout petits intervalles. Si on a besoin de connaître des aspects trajectoriels du processus, il faudra proposer une *interpolation* entre les instants de discrétisation ; c'est-à-dire, si on a simulé le processus aux temps $t_1$ et $t_1 + \varepsilon$, dire à quoi ressemble "en gros" la trajectoire complète sur $t \in [t_1, t_1 + \varepsilon]$. Dans le cas du mouvement brownien et des processus apparentés, la meilleure solution est presque toujours d'interpoler de façon linéaire, c.-à-d. de prendre la trajectoire affine $\tilde{B}_t = B_{t_1} + (t - t_1)/\varepsilon \times (B_{t_1+\varepsilon} - B_{t_1})$ pour le processus interpolé. ♧

La discrétisation la plus simple consiste à considérer les instants de discrétisation $t = 0, \varepsilon, 2\varepsilon, 3\varepsilon, \ldots$ pour $\varepsilon$ très petit. D'autres choix de discrétisation sont possibles, notamment quand il importe de connaitre la trajectoire avec plus de précision sur certaines parties spécifiques. Éventuellement, les instants de discrétisation pourront dépendre de la simulation elle-même : ainsi, si par exemple on cherche à savoir à quel instant la trajectoire dépasse un certain seuil pour la première fois, cela conduira à "zoomer" en choisissant des intervalles de temps plus petits quand la valeur du processus approche ce seuil, et le moment où cela se produira dépendra évidemment de la simulation. Le choix de la discrétisation du temps doit assurer le meilleur compromis entre deux objectifs antithétiques : d'un côté, que la simulation discrétisée (avec interpolation) ressemble le plus possible à la vraie trajectoire, ce qui suggère de prendre de très petits intervalles de temps $\varepsilon$ ; de l'autre, que le cout de simulation reste raisonnable, ce qui impose de ne pas prendre trop d'instants de discrétisation. L'équilibre précis entre ces deux contraintes dépend des exigences exactes du problème considéré ; dans ce cours, nous prendrons autour de 1 000 pas de discrétisation, ce qui est généralement une valeur appropriée.

! *Point* (LA) (Attention à l'erreur de discrétisation). Cette question de la discréti-

sation a son importance quand on voudra ensuite appliquer la méthode de Monte-Carlo à des processus aléatoires : car en plus de l'incertitude intrinsèque de la méthode de Monte-Carlo, il faudra aussi tenir compte de l'*erreur de discrétisation*, c'est-à-dire du fait que ce n'est pas tout-à-fait le vrai processus qu'on simule. Dans certains cas, l'erreur de discrétisation sera très petite par rapport à l'erreur de Monte-Carlo, au point qu'on pourra la négliger et faire comme si la simulation était exacte ; mais encore faut-il être capable de dire si on est effectivement dans un tel cas... Pour plus de rigueur, il faut disposer d'*estimations sur l'erreur* de discrétisation : nous verrons quelques telles estimations dans la suite. ♣

### Méthodes de simulation

**Définition (LB).** La *méthode d'Euler stochastique* pour simuler $(B_{t_0=0}, B_{t_1}, \dots, B_{t_k})$ (où $t_0 < t_1 < \dots < t_k$ et les $t_{i+1} - t_i$ sont très petits) consiste à simuler successivement $B_{t_1}, B_{t_2}, \dots$ en utilisant que $B_{t_{i+1}} - B_{t_i}$ est indépendant du passé, de loi Normale$\big((t_{i+1} - t_i)\sigma\sigma^\mathsf{T}\big)$. ♡ !

*Exemple* (LC). Les programmes `brownien1D.py` et `brownien2D.py` que vous trouverez sur *Arche* implémentent la méthode d'Euler stochastique pour simuler respectivement des pouvement browniens 1- et 2-dimensionnels. Leurs rendus correspondent aux figures 8.1 et 8.2 présentées un peu plus haut. ♣

**Définition (LD).** La *méthode du point médian* permet de raffiner une simulation déjà effectuée en calculant la valeur du processus en des points de discrétisation intermédiaires sans modifier la simulation des points de discrétisation déjà calculés. Celle-ci consiste à observer que la propriété de Markov implique que la loi de $(B_t)_{t\in[t_1,t_2]}$ ne dépend que de $B_{t_1}$ et $B_{t_2}$ et pas de ce qui se passe pour $t < t_1$ ou $t > t_2$, puis à utiliser la loi de $B_{(t_1+t_2)/2}$ conditionnellement à $B_{t_1}$ et $B_{t_2}$, donnée par la proposition qui suit. ♡

**Proposition (LE).** *Conditionnellement à $B_{t_1}$ et $B_{t_2}$, $B_{(t_1+t_2)/2}$ suit la loi* $(B_{t_1} + B_{t_2})/2 + \text{Normale}\big((t_2 - t_1)\sigma\sigma^\mathsf{T}/4\big)$. ◇

### Erreur de simulation

**Théorème (LF).** *Soit $(B_t)_{t\geqslant 0}$ un mouvement brownien unidimensionnel de variance par unité de temps $\sigma^2$. Conditionnellement à $B_{t_1}$ et $B_{t_2}$, la probabilité que le véritable mouvement s'écarte de plus de $\eta$ de son interpolation linéaire sur l'intervalle $[t_1, t_2]$ est majorée par* $2\exp\big(-2\eta^2/(t_2 - t_1)\sigma^2\big)$. ◇

*Remarque* (LG). La borne du théorème ci-dessus étant exponentielle, il suffit donc de prendre $t_2 - t_1$ nettement plus petit que $\eta^2/\sigma^2$ pour être assuré que la probabilité d'une erreur supérieure à $\eta$ soit infime. ♣

## Propriétés

### Loi du mouvement brownien

**Proposition (LH).** *Pour tout $(t_1, \dots, t_k) \in (\mathbb{R}_+)^k$, le $k$-uplet $(B_{t_1}, \dots, B_{t_k})$ est un vecteur gaussien centré dont la matrice de covariance est donnée (par blocs) par*

$$\text{Cov}(B_{t_i}, B_{t_j}) = (t_i \wedge t_j)\sigma\sigma^\mathsf{T}. \qquad \text{(LI)}$$

◇ !

! **Proposition (LJ).** *Si $\sigma$ est une matrice de $\mathbb{R}^{d\times d}$ et $(B_t)_{t\geqslant 0}$ un mouvement brownien $d$-simensionnel standard, alors $(\sigma B_t)_{t\geqslant 0}$ est un mouvement brownien de covariance par unité de temps $\sigma\sigma^\mathsf{T}$.* ◇

! **Proposition (LK).** *Si $(B_t^1)_{t\geqslant 0},\dots,(B_t^d)_{t\geqslant 0}$ sont des mouvements browniens unidimensionnels standard indépendants, alors $(B_t^1,\dots,B_t^d)_{t\geqslant 0}$ est un mouvement brownien $d$-dimensionnel standard.* ◇

### Propriétés du processus

! **Théorème (LL)** (Invariance d'échelle)**.** *Le mouvement brownien est invariant d'échelle d'exposant* 1/2*, au sens où pour tout $\lambda\in\mathbb{R}_+^*$, accélérer d'un facteur $\lambda$ en temps est équivalent à dilater d'un facteur $\lambda^{1/2}$ en espace :*

$$\mathrm{Loi}((B_{\lambda t})_{t\geqslant 0}) = \mathrm{Loi}((\lambda^{1/2}B_t)_{t\geqslant 0}). \tag{LM}$$

◇

! **Théorème (LN).** *Le mouvement brownien est un processus markovien. En outre, on connait exactement la loi conditionnelle de son futur :*

$$\mathrm{Loi}\big((B_{t_1+u})_{u\geqslant 0}\ \big|\ B_{t_1}=x\big) \sim \mathrm{Loi}\big((x+\tilde{B}_u)_{u\geqslant 0}\big), \tag{LO}$$

*où $(\tilde{B}_u)_{u\geqslant 0}$ suit la loi d'un (autre) mouvement brownien de même variance par unité de temps.* ◇

! **Théorème (LP).** *Pour tout $t_1\geqslant 0$, $(B_{t_1+u}-B_{t_1})_{u\geqslant 0}$ suit la même loi que $(B_u)_{u\geqslant 0}$ : on dit que le mouvement brownien est* à accroissements stationnaires. ◇

! **Théorème (LQ).** *Soit $B$ un mouvement brownien unidimensionnel. Si $T$ est un temps d'arrêt* borné*, alors $\mathbb{E}(B_T)=B_0$ : on dit que le mouvement brownien vérifie la* propriété de martingale (globale). ◇

! **Corolaire (LR).** *Soit $B$ un mouvement brownien unidimensionnel. Pour $M<\infty$, notons $\tau$ le premier instant $t$ pour lequel $|B_t|\geqslant M$. Ce processus définit un temps d'arrêt ; notons $\tilde{B}_t := B_{t\wedge\tau}$ le « mouvement brownien stoppé au temps $\tau$ ». Alors pour tout temps d'arrêt $T$ fini presque-sûrement, $\mathbb{E}(\tilde{B}_T)\underset{déf}{=}\mathbb{E}(B_{T\wedge\tau})=B_0$ : on dit que le mouvement brownien vérifie la* propriété de martingale locale. ◇

### Comportement des trajectoires

! **Théorème (LS).** *Avec probabilité* 1*, les trajectoires du mouvement brownien sont continues.* ◇

**Théorème (LT).** *En revanche, avec probabilité* 1*, les trajectoires du mouvement brownien ne sont dérivables en aucun point ! En fait, c'est même un peu plus fort que cela : (avec probabilité 1), pour tout $\varepsilon>0$, pour tout $t\in\mathbb{R}$, il est impossible de borner localement $|B_u-B_t|$ par un multiple de $|u-t|^{1/2+\varepsilon}$ au voisinage de $t$ (au sens où, pour $u\overset{>}{\to}t$ ou $u\overset{<}{\to}t$, on a $\overline{\lim}(|B_u-B_t|/|u-t|^{1/2+\varepsilon})=\infty$. Cela montre que les trajectoires du mouvement brownien ne sont pas « $(1/2+\varepsilon)$-höldériennes ». À l'inverse, pour tout $\varepsilon\in]0,1/2[$, les trajectoires du mouvement brownien sont $(1/2-\varepsilon)$-höldériennes : cela signifie que, (avec probabilité* 1*), pour tout intervalle borné $I$, on peut trouver une constante $c_I<\infty$ telle que, pour tous $t,u\in I$, on ait $|B_t-B_u|\leqslant c_I|t-u|^{1/2-\varepsilon}$.*

*En analyse fonctionnelle, on peut considérer que, dans un certain sens, être $\alpha$-höldérien (pour $\alpha \in ]0,1[$) exprime l'idée d'être "$\alpha$ fois dérivable", avec $\alpha$ fractionnaire. On peut donc retenir que le mouvement brownien est "presque" « 1/2 fois dérivable », au sens où il est « $(1/2-\varepsilon)$ fois dérivable » pour tout $\varepsilon > 0$, mais que par contre, il n'est jamais plus dérivable que cela.* [*] ♢

## 8.2 Équations différentielles stochastiques

### Le bruit blanc gaussien

☛ *Ce paragraphe est présenté dans un but purement culturel ; la suite de la section doit être lue sans en tenir compte.*

Le mouvement brownien n'est pas seulement la star de tous les processus aléatoires, c'est aussi un objet qui est au cœur de toute la théorie des équations différentielles stochastiques, à laquelle nous allons maintenant donner une introduction. Plus exactement, ce n'est pas le mouvement brownien lui-même qui est l'objet fondamental, mais la *dérivée* du mouvement brownien, qu'on appelle *bruit blanc gaussien.*

« *Quelle* dérivée ?! » protesterez-vous. « Le théorème (LT) ci-dessus a dit que le mouvement brownien n'est pas dérivable ! ». Certes, pas au sens classique. Mais il est toutefois dérivable *au sens des distributions* [†]. Le bruit blanc gaussien est donc une *distribution aléatoire* sur $\mathbb{R}$. Mais comme ce ne serait pas un objet pratique à manipuler, on préfère en général tout écrire en termes de son intégrale, à savoir le mouvement brownien, qui lui est une "brave" fonction continue.

Nous donnons quand même la définition et les propriétés fondamentales du bruit blanc gaussien, pour la culture mathématique. Notez que le bruit blanc gaussien se définit en général sur tout $\mathbb{R}$ et pas seulement sur $\mathbb{R}_+$.

**Définition (LU).** On appelle *bruit blanc gaussien* (unidimensionnel), d'intensité par unité de distance égale à $\sigma^2$, une distribution aléatoire $W$ sur $\mathbb{R}$ (à valeurs réelles) telle que, pour toute fonction-test $\varphi \in \mathscr{D}(\mathbb{R})$, on ait $\langle T, \varphi\rangle \sim \text{Normale}\big(\sigma^2 \int_{\mathbb{R}} \varphi(x)^2 \mathrm{d}x\big)$. Nous admettrons que, sous réserve que donner une bonne définition formelle pour ce qu'est une distribution aléatoire, cela définit bien la loi de $T$ de façon unique. ♡

**Théorème (LV).** *Si $T$ est un bruit blanc gaussien,*

1. *Les valeurs de $T$ sur des ouverts disjoints sont indépendantes, c.-à-d. que si $\varphi_1, \dots, \varphi_k$ sont des fonctions-test à supports disjoints, $\langle T, \varphi_1\rangle, \dots, \langle T, \varphi_k\rangle$ sont indépendants.*
2. *$T$ est invariante par translation, c.-à-d. que pour tout $a \in \mathbb{R}$, $\delta_a * T$ a la même loi que $T$.*
3. *$T$ est invariante d'échelle de facteur $-1/2$, au sens où pour tout $\lambda \in \mathbb{R}_+^*$, quand on accélère le temps par un facteur $\lambda$, i. e. qu'on considère l'image de $T$ par l'application $f(\bullet) \mapsto f(\lambda \times \bullet)$ [‡], on obtient la même loi que lorsqu'on multiplie le mouvement brownien par un facteur $\lambda^{-1/2}$ (i. e., qu'on considère son image par l'application $f(\bullet) \mapsto \lambda^{1/2} \times f(\bullet)$ [§]).*

♢

*Démonstration.* Il suffit de revenir à la définition. ♤

[*]. Le cas critique de l'exposant 1/2 est un peu plus subtil : presque-surement, il existe *quelques* valeurs de $t$ en lesquelles la trajectoire de mouvement brownien est höldérienne ; néanmoins elle n'est höldérienne sur aucun intervalle d'intérieur non vide.

[†]. Confer le cours d'introduction au parcours IM de M^me^ Lucardesi et M. Henrot ☺

[‡]. Cette application est une application qui transforme une fonction en une autre fonction. En l'occurrence, on peut montrer qu'elle se prolonge naturelle à une application qui transforme une distribution en une autre distribution : c'est l'image du bruit blanc gaussien par cette application-là dont nous sommes en train de parler en réalité.

[§]. Même remaruqe que ci-dessus : cette application, qui transforme une fonction en fonction, doit être comprise au sens de son extension naturelle, qui transforme une distribution en distribution.

*Remarque* (LW). Contrairement au mouvement brownien, on peut aussi définir le bruit blanc sur un espace de dimension supérieure à 1 [¶] ! Le définition est la suivante, et les propriétés du théorème (LV) se trasnposent directement :

*Définition* (LX). On appelle *bruit blanc gaussien $d$-dimensionnel*, d'intensité par unité de *volume* égale à $\sigma^2$, une distribution aléatoire $W$ sur $\mathbb{R}^d$ (à valeurs réelles) telle que, pour toute fonction-test $\varphi \in \mathscr{D}(\mathbb{R}^d)$, on ait $\langle T, \varphi\rangle \sim \text{Normale}\big(\sigma^2 \int_{\mathbb{R}} \varphi(x)^2 \,\text{vol}(\mathrm{d}x)\big)$. ♡

♧

## Principe général des équations différentielles stochastiques

! **Définition (LY).** Une équation différentielle stochastique (markovienne) à valeurs dans $\mathbb{R}^n$ est une équation de la forme

$$\mathrm{d}X_t = \sigma(X_t)\mathrm{d}B_t + b(X_t)\mathrm{d}t, \tag{LZ}$$

où "$\mathrm{d}B_t$" note l'incrément d'un mouvement brownien standard de $\mathbb{R}^d$ (on aura généralement $d = n$, car on peut toujours se ramener à ce cas), et $\sigma : \mathbb{R}^n \to \mathbb{R}^{n\times d}$ et $b : \mathbb{R}^n \to \mathbb{R}^n$ sont certaines fonction déterministes (continues). ♡

! *Remarque* (MA). Nous ne chercherons pas ici à savoir sous quelles conditions une telle équation a un sens. On simulera les solutions de cette équation par la méthode d'Euler stochastique, consistant à simuler successivement $X_0, X_\varepsilon, X_{2\varepsilon}, \dots$ en assimilant « $\mathrm{d}X_{n\varepsilon}$ » à « $X_{n\varepsilon+\varepsilon} - X_{n\varepsilon}$ ». ♧

! *Remarque* (MB). Quand nous simulions le mouvement brownien, il y avait une erreur due à la discrétisation de temps (l'interpolation linéaire entre deux points de discrétisation ne correspondant pas exactement au vrai mouvement brownien), mais en revanche la simulation aux points de discrétisation était *exacte*. Quand nous simulons une équation différentielle stochastique par la méthode d'Euler stochastique, en revanche, *même les points de discrétisation ne sont pas simulés exactement*, car le schéma d'Euler n'est valable en toute rigueur qu'*asymptotiquement*, quand $\varepsilon$ tend vers 0.

Il faut donc *aussi* tenir compte de cette *erreur de simulation* quand nous appliquons la méthode de Monte-Carlo à des processus aléatoires. Comme dans le cas du mouvement brownien, on dispose de bornes permettant de contrôler cette erreur de simulation, mais celles-ci ne sont pas simples. *En pratique*, comme pour le mouvement brownien, on supposera que l'erreur de simulation est bien plus faible que l'erreur de Monte-Carlo, et on fera donc *comme si* la simulation était exacte ; mais il faudra garder à l'esprit que cette hypothèse n'est pas toujours vérifiée et peut donc conduire à des erreurs dans certaines situations. ♧

! *Remarque* (MC). La notation « $\mathrm{d}B_t$ » dans l'équation différentielle stochastique laisse à penser qu'on a besoin de simuler le mouvement brownien $(B_t)_{t\geqslant 0}$ pour appliquer le schéma d'Euler. C'est certes une possibilité, mais ce serait inutilement lourd ; car l'objet $\mathrm{d}B_t$ (qui désigne les accroissements du mouvement brownien) est en fait *plus simple* que le mouvement brownien lui-même : par définition du mouvement brownien en effet, les $\mathrm{d}B_t$ sont indépendants et de loi Normale($\varepsilon$) (où $\varepsilon$ est la largeur

[¶]. Le mouvement brownien « multidimensionnel » que nous avons rencontré précédemment était un processus à *valeurs* dans $\mathbb{R}^d$. Cette généralisation serait aussi possible pour le bruit blanc gaussien, mais pas très intéressante en l'occurrence. Ici ce que nous avons, c'est un processus *indexé* par $\mathbb{R}^d$ : et cela n'a en revanche pas d'équivalent (ou du moins pas d'équivalent canonique) dans le cadre "mouvement brownien".

du pas de temps concerné). Simuler $(B_t)_{t\geqslant 0}$ par la méthode d'Euler stochastique avant de prendre ses accroissements reviendrait à simuler les $\mathrm{d}B_t$, à les sommer pour avoir le mouvement brownien, puis à prendre les différences pour... récupérer les incréments qu'on avait déjà simulés, ce qui est clairement idiot. (Au sujet de l'importance intrinsèque de $(\mathrm{d}B_t)_{t\geqslant 0}$, voir aussi la § 8.2 sur le bruit blanc gaussien). ♧

*Exemple* (MD). Des exemples d'implémentation de simulation de processus stochastiques sont donnés par les progammes `browngeom.py` et `OU.py` que vous trouverez sur *Arche* ; ils simulent respectivement un mouvement brownien géométrique et un processus d'Ornstein-Uhlenbeck, dont les définitions seront données dans la § 8.2. Les rendus de ces programmes correspondent respectivement aux figures 8.3 et 8.4. ♧

## Changement de variables

**Théorème (ME)** (Formule d'Itô en dimension 1)**.** *Soit un processus $(X_t)_t$ à valeurs réelles suivant l'équation différentielle stochastique* !

$$\mathrm{d}X_t = \sigma(X_t)\mathrm{d}B_t + b(X_t)\mathrm{d}t, \tag{MF}$$

*pour des applications $\sigma\colon \mathbb{R}\to\mathbb{R}$ et $b\colon \mathbb{R}\to\mathbb{R}$ de classe $\mathrm{C}^1$ ; soit $f\colon \mathbb{R}\to\mathbb{R}$ une appliation de régularité $\mathrm{C}^2$, et posons $Y_t := f(X_t)$. Alors le processus $(Y_t)_t$ est solution de l'équation différentielle stochastique suivante :*

$$\mathrm{d}Y_t = f'(X_t)\sigma(X_t)\mathrm{d}B_t + (f'(X_t)b(X_t) + \tfrac{1}{2}f''(X_t)\sigma(X_t)^2)\mathrm{d}t. \tag{MG}$$

♢

*Remarque* (MH). Les accroissements du mouvement brownien $(B_t)_t$ qui intervient dans l'équation différentielle régissant l'évolution de $Y$ sont les mêmes que ceux de l'évolution de $X$. ♧

Nous allons donner une "pseudo-preuve" de la formule d'Itô (qui pourrait être rendue rigoureuse au prix de beaucoup de détails techniques), qui nous permettra de comprendre d'où vient cette formule, et ainsi de la mémoriser beaucoup plus facilement.

*Démonstration.* Pour simplifier les calculs, nous allons supposer que les fonctions $\sigma$ et $b$ sont uniformément bornées, de même que les dérivées seconde et troisième de $f$. Rappelons que la seule "définition" que nous ayons de la solution d'une équation différentielle stochastique consiste à discrétiser l'équation pour un pas de temps infinitésimal : autrement dit, on sait que la solution de l'équation est "presque" celle qu'on obtient en prenant, pour $h$ une très petite valeur :

$$X_{t+h} = X_t + \sigma(X_t)\delta B_t + b(X_t)h, \tag{MI}$$

où $\delta B_t$ désigne la différence $B_{t+h} - B_t$. Nous noterons de même $\delta X_t := X_{t+h} - X_t$, $\delta Y_t := Y_{t+h} - Y_t$. Nous appliquons maintenant la formule de Taylor à l'ordre 2 à la différence $Y_{t+h} - Y_t$ :

$$Y_{t+h} - Y_t = f(X_{t+h}) - f(X_t) = f(X_t) + f'(X_t)\delta X_t + \tfrac{1}{2}f''(X_t)(\delta X_t)^2 + R_t^{(0)}, \tag{MJ}$$

avec $|R_t^{(0)}| \leqslant \frac{1}{6}\|f'''\|_\infty |\delta X_t|^3$. Observons qu'on a déjà $|\delta X_t| \leqslant \|\sigma\|_\infty|\delta B_t| + \|b\|_\infty h \leqslant 2(\|\sigma\|_\infty|\delta B_t| \vee \|b\|_\infty h)$, d'où $|\delta X_t|^3 \leqslant 8(\|\sigma\|_\infty^3|\delta B_t|^3 \vee \|b\|_\infty^3 h^3) \leqslant 8\|\sigma\|_\infty^3|\delta B_t|^3 + 8\|b\|_\infty^3 h^3$. Connaissant les moments de la valeur absolue d'une loi normale standard (en particulier, le troisième moment vaut $4/\sqrt{2\pi} \leqslant \frac{13}{8}$), on en déduit que

$$\mathbb{E}(|R|_t^{(0)}) \leqslant 13\|\sigma\|_\infty^3 h^{3/2} + 8\|b\|_\infty^3 h^3. \tag{MK}$$

Noter que cette égalité reste valable *conditionnellement à tout ce qui s'est passé jusqu'à l'instant t*.

Par ailleurs, en développant $\delta X_t = \sigma(X_t)\delta B_t + b(X_t)h$, on a

$$f'(X_t)\delta X_t + \tfrac{1}{2}f''(X_t)(\delta X_t)^2 = f'(X_t)\sigma(X_t)\mathrm{d}B_t + f'(X_t)b(X_t)h + \tfrac{1}{2}f''(X_t)\sigma(X_t)^2(\delta B_t)^2 + f''(X_t)\sigma(X_t)h\delta B_t + \tfrac{1}{2}f''(X_t)h^2. \quad (\mathtt{ML})$$

Notons $R_t^{(1)} \coloneqq \frac{1}{2}f''(X_t)h^2$ et $S_t^{(0)} \coloneqq f''(X_t)\sigma(X_t)h\delta B_t$. On a

$$\mathbb{E}(|R_t^{(1)}|) \leqslant \tfrac{1}{2}\|f''\|_\infty h^2\,; \quad (\mathtt{MM})$$

quant à $S_t^{(0)}$, il s'agit d'une variable aléatoire *centrée*, avec

$$\mathrm{Var}(S_t^{(0)}) \leqslant \|f''\|_\infty^2\|\sigma\|_\infty^2 h^3. \quad (\mathtt{MN})$$

À nouveau, tout ce que nous écrivons reste vrai conditionnellement à la connaissance du passé jusqu'à l'instant $t$.

Maintenant, posons $S_t^{(1)} \coloneqq \frac{1}{2}f''(X_t)\sigma(X_t)^2((\delta B_t)^2 - h)$, de sorte qu'on ait

$$\tfrac{1}{2}f''(X_t)\sigma(X_t)^2(\delta B_t)^2 = \tfrac{1}{2}f''(X_t)\sigma(X_t)^2 h + S_t^{(1)}. \quad (\mathtt{MO})$$

Par construction, la variable aléatoire $S_t^{(1)}$ est centrée, avec

$$\mathrm{Var}(S_t^{(1)}) = \mathbb{E}((S_t^{(1)})^2) \leqslant \tfrac{1}{4}\|f''\|_\infty^2\|\sigma\|_\infty^4\,\mathbb{E}\big(((\delta B_t)^2 - h)^2\big). \quad (\mathtt{MP})$$

Et vu que le moment quatrième de la loi normale standard vaut 3, on a en développant

$$\mathbb{E}\big(((\delta B_t)^2 - h)^2\big) = \mathbb{E}((\delta B_t)^4) + h^2 - 2\,\mathbb{E}((\delta B_t)^2)h = 2h^2. \quad (\mathtt{MQ})$$

Finalement, nous arrivons à écrire que, au niveau discrétisé, $Y_t$ satisfait effectivement "presque" l'équation différentielle stochastique annoncée, à deux termes "perturbatifs" près :

$$\delta Y_t = f'(X_t)\sigma(X_t)\delta B_t + (f'(X_t)b(X_t) + \tfrac{1}{2}f''(X_t)\sigma(X_t)^2)\delta t + S_t + R_t, \quad (\mathtt{MR})$$

avec $S_t \coloneqq S_t^{(0)} + S_t^{(1)}$, resp. $R_t \coloneqq R_t^{(0)} + R_t^{(1)}$.

La question qui se pose est maintenant de vérifier que les effets respectifs des termes $S_t$ et $R_t$ sont effectivement négligeables. Ici nous allons admettre que ce qui compte, c'est que la *somme* des $S_t$, resp. des $R_t$, soit négligeable. (En réalité, le décalage au temps $t+h$ dû aux termes perturbatifs crée à son tour un décalage sur la valeur de $\sigma(X_{t+h})$ et $b(X_{t+h})$, qui se répercute ensuite sur l'équation... Cependant nous admettrons que le fait d'avoir pris les fonctions $\sigma(\bullet)$ et $b(\bullet)$ de classe $\mathrm{C}^1$ suffit à ce que l'effet cumulé de ces décalages ne soit pas (significativement) plus grand que les effets qu'on obtenait par sommation directe des termes perturbatifs eux-mêmes).

Regardons d'abord l'effet de $R_t$. Nous avons

$$\mathbb{E}(|R_t|) \leqslant \mathbb{E}(|R_t^{(0)}|) + \mathbb{E}(|R_t^{(1)}|) \leqslant 13\|\sigma\|_\infty^3 h^{3/2} + \tfrac{1}{2}\|f''\|_\infty h^2 + 8\|b\|_\infty^3 h^3\,; \quad (\mathtt{MS})$$

Dans cette expression, lorsque $h \to 0$, c'est le premier terme qui est dominant : pour peu que $h$ ait été choisi suffisamment petit nous avons donc

$$\mathbb{E}(|R_t|) \leqslant 14\|\sigma\|_\infty^3 h^{3/2}. \quad (\mathtt{MT})$$

Si nous sommons tous les termes $R_t$ entre 0 et $T$, il y a $T/h$ tels termes : on a donc

$$\mathbb{E}\Big(\Big|\sum_{t=0}^{T} R_t\Big|\Big) \leqslant (T/h) \times 14\|\sigma\|_\infty^3 h^{3/2} = 14T\|\sigma\|_\infty^3 h^{1/2} \overset{h\to 0}{\to} 0\,: \quad (\mathtt{MU})$$

ainsi l'effet dû à la perturbation en $R_t$ disparait lorsqu'on fait tendre $h$ vers 0 (ce qui est le cadre dans lequel l'équation différentielle stochastique est définie).

Pour la constribution des $S_t$, c'est un peu plus subtil : cette fois-ci nous allons devoir utiliser que les $S_t$ sont centrés. Observons déjà qu'on a, d'après l'inégalité triangulaire pour la norme $\mathrm{L}^2$,

$$\begin{aligned}\mathbb{E}(S_t^2) &\leqslant (\mathbb{E}((S_t^{(0)})^2)^{1/2} + \mathbb{E}((S_t^{(1)})^2)^{1/2})^2 \leqslant 2(\mathbb{E}((S_t^{(0)})^2)^{1/2} \vee \mathbb{E}((S_t^{(1)})^2)^{1/2})^2 \\ &= 2\,\mathbb{E}((S_t^{(0)})^2) \vee 2\,\mathbb{E}((S_t^{(1)})^2) \leqslant 2\,\mathbb{E}((S_t^{(0)})^2) + 2\,\mathbb{E}((S_t^{(1)})^2) \\ &= \|f''\|_\infty\|\sigma\|_\infty^4 h^2 + 2\|f''\|_\infty^2\|\sigma\|_\infty^2 h^3. \quad (\mathtt{MV})\end{aligned}$$

En outre, pour $t' > t$, $S_{t'}$ est centré conditionnellement à tout ce que se passe avant le temps $t'$, donc en particulier conditionnellement à $S_t$, de sorte que $\mathbb{E}(S_t S_{t'}) = \mathbb{E}\big(S_t\,\mathbb{E}(S_{t'} \mid \mathscr{F}_{t'})\big) = 0$ (où j'ai noté par $\mathscr{F}_{t'}$ la tribu du passé jusqu'à l'instant $t'$). Dès lors,

$$\begin{aligned}\mathbb{E}\Big(\Big(\sum_{t=0}^{T} S_t\Big)^2\Big) = \sum_{t=0}^{T}\sum_{t'=0}^{T}\mathbb{E}(S_t S_{t'}) = \sum_{t=0}^{T}\mathbb{E}(S_t^2)\\ \leqslant (T/h)\times(\|f''\|_\infty\|\sigma\|_\infty^4 h^2 + 2\|f''\|_\infty^2\|\sigma\|_\infty^2 h^3)\\ = T(\|f''\|_\infty\|\sigma\|_\infty^4 + 2\|f''\|_\infty^2\|\sigma\|_\infty^2 h)h, \quad \text{(MW)}\end{aligned}$$

qui tend vers 0 lorsque $h \to 0$ : c'est donc que l'effet des termes en $S_t$ est négligeable également dans la limite continue. Finalement nous avons bien démontré que, à la limite continue, $(Y_t)_t$ suit l'équation différentielle stochastique annoncée ! ♤

Cette "démonstration" nous montre le raisonnement à suivre pour comprendre la formule d'Itô :

**Principe (MX)** (Application pratique de la formule d'Itô)**.** *Pour appliquer la formule d'Itô dans les calculs, on peut procéder ainsi :* !

1. *Appliquer la formule de Taylor* au deuxième ordre *(on tient compte de la dérivée seconde de $f$).*
2. *Considérer que les termes de* $\mathrm{d}\boldsymbol{B}_t$ *sont des termes d'ordre* $\mathrm{O}(\mathrm{d}t^{1/2})$.
3. *Négliger les termes d'ordre* $\mathrm{o}(\mathrm{d}t)$*, ce qui inclut notamment les termes en* $\mathrm{O}(\mathrm{d}t^{3/2})$.
4. *Remplacer les termes d'ordre* $\mathrm{d}t$ *qui sont du type « produits d'accroissements infinitésimaux de mouvements brownien » par leur espérance, qui sera alors un terme de dérive (i. e. un terme de type* $\mathrm{d}t$*).*

♢

Nous allons illustrer, de manière très générale, la façon dont ce raisonnement fonctionne pour établir la formule d'Itô dans le cadre multidimensionnel : formule d'apparence assez effrayante, mais qui se ramène en réalité au simple principe énoncé ci-dessus !

Considérons donc, cette fois-ci, que notre processus $(\vec{X}_t)_t$ est à valeurs dans $\mathbb{R}^n$, dont les fluctuations sont régies par un mouvement brownien $d$-dimensionnel ($\boldsymbol{\sigma}$ est donc une matrice de $\mathbb{R}^{n\times d}$), et que $f$ est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^m$. On a alors, par construction, en notant $\boldsymbol{X}_j$ la $j$-ième cordonnée de $\vec{\boldsymbol{X}}$, $\boldsymbol{B}_i$ la $i$-ième coordonnée du mouvement brownien, &c. [‖] :

$$\mathrm{d}X_j = \sum_{i=1}^{d}\sigma_{ji}(\vec{X}_t)\mathrm{d}\boldsymbol{B}_i + b_j(\vec{X}_t). \quad \text{(MY)}$$

Le développement au second ordre de l'accroissement de $\vec{\boldsymbol{Y}}$ nous donne par ailleurs que

$$\mathrm{d}Y_k = \sum_{j=1}^{n}\frac{\partial f_k}{\partial x_j}(\vec{X}_t)\mathrm{d}X_j + \tfrac{1}{2}\sum_{j=1}^{n}\sum_{j'=1}^{n}\frac{\partial^2 f_k}{\partial x_j \partial x_{j'}}\mathrm{d}X_j\mathrm{d}X_{j'}, \quad \text{(MZ)}$$

[‖]. Attention ! Ci-dessous, pour alléger les notations, lorsqu'on parle d'accroissements infinitésimaux, il est sous-entendu qu'il s'agit d'accroissements entre les temps $t$ et $t + \mathrm{d}t$ : une notation comme « $\mathrm{d}\boldsymbol{X}_j$ », par exemple, devrait être lue comme « $\mathrm{d}(\boldsymbol{X}_j)_t$ » : car ce n'est évidemment pas l'indice $j$ qui augmente infinitésimalement, mais bien le temps... !

d'où après développement :

$$\begin{aligned}
\mathrm{d}Y_k = &\sum_{i=1}^{d}\sum_{j=1}^{n}\frac{\partial f_k}{\partial x_j}(\vec{X}_t)\sigma_{ji}(\vec{X}_t)\mathrm{d}\boldsymbol{B}_i \\
&+\sum_{j=1}^{n}\frac{\partial f_k}{\partial x_j}(\vec{X}_t)b_j(\vec{X}_t)\mathrm{d}t \\
&+\tfrac{1}{2}\sum_{i=1}^{d}\sum_{i'=1}^{d}\sum_{j=1}^{n}\sum_{j'=1}^{n}\frac{\partial^2 f_k}{\partial x_j\partial x_{j'}}\sigma_{ji}(\vec{X}_t)\sigma_{j'i'}(\vec{X}_t)\mathrm{d}\boldsymbol{B}_i\mathrm{d}\boldsymbol{B}_{i'} \\
&+\sum_{i=1}^{d}\sum_{j=1}^{n}\sum_{j'=1}^{n}\frac{\partial^2 f_k}{\partial x_j\partial x_{j'}}\sigma_{ji}(\vec{X}_t)b_{j'}(\vec{X}_t)\mathrm{d}\boldsymbol{B}_i\mathrm{d}t \\
&+\tfrac{1}{2}\sum_{j=1}^{n}\sum_{j'=1}^{n}\frac{\partial^2 f_k}{\partial x_j\partial x_{j'}}b_j(\vec{X}_t)b_{j'}(\vec{X}_t)\mathrm{d}t^2.
\end{aligned} \tag{NA}$$

Les deux premiers termes de la formume ci-dessous sont des termes classiques d'équation différentielle stochastique ; nous les laissons donc tels quels. Le quatrième terme est d'ordre $\mathrm{O}(\mathrm{d}t^{3/2})$ (puisque nous avons dit que $\mathrm{d}\boldsymbol{B}_i$ devait être traité comme un $\mathrm{O}(\mathrm{d}t^{1/2})$), et le cinquième terme d'ordre $\mathrm{O}(\mathrm{d}t^2)$ : nous pouvons donc les négliger puisque ce sont des $\mathrm{o}(\mathrm{d}t)$. Reste le troisième terme, qui fait intervenir un produit $\mathrm{d}\boldsymbol{B}_i\mathrm{d}\boldsymbol{B}_{i'}$ : nous devons alors chercher l'espérance de ce terme. Comme les différents $\mathrm{d}\boldsymbol{B}_i$ sont indépendants (et centrés), on a pour $i \neq i'$ que $\mathbb{E}(\mathrm{d}\boldsymbol{B}_i\mathrm{d}\boldsymbol{B}_{i'}) = 0$, de sorte qu'on n'a qu'à considérer le cas où $i = i'$. On a alors, par définition du mouvement brownien standard, $\mathbb{E}(\mathrm{d}\boldsymbol{B}_i{}^2) = \mathrm{d}t$, ce qui nous donne finalement la formule d'Itô multidimensionnelle :

**Théorème (NB)** (Formule d'Itô multidimensionnelle)**.** *Soit un processus $\vec{X}$ en dimension $n$ suivant l'équation différentielle stochastique « $\mathrm{d}\vec{X} = \boldsymbol{\sigma}(\vec{X}_t)\mathrm{d}\vec{B} + \vec{b}(\vec{X}_t)\mathrm{d}t$ », pour $(\vec{B}_t)_t$ un mouvement brownien $d$-dimensionnel, et $\boldsymbol{\sigma}\colon \mathbb{R}^n \to \mathbb{R}^{n\times d}$ et $\vec{b}\colon \mathbb{R}^n \to \mathbb{R}^n$ suffisamment régulières (de classe $\mathrm{C}^1$). Si $\vec{f}\colon \mathbb{R}^n \to \mathbb{R}^m$ est de classe $\mathrm{C}^2$, alors le processus aléatoire $m$-dimensionnel $\vec{Y} \coloneqq \vec{f}(\vec{X})$ suit l'équation différentielle stochastique suivante :*

$$\begin{aligned}
\mathrm{d}Y_k = &\sum_{i=1}^{d}\sum_{j=1}^{n}\frac{\partial f_k}{\partial x_j}(\vec{X}_t)\sigma_{ji}(\vec{X}_t)\mathrm{d}\boldsymbol{B}_i \\
&+\sum_{j=1}^{n}\frac{\partial f_k}{\partial x_j}(\vec{X}_t)b_j(\vec{X}_t)\mathrm{d}t \\
&+\tfrac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{n}\sum_{j'=1}^{n}\frac{\partial^2 f_k}{\partial x_j\partial x_{j'}}\sigma_{ji}(\vec{X}_t)\sigma_{j'i}(\vec{X}_t)\mathrm{d}t.
\end{aligned} \tag{NC}$$

◇

*Remarque* **(ND)**. Le dernier terme de la formule (NC), qui n'aurait pas été prévu par un développement de Taylor au premier ordre, est appelé « terme d'Itô ». Il existe une autre façon de définir la théorie des équations différentielles stochastiques, appelée *formalisme de Stratonovitch*[**] : dans cette variante (moins répandue en pratique), les changements de variables sont beaucoup plus intuitifs ; par contre,

[**]. En anglais, le nom russe Стратоно́вич est rendu par « Stratonovich » (sans 't' avant le "ch" final).

on perd les propriétés de martingale et la facilité de simulation... On n'a rien sans rien ! ♣

*Remarque* (NE). Les formules sommatoires fournies par la formule d'Itô multidimensionnelle peuvent en réalité être écrites de façon beaucoup plus dans un cadre d'algèbre multilinéaire, en utilisant des tenseurs appropriés : notant $\mathrm{D}f$ et $\mathrm{D}^2 f$ les tenseurs des dérivées première et seconde de $f$, on a

$$\mathrm{d}\vec{Y} = \mathrm{D}f \cdot \boldsymbol{\sigma}\mathrm{d}\vec{B} + (\mathrm{D}f \cdot \vec{b} + \tfrac{1}{2}\mathrm{D}^2 f \cdot \boldsymbol{\sigma}\boldsymbol{\sigma}^{\mathsf{T}})\mathrm{d}t : \qquad (\mathtt{NF})$$

même sans comprendre exactement ce que signifie cette formule, on voit bien qu'il s'agit d'une adaptation tensorielle immédiate de la formule unidimensionnelle (MG) ! ♣

## Exemples

Les trois exemples de processus stochastiques ci-dessous sont des standards sur lesquels on s'appuie en modélisation, et doivent donc être connus :

*Exemple* (NG) (Mouvement brownien avec dérive). Si $\sigma(X_t)$ et $b(X_t)$ sont constants, la solution de l'équation différentielle stochastique (qui est alors évidemment égale à $X_0 + B_t + bt$) s'appelle un *mouvement brownien avec dérive*, de variance par unité de temps $\sigma\sigma^{\mathsf{T}}$ et de vitesse de dérive $b$. ♣ !

*Exemple* (NH). On appelle *mouvement brownien géométrique* de paramètres $\sigma^2$ et $b$ la solution de l'équation différentielle stochastique : !

$$\mathrm{d}X_t = \sigma X_t \mathrm{d}B_t + (b + \sigma^2/2)X_t \mathrm{d}t. \qquad (\mathtt{NI})$$

D'après la formule d'Itô, si $(B_t)_{t\geqslant 0}$ est un mouvement brownien unidimensionnel avec dérive de paramètres $\sigma^2$ et $\mu$, alors $(\mathrm{e}^{B_t})_{t\geqslant 0}$ est un mouvement brownien géométrique de mêmes paramètres. ♣

*Exemple* (NJ). Pour $\lambda > 0$, la solution de l'équation différentielle stochastique (en dimension 1) !

$$\mathrm{d}X_t = \sigma \mathrm{d}B_t - \lambda X_t \mathrm{d}t \qquad (\mathtt{NK})$$

est appelée *processus d'Ornstein-Uhlenbeck* de paramètres $\sigma$ (paramètre de bruit) et $\lambda$ (paramètre de relaxation). Si on part d'une condition initiale $X_0$ distribuée selon la loi Normale($\sigma^2$ / $2\lambda$) (étant entendu que le bruit blanc gaussien est indépendant de cette condition initiale), alors on peut montrer ce processus est « stationnaire », c.-à-d. que pour tout $t_1 \geqslant 0$, $(X_{t_1+t})_{t\geqslant 0}$ a la même loi que $(X_t)_{t\geqslant 0}$. On dit que la loi Normale($\sigma^2$ / $2\lambda$) est une *mesure d'équilibre* du processus. ♣

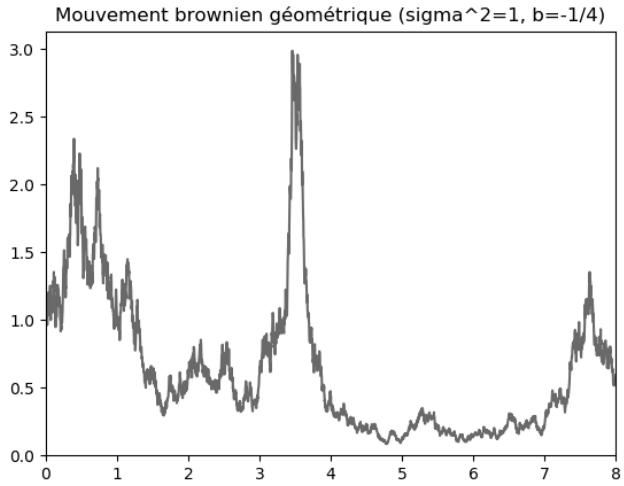


FIGURE 8.3 – Mouvement brownien géométrique. Remarquez qu'on voit bien comme le bruit est proportionnel à la valeur du processus.

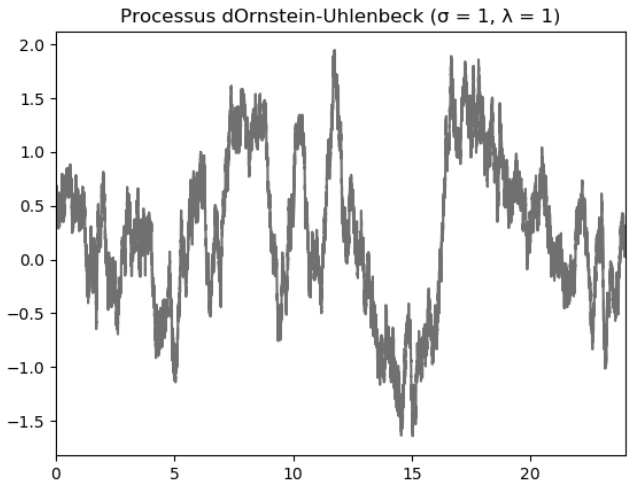


FIGURE 8.4 – Processus d'Ornstein-Uhlenbeck (initialisé depuis sa mesure stationnaire). On devine sur ce dessin que, malgré l'agitation extrêmement violente du processus, une "force" l'empêche de trop s'éloigner de zéro...

# Annexe A

# Annexe : Détails techniques

## A.1 Le championnat de basket

### Précisions sur le contexte

Le championnat national de basketball met en lice $16 =: Z$ équipes. Chacune de ces équipes rencontre chaque autre une fois et une seule. À la fin du championnat, l'équipe qui a remporté le plus de victoires [*] gagne le championnat. Dans l'hypothèse où plusieurs équipes sont premières ex-æquo au nombre de victoires, la gagnante est tirée au sort uniformément parmi ces premières ex-æquo.

La modélisation de notre passionné consiste à attribuer à chaque équipe un numéro de 1 (la plus forte sur le papier) à 16 (la moins forte). L'équipe de Nancy est associée au nombre $3 =: Nancy$. Lorsque l'équipe $i$ rencontre l'équipe $j$, la probabilité que ce soit $i$ qui gagne vaut alors $i^{-1/2} / (i^{-1/2} + j^{-1/2}) =: p_{vict}(i,j)$ [†]. En outre, comme nous l'avons déjà dit, les résultats des différents matchs sont indépendants.

### Calcul de la densité $\mathrm{d}P/\mathrm{d}Q$

L'univers que nous considérons pour ce calcul de densité est $\Omega := \{0,1\}^E$, où $E := \{(i,j) \in \{0,\dots,15\}^2 \mid i < j\}$. Pour $\omega \in \Omega$, les résultats du championnat sous cette éventualité sont décrits en disant que pour $(i,j) \in E$, $\omega_{(i,j)}$ vaut 1 si l'équipe n° $i$ a battu l'équipe n° $j$ lors de leur confrontation, et 0 si c'est n° $j$ qui a battu n° $i$. Sous la loi $P$ comme sous la loi $Q$, les $(\omega_e)_{e\in E}$ sont indépendants, de sorte que

$$\frac{\mathrm{d}P}{\mathrm{d}Q}((\omega_e)_{e\in E} = (r_e)_{e\in E}) = \prod_{e\in E} \frac{\mathrm{d}(\omega_e * P)}{\mathrm{d}(\omega_e * Q)}(b_e) = \prod_{e\in E} \frac{\mathrm{d}P(\omega_e = b_e)}{\mathrm{d}Q(\omega_e = b_e)}. \tag{NL}$$

Par définition de $Q$, on a :

$$\frac{P(\omega_{(i,j)} = 1)}{Q(\omega_{(i,j)} = 1)} = \begin{cases} 1 & \text{si } i,j \neq 13; \\ (1 + (j+1)^{-1/2}) / (1 + \sqrt{14}(j+1)^{-1/2}) & \text{si } i = 13 \text{ [‡]}; \\ ((i+1)^{-1/2} + 1) / ((i+1)^{-1/2} + 1/\sqrt{14}) & \text{si } j = 13; \end{cases} \tag{NM}$$

[*]. Il n'y a pas de match nul au basketball.

[†]. Cette modélisation est bien cohérente, dans la mesure où $p(i,j) + p(j,i) = 1$.

et

$$\frac{P(\omega_{(i,j)}=0)}{Q(\omega_{(i,j)}=0)} = \begin{cases} 1 & \text{si } i,j \neq 13; \\ (1+j^{-1/2}) \,/\, (1/\sqrt{14}+j^{-1/2}) & \text{si } i=13; \\ (i^{-1/2}+1) \,/\, (\sqrt{14}i^{-1/2}+1) & \text{si } j=13; \end{cases} \tag{NO}$$

Au final,

$$\frac{\mathrm{d}P}{\mathrm{d}Q}((\omega_e)_{e\in E}=(r_e)_{e\in E}) = \prod_{i<13} \frac{P(\omega_{(i,13)}=r_{(i,13)})}{Q(\omega_{(i,13)}=r_{(i,13)})} \prod_{j>13} \frac{P(\omega_{(13,j)}=r_{(13,j)})}{Q(\omega_{(13,j)}=r_{(13,j)})}. \tag{NP}$$

*Remarque* (NQ). Comme la densité $\mathrm{d}P/\mathrm{d}Q$ apparait comme un produit de fonctions des $\omega_e$, et que les $\omega_e$ sont simulés indépendamment, on va pouvoir calculer la densité en même temps qu'on simule les $\omega_e$, en maintenant un produit à jour. ♧

## A.2 Le Keno

### Précisions sur le contexte

Le jeu de Keno se joue sur une grille comportant les numéros de 1 à 80. Le joueur doit choisir 20 de ces numéros, dont un auquel il attribue la valeur 20, un auquel il attribue la valeur 19, ..., et un auquel il attribue la valeur 1. Lors du tirage, la machine tire au hasard 20 boules parmi un jeu de boules numérotées de 1 à 80. Le *score* d'un joueur est alors défini comme la somme de valeurs qu'il a associées aux numéros qu'il a cochés et qui ont effectivement été tirés. La gain est ensuite défini de la façon suivante. En-dessous d'un score de 75, le joueur ne gagne rien. Pour un score de 75, le joueur regagne juste les 10 € qu'il avait misés. Puis, à chaque point de score supplémentaire, le gain (brut) du joueur augmente de 10 %, en arrondissant à chaque fois au nombre entiers d'euros le plus proche (par-dessus si c'est un demi-entier). Ainsi un score de 76 points rapporte 11 €, un score de 77 points rapporte 12 €, ..., un score de 80 points rapporte 15 €, un score de 81 points rapporte 17 €, ..., et un score parfait de 210 points rapporte 3 862 110 € !

### Calcul de l'espérance conditionnellement à $\mathscr{F}_{19}$

Dans cette sous-section nous explicitons la valeur de l'espérance conditionnelle du gain sous la tribu $\mathscr{F}_{19}$, c'est-à-dire lorsqu'on sait déjà quels numéros ont été tirés ou pas parmi les numéros cochés de 1 à 19 points. Nous reprenons les notations du texte principal. Notons $\bar{\omega}$ la donnée de $(\omega_1,\dots,\omega_{19})$ dont nous disposons après la quinzième étape de la simulation du tirage du Keno ; notons $\bar{n}_{\text{sorties}}$ le nombre de boules sorties parmi les numéros cochés de 1 à 19 (c.-à-d. $\bar{n}_{\text{sorties}} \coloneqq \sum_{i=1}^{19} \omega_i$), et $\bar{s}$ la part du score due à ces boules (c.-à-d. $\bar{s} \coloneqq \sum_{i=1}^{19} \omega_i i$). Notons que $\bar{n}_{\text{sorties}}$ et $\bar{s}$ sont toutes les deux des fonctions de $\bar{\omega}$, autrement dit des v.a. $\mathscr{F}_{15}$-mesurables. Sous la loi conditionnelle relativement à $\mathscr{F}_{19}$, nous savons que $\bar{n}_{\text{sorties}}$ boules ont été tirées parmi les numéros cochés de 1 à 19, mais nous ne savons rien concernant les autres

[‡]. Explication du calcul (il en va de même pour les calculs analogues) :

$$\frac{P(\omega_{(i,j)}=1)}{Q(\omega_{(i,j)}=1)} = \frac{(i+1)^{-1/2} \,/\, ((i+1)^{-1/2}+(j+1)^{-1/2})}{1^{-1/2} \,/\, (1^{-1/2}+(j+1)^{-1/2})} = \frac{1+(j+1)^{-1/2}}{1+(i+1)^{1/2}(j+1)^{-1/2}} = \frac{1+(j+1)^{-1/2}}{1+\sqrt{14}(j+1)^{-1/2}}. \tag{NN}$$

numéros, de sorte qu'il nous reste $20 - \bar{n}_{\text{sorties}}$ boules à tirer parmi les 61 numéros restants. Ainsi, la probabilité conditionnelle que la boule cochée pour 20 points soit tirée vaut $(20 - \bar{n}_{\text{sorties}}) / 61$ — et la probabilité qu'elle ne soit pas tirée vaut donc $(41 + \bar{n}_{\text{sorties}}) / 61$. Sous la première éventualité, le score total sera de $\bar{s} + 20$, et sous la seconde, de $\bar{s}$ ; ce qui correspond, en notant $g$ la fonction qui associe le gain à un score donné, à des gains respectifs de $g(\bar{s} + 20)$ et $g(\bar{s})$. Finalement, l'espérance conditionnelle est donc :

$$\mathbb{E}(gain) = \frac{41 + \bar{n}_{\text{sorties}}}{61} g(\bar{s}) + \frac{20 - \bar{n}_{\text{sorties}}}{61} g(\bar{s} + 20). \tag{NR}$$

## A.3 Le lapin de Douady

**Définition (NS).** Dans cette section, $c$ désigne l'unique solution complexe dont la partie imaginaire soit strictement positive de l'équation :

$$c^3 + 2c^2 + c + 1 = 0. \tag{NT}$$

On a $c \simeq -0{,}123 + 0{,}745\mathrm{i}$. ♡

**Définition (NU).** Un point $(x, y) \in \mathbb{R}^2$, représenté par le nombre complexe $z \coloneqq x + y\mathrm{i} \in \mathbb{C}$, appartient au lapin de Douady si et seulement si la suite $(u_n^{(z)})_{n \in \mathbb{N}}$ définie par :

$$\begin{cases} u_0^{(z)} \coloneqq z; \\ u_{n+1}^{(z)} \coloneqq z^2 + c \end{cases} \tag{NV}$$

reste bornée [§]. ♡

On peut montrer que la suite $(u_n^{(z)})_n$ définie ci-dessus reste bornée dès lors qu'un des éléments de la suite est de module inférieur à 1/4 ; qu'à l'inverse elle diverge à l'infini dès lors qu'un élément est de module supérieur à 2 ; et que pour presque-tout $z$, un de ces deux critères sera effectivement vérifié : ce qui permet donc de trancher si $z$ est ou non dans le lapin de Douady, en calculant les valeurs successives de $u_n^{(z)}$ jusqu'à ce que le module de l'une d'elle sorte de l'intervalle $[1/4, 2]$. Le lapin de Douady a l'allure donnée par la figure 2.1. Comme celle-ci est très irrégulière, il n'est pas sûr qu'un maillage régulier soit le plus adapté ; nous allons donc procéder ici par méthode de Monte-Carlo.

## A.4 SOS analyse complexe : Simulation de $\mathbb{P}$

Nous allons non seulement expliquer comment simuler la loi $\mathbb{P}$ choisie dans l'exemple (CF), mais aussi éclaircir d'où vient le choix de cette loi.

L'idée est de partir d'une loi de Pareto : considérons ainsi la loi $\mathbb{P}_2$ sur $[1, +\infty)$ caractérisée par

$$\mathbb{P}_2(\bullet \geqslant x) = 1/x \qquad \forall x \geqslant 1. \tag{NW}$$

L'avantage de cette fonction est qu'elle est très facile à simuler, puisque sa fonction de répartition complémentaire est immédiatement donnée par $\bar{F}(x) = 1/x$ sur $[1, +\infty)$ ; on peut alors appliquer la méthode de la fonction quantile, observant que

[§]. À noter qu'en l'occurrence, si la suite $(u_n^{(z)})_n$ n'est pas bornée, alors elle diverge nécessairement à l'infini, au sens où $|u_n^{(z)}| \to \infty$ lorsque $n \to \infty$.

la bijection inverse de $\bar{F}$ est $\bar{F}^{-1} : p \mapsto 1/p$ sur $(0,1)$, pour en déduire que si $U$ est uniforme sur $[0,1]$, alors $1/U$ suit la loi $\mathbb{P}_2$. En outre, en dérivant $\bar{F}$, on montre que $\mathbb{P}_2$ est une loi à densité telle que $\mathrm{d}\mathbb{P}_2(x) = \mathbf{1}_{x\geqslant 1} 1/x^2\,\mathrm{d}x$.

Maintenant, on va modifier cette loi de Pareto pour en faire une loi sur $\mathbb{R}$ tout entier. Pour commencer, on considère la loi $\mathbb{P}_1$ qui est la mesure-image de $\mathbb{P}_2$ par $x \mapsto x-1$, c.-à-d. que si $X$ est une v.a. de loi $\mathbb{P}_2$, $X-1$ sera une v.a. de loi $\mathbb{P}_1$. Cela permet immédiatement la simulation de $\mathbb{P}_2$à partir de celle de $\mathbb{P}_1$ ; quant à la densité de $\mathbb{P}_2$, on la calcule à l'aide de la formule de changement de variables :

$$\mathbb{E}_1(\varphi) = \mathbb{E}_2\big(\varphi(X-1)\big) = \int_{\mathbb{R}} \varphi(x-1)\mathbf{1}_{x\geqslant 1} 1/x^2\,\mathrm{d}x \overset{y:=x-1}{=} \int \varphi(y)\mathbf{1}_{y\geqslant 0} 1/(y+1)^2\,\mathrm{d}y, \quad (\texttt{NX})$$

de sorte que $\mathbb{P}_2$ est une loi à densité, avec $\mathrm{d}\mathbb{P}_1(y) = \mathbf{1}_{y\geqslant 0} 1/(y+1)^2\,\mathrm{d}y$.

On peut aussi transformer $\mathbb{P}_1$ en une loi sur $\mathbb{R}_-$, notée $\mathbb{P}_1^-$, qui serait l'image de $\mathbb{P}_1$ par $x \mapsto -x$ (à nouveau, la simulation à partir de $\mathbb{P}_1$ est immédiate) ; la formule de changement de variables nous donne que $\mathrm{d}\mathbb{P}_1^-(x) = \mathbf{1}_{x\leqslant 0} 1/(|x|+1)^2 \mathrm{d}x$ [¶].

Enfin, pour fabriquer une loi sur $\mathbb{R}$ tout entier, on prendra la loi $\mathbb{P} := 1/2 \times \mathbb{P}_1 + 1/2 \times \mathbb{P}_1^-$, dont la densité se calcule immédiatement par $\mathrm{d}\mathbb{P}(x) = 1/2 \times \mathrm{d}Pr_1(x) + 1/2 \times \mathbb{P}_1^-(x) = 1\,/\,2(|x|+1)^2\,\mathrm{d}x$. Quant à la simulation, il suffit de tirer d'abord une variable de Bernoulli de paramètre 1/2, puis, si elle tombe sur 1, de faire une simulation de $\mathbb{P}_1^-$ (indépendante du premier tirage), et si elle tombe sur 0 de faire une simulation de $\mathbb{P}_1$. Pour cette deuxième étape, d'après la façon de simuler $\mathbb{P}_2$ et la définition de $\mathbb{P}_1$ et $\mathbb{P}_1^-$, nous voyons que si $U$ est une loi uniforme sur $[0,1]$, $1/U - 1$ suivra la loi $\mathbb{P}_1$ et que $1 - 1/U$ suivra la loi $\mathbb{P}_1^-$. Au final, cela conduit au programme `SOS.py` donné dans l'exemple `(CF)`.

## A.5 La centrale nucléaire

La valeur moyenne des paramètres $X, Y, Z$, et leur matrice de covariance, sont respectivement

$$\vec{m} := \begin{pmatrix} -3 \\ -4 \\ -5 \end{pmatrix}; \qquad \boldsymbol{C} := \begin{pmatrix} 1 & 0{,}1 & 0{,}2 \\ 0{,}1 & 1 & 0{,}3 \\ 0{,}2 & 0{,}3 & 1 \end{pmatrix} \qquad (\texttt{NY})$$

Pour le calcul de la densité de $\boldsymbol{P}$ par rapport à $\boldsymbol{Q}$, on utilise qu'aussi bien $\boldsymbol{P}$ que $\boldsymbol{Q}$, en tant que lois (multi)normales de matrice de covariance non dégénérée, ont toutes les deux une densité par rapport à la mesure de Lebesgue : par conséquent, la densité de $\boldsymbol{P}$ par rapport à $\boldsymbol{Q}$ sera simplement le quotient de ces deux densités.

La densité d'une loi multinormale (non dégénérée) est un résultat classique sur les vecteurs gaussiens : pour $\mathrm{d}\vec{x}$ un voisinage infinitésimal de $\vec{x} \in \mathbb{R}^3$, on a

$$\begin{aligned} \boldsymbol{P}(\mathrm{d}\vec{x}) &= \mathbb{P}\big(\mathrm{Normale}(\vec{m}, \boldsymbol{C}) \in \mathrm{d}\vec{x}\big) \\ &= \frac{1}{2\sqrt{2\pi^3 \det^{1/2} \boldsymbol{C}}} \exp\big(-\tfrac{1}{2}(\vec{x}^{\mathsf{T}} - \vec{m}^{\mathsf{T}})\boldsymbol{C}^{-1}(\vec{x} - \vec{m})\big)\,\mathrm{vol}(\mathrm{d}\vec{x}), \quad (\texttt{NZ}) \end{aligned}$$

avec une formule similaire pour $\boldsymbol{Q}(\mathrm{d}\vec{x})$ en remplaçant « $\vec{m}$ » par $\vec{0}$. En prenant le quotient, de nombreuses simplifications apparaissent, et on trouve que la densité

[¶]. On prendra garde à ne pas oublier que pour une intégrale "au sens de la mesure", qui n'a pas de sens de parcours, le changement de variables $y = f(x)$ donne $\mathrm{d}y = |Df(x)|\mathrm{d}x$, avec une valeur absolue !

relative est

$$\left(\frac{\mathrm{d}\boldsymbol{Q}}{\mathrm{d}\boldsymbol{P}}\right)(\vec{x}) = \frac{\exp\left(-\frac{1}{2}(\vec{x}^{\mathsf{T}} - \vec{m}^{\mathsf{T}})\boldsymbol{C}^{-1}(\vec{x} - \vec{m})\right)}{\exp\left(-\frac{1}{2}\vec{x}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x}\right)}$$
$$= \exp\left(\tfrac{1}{2}(-\vec{x}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x} + \vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x} + \vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x} - \vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{m} + \vec{x}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x})\right)$$
$$= \exp\left(\vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x} - \tfrac{1}{2}\vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{m}\right) \quad (\mathtt{OA})$$

(où nous avons utilisé que, puisque $\vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x}$ est un nombre réel et $\boldsymbol{C}^{-1}$ une matrice symétrique, $\vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x} = (\vec{m}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{x})^{\mathsf{T}} = \vec{x}^{\mathsf{T}}(\boldsymbol{C}^{-1})^{\mathsf{T}}\vec{m}u = \vec{x}^{\mathsf{T}}\boldsymbol{C}^{-1}\vec{m}u$).

## A.6 Le yahtzee de la mort : Description des stratégies

En premier lieu, il convient de remarquer qu'un certain nombre de principes évidents s'imposent concernant le choix des dés à relancer à relancer, au-delà de toute stratégie particulière :

1. Si un dé affiche un score strictement plus grand qu'un autre, il ne faut jamais choisir de relancer le premier et pas le second ;
2. Il faut toujours relancer un dé affichant '1' ;
3. Il ne faut jamais relancer un dé affichant '6' ;
4. Il faut toujours relancer assez de dés pour avoir une chance d'obtenir au moins 24 dès le tour suivant ;
5. Si le total des dés atteint déjà 24, il faut s'arrêter [‖].

Cela étant posé, les stratégies respectives de Charlotte et Xénia consistent, sous réserve du respect des règles ci-dessus, à suivre les principes suivants :

**Charlotte :** À la première étape, on garde un dé si et seulement s'il affiche une valeur d'au moins 5 ; à la seconde étape, on garde un dé si et seulement s'il affiche une valeur qu'au moins 4.

**Xénia :** On garde autant de dés que possible dont la moyenne des valeurs soit au moins de $\frac{24}{5}$.

[‖]. Les conditions 2 et 5 peuvent parfois être en contradiction, mais dans ce cas, c'est qu'on est déjà assuré de l'emporter : on peut donc résoudre la contradiction de façon arbitraire sans changer le résultat, par exemple en disant que la condition 5 l'emporte sur la condition 2.