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

Rémi Peyre

printemps 2014

Rappels de probabilités

0.1 Espaces métriques séparables

Définition 0.1 (Séparabilité)**.** Un espace métrique E est dit *séparable*[1] quand il existe une partie $X \subset E$ dénombrable telle que tout point de E puisse s'écrire comme la limite d'une suite d'éléments de X. (c'est-à-dire telle que X soit *dense* dans E).

Proposition 0.2. *Les espaces $\mathbb{R}^n$, muni de n'importe quelle métrique, sont séparables.*

Proposition 0.3. *Si K est un espace métrique compact et E un espace métrique séparable, alors l'ensemble $\mathcal{C}(K, E)$ des fonctions continues de K dans E, muni de la métrique de la convergence uniforme, est séparable.*

Proposition 0.4. *Un sous-espace d'un espace métrique séparable est séparable.*

En probabilités, il faut toujours travailler sur des espaces séparables. En pratique ce n'est pas un problème, car les espaces que l'on rencontre "couramment" le sont presque toujours.

0.2 Notions de convergence

0.2.1 Convergences de variables aléatoires

☛ *Dans toute cette section, les variables aléatoires sont considérées à équivalence presque-sure près.*

Définition 0.5 (Convergence presque-sure)**.** Pour $(\Omega, \mathbb{P})$ un espace probabilisé, soient $(X_n)_{n\in\mathbb{R}}$ et X_∞ des variables aléatoires sur Ω à valeurs dans un espace métrique E. On dit que X_n *converge presque-surement* vers X_∞ quand

$$\mathbb{P}\big(\{\omega \mid X_n(\omega) \overset{n\to\infty}{\to} X_\infty(\omega)\}\big) = 1.$$

Définition 0.6 (Convergence L^p)**.** Pour $(\Omega, \mathbb{P})$ un espace probabilisé, soient $(X_n)_{n\in\mathbb{R}}$ et X_∞ des variables aléatoires sur Ω à valeurs dans un espace métrique E. On dit que X_n *converge en norme L^p* vers X_∞, pour $1 \leqslant p < \infty$, quand

$$\mathbb{E}\big(dist(X_n(\omega), X_\infty(\omega))^p\big) \overset{n\to\infty}{\to} 0.$$

1. Ne pas confondre avec la notion d'espace topologique *séparé* : les deux notions n'ont rien à voir...

Définition 0.7 (Convergence L^∞)**.** Pour $(\Omega, \mathbb{P})$ un espace probabilisé, soient $(X_n)_{n\in\mathbb{R}}$ et X_∞ des variables aléatoires sur Ω à valeurs dans un espace métrique E. On dit que X_n *converge en norme* L^∞ *(ou « uniformément »)* vers X_∞ quand

$$\inf\big\{\varepsilon \in \mathbb{R}_+ \;\big|\; \mathbb{P}\big(dist(X_n(\omega), X_\infty(\omega)) \leqslant \varepsilon\big) = 1\big\} \overset{n\to\infty}{\to} 0.$$

Définition 0.8 (Convergence en probabilité)**.** Pour $(\Omega, \mathbb{P})$ un espace probabilisé, soient $(X_n)_{n\in\mathbb{R}}$ et X_∞ des variables aléatoires sur Ω à valeurs dans un espace métrique E. On dit que X_n *converge en probabilité* vers X_∞ quand

$$\forall \varepsilon > 0 \qquad \mathbb{P}\big(dist(X_n(\omega), X_\infty(\omega)) \leqslant \varepsilon\big) \overset{n\to\infty}{\to} 1.$$

0.2.2 Convergence de lois

Définition 0.9 (Convergence de lois)**.** Soit E un espace métrique séparable, et soient $(\mu_n)_{n\in\mathbb{R}}$ et μ_∞ des lois (càd. des mesures de probabilité) sur E. On dit que la lois μ_n converge (on précise parfois « converge en loi », ou « converge faiblement ») vers μ_∞ quand, pour toute fonction f continue et bornée sur E, $\int_E f(x)\, d\mu_n(x) \overset{n\to\infty}{\to} \int_E f(x)\, d\mu_\infty(x)$.

Théorème 0.10 (Convergence en probabilité et convergence en loi)**.** *Soient $(X_n)_{n\in\mathbb{R}}$ et X_∞ des variables aléatoires définies sur un même espace probabilisé $(\Omega, \mathbb{P})$, à valeurs dans un espace métrique séparable E ; notons $(\mu_n)_n$ et μ_∞ les lois respectives de ces variables aléatoires. Si la variable aléatoire X_n converge en probabilité vers X_∞ quand $n \to \infty$, alors la loi μ_n converge vers μ_∞.*

Théorème 0.11 (Convergence vers une constante)**.** *Soient $(X_n)_{n\in\mathbb{R}}$ des variables aléatoires définies sur un même espace probabilisé $(\Omega, \mathbb{P})$, à valeurs dans un espace métrique séparable E ; notons $(\mu_n)_n$ les loi srespectives de ces variables aléatoires. Soit $x \in E$. Si les lois μ_n convergent vers la δ_x (càd. la masse de Dirac en x), alors les variables X_n convergent en probabilité vers la variable aléatoire constante égale à x.*

Remarque 0.12. On voit donc que pour des variables aléatoires, converger en probabilité vers une constante ne dépend que de la loi individuelle de chacune des variables aléatoires, et pas de leur loi jointe. Il en va de même pour la convergence L^p vers une constante (y compris pour $p = \infty$), puisque la distance L^p entre X_n et c ne dépend que de la loi de X_n.

0.3 Loi des grands nombres

Théorème 0.13 (Loi des grands nombres, version fréquentielle)**.** *Soient, sur un espace probabilisé $(\Omega, \mathbb{P})$, des variables aléaoires de Bernoulli $(X_i)_{i\in\mathbb{R}}$ indépendantes et de même paramètre p. Alors la variable aléatoire $\bar{X}_n := n^{-1} \sum_{i=0}^{n-1} X_i$ converge en probabilité vers la constante p.*

Théorème 0.14 (Variantes de la loi des grands nombres, version fréquentielle)**.** *Dans le théorème précédent, la convergence de $\bar{X}_n$ vers p a en fait lieu aussi dans $L^p(\mathbb{P})$ pour tout $p < \infty$, ainsi que presque-surement.*

Théorème 0.15 (Loi "faible" des grands nombres)**.** *Soient, sur un espace probabilisé* $(\Omega, \mathbb{P})$*, des variables aléaoires réelles* $(X_i)_{i\in\mathbb{R}}$ *indépendantes et de même loi. Supposons que cette loi soit telle que* X_n *soit intégrable et notons* $\mathbb{E}(X_n) =: m$ *(cette valeur ne dépendant pas de* m*). Alors la variable aléatoire* $\bar{X}_n := n^{-1}\sum_{i=0}^{n-1} X_i$ *converge en probabilité vers la constante* m*.*

Théorème 0.16 (Loi L^1 des grands nombres)**.** *Dans le théorème précédent, la convergence de* $\bar{X}_n$ *vers* m *a en fait lieu aussi dans* $L^1(\mathbb{P})$*.*

Théorème 0.17 (Loi "forte" des grands nombres)**.** *Toujours dans le théorème 0.15,* $\bar{X}_n$ *converge en fait aussi presque-surement vers* m*.*

0.4 Concepts statistiques

Définition 0.18 (Modèle statistique)**.** Un modèle statistique est la donnée d'un espace probabilisable Ω et d'une famille de lois $(\mathbb{P}_\theta)_{\theta\in\Theta}$ sur Ω. On appelle θ le *paramètre* ; et Θ l'*espace des paramètres*. Une *expérience statistique* est le tirage au sort d'une variable aléatoire X selon une loi $\mathbb{P}_\theta$ où le paramètre θ, quoique fixé, est *inconnu*. La valeur tirée de X est appelée *observation*.

Définition 0.19 (Paramètre)**.** Un *paramètre du modèle* est, en toute généralité, une fonction *déterministe* de θ.

Définition 0.20 (Estimateur)**.** Un *estimateur* est, en toute généralité, une fonction *déterministe* de l'observation X.

Définition 0.21 (Estimateur convergent)**.** Une famille d'estimateurs $(T_n)_{n\in\mathbb{R}}$ étant donnée, on dit que $T_n(X)$ est une estimateur *convergent* d'un paramètre du modèle $\lambda(\theta)$ quand, *quelle que soit la valeur du paramètre* $\theta \in \Theta$, sous la loi $\mathbb{P}_\theta$, la variable aléatoire $T_n(X)$ converge en probabilité vers $\lambda(\theta)$ quand $n \to \infty$.

Définition 0.22 (Convergence L^p d'un estimateur)**.** On dit que l'estimateur $T_n(X)$ *converge en norme* L^p d'un paramètre du modèle $\lambda(\theta)$ quand, sous la loi $\mathbb{P}_\theta$, la variable aléatoire $T_n(X)$ converge dans L^p vers $\lambda(\theta)$ quand $n \to \infty$.

Définition 0.23 (Estimateur fortement convergent)**.** On dit que $T_n(X)$ est une estimateur *fortement convergent* d'un paramètre du modèle $\lambda(\theta)$ quand, sous la loi $\mathbb{P}_\theta$, la variable aléatoire $T_n(X)$ converge *presque-surement* vers $\lambda(\theta)$ quand $n \to \infty$.

Chapitre 1

Principe de la méthode de Monte-Carlo

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

1.1.1 Estimation d'une probabilité par la méthode de rejet

Exemple 1.1 (Le championnat de basket). Un passionné de basket a évalué le niveau des seize équipes impliquées dans le championnat national et modélisé la probabilité du résultat des matchs entre les différentes équipes. Dans sa modélisation, les équipes sont numérotées de 1 (la plus forte) à 16 (la plus faible), et la probabilité que l'équipe i gagne contre l'équipe j vaut $j^{1/2}$ / $(i^{1/2} + j^{1/2})$ [il n'y a jamais de match nul en basket], les résultats des différentes rencontres du championnat étant indépendants. Le championnat consiste à ce que chaque équipe rencontre une fois chaque autre, et l'équipe championne est celle qui a gagné le plus de matches (en cas d'ex-æquo, on tire au sort la vainqueure parmi les premières du classement).

Note passionné cherche à estimer la probabilité que Nancy, qui est l'équipe #3, gagne le championnat. Mais les calculs théoriques seraient inextricables... Il décide donc *simuler* aléatoirement un grand nombre de réalisations du championnat (disons 60 000), de compter la fréquence *empirique* avec laquelle Nancy l'emporte sur l'ensemble de ces simulations, et de considérer cette fréquence comme une *approximation* raisonnable de la probabilité véritable de victoire : c'est ce qu'on appelle la *méthode de Monte-Carlo*. Le programme correspondant est donné dans l'annexe A.1.1 ; son exécution donne :

```
>> championnat
Probabilité de victoire évaluée pour l'équipe de Nancy :
    0.1086
```

Définition 1.2. Supposons qu'on cherche à évaluer la probabilité $\mathbb{P}(A) =: p$ d'un évènement A sous une loi $\mathbb{P}$ qu'on sait simuler. La *méthode de Monte-Carlo* consiste alors à réaliser un grand nombre N de simulations indépendantes de la loi $\mathbb{P}$, à compter le nombre n de ces simulations pour lesquelles A a eu lieu, et à estimer $\mathbb{P}(A)$ par la

proportion $\hat{p}_N$ de telles simulations :

$$\hat{p}_N := n \,/\, N.$$

Théorème 1.3. *Quand N tend vers l'infini, l'estimateur $\hat{p}_N$ est convergent vers p.*

Démonstration. C'est simplement une reformulation de la version fréquentielle de la loi des grands nombres. □

Remarque 1.4. Dans le théorème 1.3, on peut même préciser que $\hat{p}_N$ converge *fortement*. Il y a aussi convergence L^p pour tout $p < \infty$

1.1.2 Estimation d'une espérance

La méthode présentée dans la définition 1.2 est en fait un cas particulier de la méthode générale de Monte-Carlo, qui permet d'évaluer des *espérances* :

Définition 1.5. Supposons qu'on cherche à évaluer l'espérance $\mathbb{E}(f) =: m$ d'une variable aléatoire f de classe L^1 sous une loi $\mathbb{P}$ qu'on sait simuler. La *méthode de Monte-Carlo* consiste alors à réaliser une grand nombre N de simulations de la loi $\mathbb{P}$, à regarder la valeur $f(\omega_i)$ de f pour chacune de ces simulations, et à estimer $\mathbb{E}(f)$ par la moyenne empirique $\hat{m}_N$ de ce valeurs :

$$\hat{m}_N := N^{-1} \sum_{i=1}^{N} f(\omega_i).$$

Remarque 1.6. Dans le cas particulier où f est l'indicatrice $\mathbf{1}_{\{A\}}$ d'un évènement A, la définition 1.5 redonne la définition 1.2.

Théorème 1.7. *L'estimateur $\hat{m}$ converge (fortement) vers $\mathbb{E}(f)$, et converge aussi dans L^1.*

Démonstration. C'est encore la loi (forte) des grands nombres, cette fois-ci dans sa version générale. □

Remarque 1.8. La méthode de Monte-Carlo est donc une façon d'évaluer une quantité *déterministe* (l'espérance) par un procédé *aléatoire* (les simulations) ! Ce paradoxe montre que le hasard peut parfois être un atout plutôt qu'un handicap...

Remarque 1.9. Attention, les théorèmes 1.3 et 1.7 ne sont valables en toute rigueur que si les simulations ω_i sont *exactement* i.i.d. $\mathbb{P}$. En pratique, les calculs informatiques utilisent généralement des nombres seulement *pseudo*-aléatoires, ce qui peut faire échouer la méthode de Monte-Carlo dans les cas les plus extrêmes.

Exemple 1.10 (La loterie). La loterie nationale a décidé de lancer un nouveau jeu. Les joueurs doivent parier sur douze numéros différents entre 1 et 36 en faisant un pari à 12 points, un pari à 11 points, un pari à 10 points, etc. Le soir du tirage, la loterie tire douze numéros au sort, et les joueurs comptent le total des points qu'ils ont marqués [1] Une joueuse gagne si elle a marqué au moins 39 points, le montant remporté G dépendant du score s selon la formule

$$G(s) = \mathbf{1}_{\{p \geqslant 39\}} \lfloor 10 e^{(p-39)/8} \rfloor \ :$$

(La joueuse remporte ainsi 10 € pour 39 points, 11 € pour 40 points, ..., 39 € pour 50 points, ..., et jusqu'à 1 309 € pour un score parfait de 78 points correspondant à avoir coché tous les bons numéros.

La question qui se pose évidemment pour la loterie nationale est : "à quel prix vendre ce jeu ?". En effet, il faut que le prix d'achat des cartons compense le montant des paiements, autrement dit qu'il dépasse l'*espérance de gain* d'une joueuse... Calculer cette espérance de gain est trop complexe ; par contre, on peut la simuler (notez que l'espérance de gain ne dépend pas des numéros cochés), ce que fait le programme de l'annexe A.2.1. L'exécution donne, pour 100 000 simulations :

```
>> loterie
Gain moyen évalué :
    2.3409
```

1.1.3 Application au calcul d'intégrales

La méthode de Monte-Carlo peut aussi être utilisée pour calculer des intégrales, en réécrivant celles-ci sous forme d'espérances par la technique suivante :

Proposition 1.11. *Pour f une fonction intégrable sur $\mathbb{R}^d$, et $\mathbb{P}$ une mesure de probabilité sur $\mathbb{R}^d$ ayant par rapport à la mesure de Lebesgue une densité $d\mathbb{P}/dx$ qui est non nulle partout où f est non nulle, on a :*

$$\int_{\mathbb{R}^d} f(x)\,dx = \mathbb{E}\big((d\mathbb{P}/dx)^{-1} f\big).$$

Cela permet alors d'utiliser l'estimateur de Monte-Carlo (1.5) *pour évaluer $\int f(x)\,dx$.*

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

1. Par exemple, si un joueuse a parié 12 pts sur '1', 11 pts sur '2', 10 pts sur '3' etc., et qu'il sort les numéros '3', '6', '9', ... et '36', alors elle marque 22 points : 10 points pour '3', 7 points pour '6', 4 points pour '9' et 1 point pour '12'.

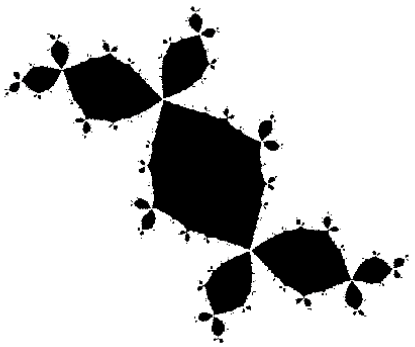

FIGURE 1.1 – Le lapin de Douady.

Remarque 1.12. En fait, il n'y a essentiellement aucune différence entre une espérance et une intégrale. Dans un sens en effet, la théorie de la mesure nous dit qu'une espérance n'est rien d'autre qu'une intégrale par rapport à une mesure *générale* (quand celle-ci est de probabilité) ; et dans l'autre, une intégrale par rapport à la mesure de Lebesgue peut toujours s'écrire comme une espérance, en introduisant une mesure de probabilité $\mathbb{P}$ à densité partout non nulle par rapport à dx et en appliquant (1.11). Dans la suite de ce cours, tout ce que nous dirons sur l'application de la méthode de Monte-Carlo pour une espérance s'appliquera donc aussi au cas des intégrales, et réciproquement.

Remarque 1.13. Il y a très nombreuses possibilités pour choisir la loi $\mathbb{P}$ destinée à appliquer la proposition 1.11, qui conduisent à autant d'estimateurs de Monte-Carlo *différents* pour la *même* intégrale ! Certains de ces estimateurs s'avèreront plus efficaces que d'autres, ce qui rendra le choix de $\mathbb{P}$ particulièrement important. Quand c'est pour évaluer une *espérance* qu'on utilise la méthode de Monte-Carlo, il y a évidemment un choix de $\mathbb{P}$ naturel, à savoir la loi sous laquelle l'espérance est prise ; cependant nous verrons qu'on peut quand même changer la loi d'échantillonnage, et ce parfois avec profit : cela correspond à ce qu'on appelle l'*échantillonnage préférentiel*, dont nous parlerons dans la § 2.2.

Exemple 1.14 (Le lapin de Douady). Le lapin de Douady est un objet fractal qu'on obtient de la façon suivante. On pose $c :\simeq -0{,}123 + 0{,}745i$ qui est la solution imaginaire positive de l'équation "$(c^2 + c)^2 + c = 0$". Un point $z \in \mathbb{C}$ appartient à l'ensemble de Douady si et seulement si la suite définie par $u_0 = z$ et $u_{n+1} = u_n^2 + c$ ne diverge pas : on peut montrer que cela est le cas dès qu'une itérée de la suite est inférieure à $1/4$ en module, qu'à l'inverse la suite diverge dès qu'une itérée dépasse 2 en module, et que presque-tout point de départ finit par se retrouver dans un de ces deux cas, ce qui permet de trancher. Le lapin de Douady a l'allure donnée par la figure **??**. 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.

L'aire de cette figure, est définie comme $\int_{\mathbb{R}^2} \mathbf{1}_{\{x+yi \in lapin\}}\, dxdy$; c'est donc à priori une intégrale et pas du tout une espérance. Voyons comment nous pouvons la réécrire comme une espérance.

La définition du lapin de Douady nous assure que la figure est complètement conte-

nue dans le carré $[-2,2]\times[-2,2]$. Nous pouvons donc réécrire l'aire comme

$$\int_{(x,y)\in[-2,2]\times[-2,2]} \mathbf{1}_{\{x+yi\in lapin\}}\, dxdy.$$

Soit maintenant $\mathbb{P}$ la probabilité uniforme sur la carré $[-2,2]\times[-2,2]$. Cette probabilité est facile à simuler : en effet, il suffit de tirer deux variables aléatoires U et V uniformes sur $[0,1]$ et indépendantes, et tirer alors $(x,y) = (-2+4U, -2+4V)$. En outre, la densité de $\mathbb{P}$ par rapport à la mesure de Lebsegue est immédiate à calculer, et vaut $1/16$ uniformément sur le carré.

Cela conduit au programme donné dans l'annexe A.3.1 pour estimer l'aire du lapin. Une exécution m'a donné :

```
>> lapin
Aire estimée du lapin de Douady :
    1.3075
```

Exemple 1.15 (SOS analyse complexe). Voici un autre exemple d'application de la méthode de Monte-Carlo au calcul d'une intégrale. Dans cet exemple, un ingénieur a été amené à calculer à la main l'intégrale

$$I := \int_0^\infty \frac{\sin x}{(x^2+1)x}\, dx.$$

Il a trouvé un résultat de $\pi/2 \simeq 1{,}571$, mais il a un doute, étant donné que sa petite sœur en licence de mathématiques prétend quand à elle que le résultat correct est $\pi(1-1/e)/2 \simeq 0{,}993...$ L'idée vient alors à notre ingénieur de vérifier le résultat numériquement, en évaluant I par la méthode de Monte-Carlo (car il n'a pas sous la main de logiciel spécialisé en calcul numérique).

Dans un premier temps, il faut choisir une probabilité $\mathbb{P}$ sur $\mathbb{R}$ ayant une densité non nulle partout où l'intégrande est non nulle (càd. sur $\mathbb{R}_+$ — à un ensemble négligeable près), qui soit facile à simuler et dont la densité soit facile à calculer. Notre ingénieur pense à la densité de Pareto sur caractérisée par

$$\mathbb{P}(\geqslant x) = \frac{1}{1+x} \quad \forall x \geqslant 0,$$

qui se simule par $1/U - 1$ où U est une variable uniforme sur $[0,1]$, et dont la densité est

$$d\mathbb{P}/dx = \mathbf{1}_{\{}x \geqslant 0/(1+x)^2.$$

On peut alors écrire

$$I = \mathbb{E}_{\mathbb{P}}\left(\frac{(1+x)^2\sin x}{(x^2+1)x}\right),$$

ce qui conduit au programme de l'annexe **??**. Pour 1 000 000 simulations, on trouve :

```
>> integrale
Évaluation de I :
    0.9939
```

Clairement, c'est donc la petite sœur qui avait raison [2].

2. Moralité : Il faut toujours écouter sa petite soeur ! ;-)

1.2 Intervalles de confiance

1.2.1 Intervalle de confiance pour une espérance calculée par Monte-Carlo

Savoir que l'estimateur de Monte-Carlo converge est une bonne chose, mais tant qu'on n'a pas d'information sur la vitesse de sa convergence, l'interprétation du résultat est essentiellement impossible... Il est donc important d'être capable non seulement de fournir une estimation pour $\mathbb{E}(f)$, mais aussi d'indiquer quelle est sa *précision*.

Bien entendu, l'estimateur de Monte-Carlo est aléatoire, et de ce fait il peut *théoriquement* donner un résultat complètement aberrant : par exemple, si on cherche à évaluer la probabilité qu'une pièce non truquée tombe sur "pile", il y a une probabilité non nulle (quoique extrêmement faible) que tous les lancers faits pour évaluer la probabilité donnent "face" et donc qu'on évalue à tort la probabilité de "pile" comme étant nulle ! Nous ne pourrons donc pas donner un intervalle de *certitude* pour la valeur réelle de $\mathbb{E}(f)$, mais seulement un intervalle de *confiance*, c'est-à-dire un intervalle dont on sait qu'il contiendra effectivement $E(f)$ avec une très grande probabilité p fixée à l'avance (p.ex. $p = 95\ \%$).

C'est à la construction de tels intervalles de confiance que s'intéresse cette section.

Théorème 1.16. *Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo* avec f de classe L^2*, et notons σ la variance (supposée non nulle) de f. Alors, quand $N \to \infty$, la probabilité que $\mathbb{E}(f)$ soit effectivement dans l'intervalle $[\hat{m}+c_-\sigma N^{-1/2}$, $\hat{m} + c_+\sigma N^{-1/2}]$ converge vers $\mathbb{P}\big(\mathcal{N}(1) \in [c_-, c_+]\big) = (2\pi)^{-1/2} \int_{c_-}^{c^+} e^{-x^2/2}\, dx$. En particulier,*

$$\varliminf_{N\to\infty} \mathbb{P}\big(\mathbb{E}(f) \in [\hat{m} + c_-\sigma N^{-1/2}, \hat{m} + c_+\sigma N^{-1/2}]\big) \geqslant (2\pi)^{-1/2} \int_{c_-}^{c^+} e^{-x^2/2}\, dx \ :$$

on dit que $[\hat{m} + c_-\sigma N^{-1/2}, \hat{m} + c_+\sigma N^{-1/2}]$ est un intervalle de confiance asymptotique à la probabilité $(2\pi)^{-1/2} \int_{c_-}^{c^+} e^{-x^2/2}\, dx$ *pour* $\mathbb{E}(f)$.

Démonstration. Que $\mathbb{E}(f)$ soit dans l'intervalle $[\hat{m}+c_-\sigma N^{-1/2}, \hat{m}+c_+\sigma N^{-1/2}]$ est équivalent à ce que $\sigma^{-1}N^{1/2}(\mathbb{E}(f) - \hat{m})$ soit dans $[c_-, c_+]$. Or

$$\sigma^{-1}N^{1/2}(\mathbb{E}(f) - \hat{m}) = N^{-1/2} \sum_{i=1}^{N} \sigma^{-1}(\mathbb{E}(f) - f(\omega_i)),$$

où les $\sigma^{-1}(E(f) - f(\omega_i))$ sont des v.a. i.i.d. L^2 centrées et de variance unitaire. Le théorème-limite central dit alors que $\sigma^{-1}N^{1/2}(\mathbb{E}(f) - \hat{m})$ converge en loi vers $\mathcal{N}(1)$, d'où le résultat annoncé. □

En pratique, σ n'est généralement pas connu. Il faut alors utiliser le théorème suivant :

Corollaire 1.17. *Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo avec f de classe L^2. Notons $\hat{\sigma}$ la* variance empirique *de f à l'issue des N simulations, autrement dit la variance qu'aurait f sous la loi uniforme sur les ω_i :*

$$\hat{\sigma} := \left(N^{-1} \sum_{i=1}^{N} f(\omega_i)^2 - \left(N^{-1} \sum_{i=1}^{N} f(\omega_i) \right)^2 \right)^{1/2}.$$

Alors, quand $N \to \infty$, l'intervalle $[\hat{m} + c_- \hat{\sigma} N^{-1/2}, \hat{m} + c_+ \hat{\sigma} N^{-1/2}]$ est un intervalle de confiance asymptotique à la probabilité $\mathbb{P}\big(\mathcal{N}(1) \in [c_-, c_+]\big)$ pour $\mathbb{E}(f)$.

Démonstration. De même que pour la preuve du théorème 1.16, nous allons démontrer que $\hat{\sigma}^{-1} N^{1/2} (\mathbb{E}(f) - \hat{m})$ converge en loi vers $\mathcal{N}(1)$. Nous observons que $\hat{\sigma}^{-1} N^{1/2} (\mathbb{E}(f) - \hat{m}) = \sigma^{-1} N^{1/2} (\mathbb{E}(f) - \hat{m}) \,/\, \sigma^{-1}\hat{\sigma}$, où $\sigma^{-1}\hat{\sigma}$ converge en probabilité vers la constante 1 d'après la loi des grands nombres.

Nous allons maintenant utiliser un lemme élémentaire concernant la convergence en loi : « si X_N est une variable aléatoire convergeant en loi vers X, que Λ_N est une variable aléatoire convergeant en probabilité vers la *constante* λ_0, et que Φ est une application continue en tous les points de la forme (x, λ), alors $\Phi(X_N, \Lambda_N)$ converge en loi vers $\Phi(X, \lambda_0)$ ». En appliquant ce lemme avec $X_N = \sigma^{-1} N^{1/2} (\mathbb{E}(f) - \hat{m})$, $X = \mathcal{N}(1)$, $\lambda_N = \sigma^{-1}\hat{\sigma}$, $\lambda_0 = 1$ et $\Phi(x, \lambda) = x \,/\, \lambda$, on trouve alors le résultat annoncé. □

Remarque 1.18. Attention, les théorèmes 1.16 et 1.17 exigent pour être valides *que f soit de classe L^2*, ce qui est strictement plus fort que la condition L^1 pour avoir la convergence de l'estimateur de Monte-Carlo. Dans la suite de ce cours, nous ne considèrerons plus que le cas où f est effectivement de classe L^2, vu que c'est ce qui arrive le plus souvent ; toutefois il y a aussi des situations où la condition L^2 n'est pas satisfaite, et alors il ne faut surtout pas utiliser le théorème 1.17 sous peine d'obtenir des résultats complètement aberrants !

Remarque 1.19. Bien qu'on parle souvent simplement d'« intervalle de confiance à la probabilité p », il convient de garder à l'esprit que les intervalles de confiance que donne le théorème 1.17 sont seulement *asymptotiques*, càd. que la probabilité de confiance p n'est valable que quand N tend vers l'infini. En pratique, on pose comme « acte de foi » que N a été choisi suffisamment grand pour que l'intervalle de confiance trouvé soit presque de probabilité p, mais il peut y avoir des cas où c'est complètement faux...

Remarque 1.20. En première approximation, on pourra retenir la règle empirique suivante : si on souhaite obtenir un intervalle de confiance de niveau $1-\varepsilon$, il faut procéder à au moins $0{,}001\varepsilon^{-2}$ simulations pour pouvoir considérer que le niveau réel de l'intervalle de confiance a à peu près atteint sa valeur asymptotique. Attention ; cette règle n'est qu'une règle empirique très grossière qui ne fonctionne que dans les cas « gentils » et peut facilement être mise en défaut.

Pour pouvoir donner une valeur numérique à la probabilité de confiance ou à la largeur de l'intervalle, il faut disposer de la table de répartition de la loi normale. Nous donnons dans la figure 1.1 deux mini-tables de correspondance entre intervalle de confiance et probabilité de confiance dans le cas où $c_- = -c_+$. De manière générale, on peut calculer ce genre de tables à l'aide de ce qu'on appelle la *fonction d'erreur* erf, qui

est implémentée sur la plupart des systèmes de calcul numérique (par exemple, sous MATLAB on l'obtient par "`erf`") : on a $\mathbb{P}\big(\mathcal{N}(1) \leqslant c\big) = \big(1 + \mathrm{erf}(c/\sqrt{2})\big)/2$. L'inverse erf^{-1} de la fonction d'erreur (qu'on obtient par "`erfinv`" sous MATLAB) permet de faire le cheminement inverse : $c = \sqrt{2}\,\mathrm{erf}^{-1}(2p-1)$ est le seuil tel que $\mathbb{P}\big(\mathcal{N}(1) \leqslant c\big) = p$.

Exemple 1.21 (La loterie, suite). Reprenons le problème de la loterie que nous avons traité tout-à-l'heure (exemple 1.10). La loterie nationale a évalué que les gains des joueurs s'élèveraient à environ 2,34 €€ par carton. Elle envisage donc de vendre chaque carton pour 2,50 €€, ce qui, compte tenu des frais, lui rapportera un chiffre d'affaires de 2,40 €€ par carton. Mais il s'agit de ne pas s'être trompée sur l'évaluation des gains... Car une erreur de seulement 6 c€€ par carton pourrait amener la loterie à se retrouver globalement perdante, ce qui entrainerait sa banqueroute à long terme ! Il faut donc assortir l'évaluation des gains d'une marge d'erreur permettant de maitriser les risques inhérents à la méthode de Monte-Carlo.

Le programme de l'annexe A.2.2 donne ainsi l'implémentation de la méthode de Monte-Carlo assortie d'un intervalle de confiance à 99,5%. Remarquons que l'utilisation de la méthode exposée ci-dessus pour l'intervalle de confiance vérifie effectivement les conditions nécessaires requises pour que ce soit pertinent : d'une part, comme les gains des joueurs sont bornés, ils sont à fortiori L^2. En outre, nous avons requis un niveau de confiance de 1 %, pour lequel la règle empirique de la remarque 1.20 nous suggère de procéder à au moins 40 simulations : avec nos $7 \cdot 10^4$ simulations, autant dire que nous avons de la marge !

On obtient le résultat suivant :

```
>> loterie_IC
Intervalle de confiance pour le gain moyen :
2.3409 ± 0.033082
```

L'intégralité de cet intervalle de confiance étant compris en-dessous de 2,40, la loterie peut en conclure qu'avec un risque d'erreur d'au plus 5 ‰, elle rentrera dans ses fonds !

Remarque 1.22. En l'occurrence, puisqu'ici la seule chose qui inquiète les organisateurs est le risque que le gain moyen des joueurs *dépasse* 2,40 €€, ils ne sont pas obligés de prendre un intervalle de confiance *symétrique* autour de la valeur estimée, mais pourraient plutôt faire un *test à droite* sur le fait que la vraie moyenne soit effectivement en dessous de 2,40. En effet, ici, d'après la symétrie de la loi normale, on a estimé qu'il y avait 2,5 ‰ de risques que la vrai gain moyen soit en dessous de $2{,}341 - 0{,}033$ €€ (ce qui ne nous dérangerait en fait pas !) et 2,5 ‰ de risques que la vrai gain moyen soit au-dessus de $2{,}341 + 0{,}033$ (et donc potentiellement au-delà de 2,40) : la p-valeur d'un test asymétrique serait donc en fait ici (inférieure à) 2,5 ‰ plutôt que 5 ‰. Toutefois, ici nous préférons nous concentrer sur la notion d'intervalle de confiance, pour illustrer la façon dont le résultat de la méthode de Monte-Carlo converge vers la vraie valeur quand le nombre de simulations augmente.

1.2.2 Notion d'efficacité

Le théorème 1.16 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

c	$\mathbb{P}\big(\lvert\mathcal{N}(1)\rvert > c\big)$
0	1
0,25	0,81
0,5	0,62
0,75	0,46
1	0,32
1,25	0,22

c	$\mathbb{P}\big(\lvert\mathcal{N}(1)\rvert > c\big)$
1,5	0,14
1,75	$8{,}1 \cdot 10^{-2}$
2	$4{,}6 \cdot 10^{-2}$
2,25	$2{,}5 \cdot 10^{-2}$
2,5	$1{,}3 \cdot 10^{-2}$
2,75	$6{,}0 \cdot 10^{-3}$

c	$\mathbb{P}\big(\lvert\mathcal{N}(1)\rvert > c\big)$
3	$2{,}7 \cdot 10^{-3}$
3,25	$1{,}2 \cdot 10^{-3}$
3,5	$4{,}7 \cdot 10^{-4}$
3,75	$1{,}8 \cdot 10^{-4}$
4	$6{,}4 \cdot 10^{-5}$
∞	0

c	$\mathbb{P}\big(\lvert\mathcal{N}(1)\rvert > c\big)$
0	1
0,68	0,5
1,29	0,2
1,65	0,1
1,96	0,05

c	$\mathbb{P}\big(\lvert\mathcal{N}(1)\rvert > c\big)$
2,33	0,02
2,58	0,01
2,81	$5 \cdot 10^{-3}$
3,10	$2 \cdot 10^{-3}$
3,30	$1 \cdot 10^{-3}$

c	$\mathbb{P}\big(\lvert\mathcal{N}(1)\rvert > c\big)$
3,49	$5 \cdot 10^{-4}$
3,72	$2 \cdot 10^{-4}$
3,90	$1 \cdot 10^{-4}$
∞	0

TABLE 1.1 – Tables sommaires du niveau de confiance (asymptotique) de l'intervalle en fonction de sa largeur (à gauche), resp. de la largeur requise pour atteindre un certain niveau de confiance (à droite). Les arrondis sont faits dans le sens conservatif, càd. que la probabilité de sortir de l'intervalle est surévaluée.

.

augmente. Cela est dû au fait que l'estimateur $\hat{m}_N$ tombe à une distance d'ordre $O(N^{-1/2})$ de la véritable valeur $\mathbb{E}(f)$: en effet, la variance de $\hat{m}_N$ décroit en $O(N^{-1})$, puisque d'après la formule d'additivité des variances pour les variables indépendantes, $\mathrm{Var}(\hat{m}_N) = (N^{-1})^2 \times N\,\mathrm{Var}(f) = N^{-1}\,\mathrm{Var}(f)$. 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 1.23. On appelle *efficacité* d'un calcul de Monte-Carlo l'inverse de la variance de $\hat{m}$ divisée par le cout de calcul. L'efficacité est essentiellement indépendante du nombre de simulations et correspond à l'inverse de la variance de f 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 (1.5), cette efficacité par simulation est simplement l'inverse de $\mathrm{Var}(f)$.

Remarque 1.24. 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 1.25. 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 requis (qui se mesure en s), l'efficacité sera en $^{-2} \cdot s^{-1}$.

1.3 Intérêt et limites de la méthode de Monte-Carlo

1.3.1 Robustesse

Remarque 1.26. On voit ainsi que l'erreur commise par la méthode de Monte-Carlo décroit comme $N^{-1/2}$ (sous réserve que f soit 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 1.1.3, cela signifie que grâce à la méthode de Monte-Carlo on pourra calculer (presque) n'importe quelle intégrale avec une précision en $O(N^{-1/2})$ quand N est le nombre de calculs effectués. 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 $O(N^{-2/d})$ si la fonction à intégrer est lisse, et en $O(N^{-1/d})$ seulement si la fonction n'est que lipschitzienne... On en conclut que 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.

1.3.2 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. [3]

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. Supposons ainsi qu'on dispose de 100 unités de calculs pour effectuer 1 000 000 de simulations. Si chaque unité de calcul effectue indépendamment [4] 10 000 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 1.27. Noter que le type de parallélisation concernant la méthode de Monte-Carlo est particulièrement commode, car il n'y a besoin d'essentiellement *aucune* coordination entre les unités de calcul : ce n'est qu'à la fin que les cerveaux mettront

3. Attention : un calcul parallélisable peut 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.

4. Attention, 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.

en commun leur différents résultats partiels ! D'autre types de parallélisation ne présentent pas cet avantage. Par exemple, si on souhaite modéliser l'évolution des vagues à la surface de la mer, on peut découper la mer en petites parcelles et attribuer à chauqe unité de calcul la simulation de l'évolution d'une parcelle (car deux parcelles éloignées n'interfèrent pas sur de courtes échelles de temps), mais il faudra régulièrement faire communiquer les différentes unités de calcul entre elles pour tenir compte de l'évolution des conditions aux bords due à l'interaction avec les parcelles voisines.

Remarque 1.28. Il existe un type particulier de parallélisation appelé *vectorisation*, où la parallélisation de fait au sein même d'une unité intégrée (par exemple un carte graphique), sous réserve que tous les calculs à faire séparément soient les mêmes (mais avec des données différentes bien sûr) et ne soient pas trop compliqués. Toutefois, nous n'étudierons pas cette méthode ici, car elle demande de tenir compte des spécificités de l'unité de parallélisation (avec codage dans un langage de programmation spécifique ou en assembleur).

À noter que, bien que des logiciels comme *Scilab* et MATLAB soient « spécialisés dans le traitement des calculs vectoriels », ils n'utilisent pas la vectorisation (tout au plus le pipeline), sauf ajout d'un module complémentaire spécifique.

1.3.3 Raffinement

Si un premier calcul de Monte-Carlo a été effectué (disons avec 200 000 simuations) et que sa précision s'avère insuffisante (disons qu'il aurait fallu 500 000 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 première simulations, il n'y aura plus que 300 000 simulations à faire indépendamment pour arriver au total désiré de 500 000.

En outre, il est particulièrement remarquable qu'on n'a pas réellement besoin de connaitre le résultat détaillé des 200 000 premières simulations, mais seulement d'un résultat intermédiaire très concis, à savoir le nombre de simulations effectuées, le total des résultats et le total des carrés ! Pour cette raison, il ne faut jamais hésiter à stocker ces résultats d'une méthode de Monte-Carlo, quand bien même on pense avoir peu de chances de souhaiter s'en resservir.

Remarque 1.29. Précisons toutefois qu'un élément qui permet cette forme de raffinement sans repartir de zéro est que les calculs en virgule flottant de notre ordinateur seront normalement de précision largement supérieure à celle que nous pensons obtenir au final avec notre méthode de Monte-Carlo. Si on souhaitait raffiner un premier calcul pour aller au-delà de la précision de la machine, alors cela ne marcherait pas, car les résultats partiels étant obtenus avec une précision insuffisante, on ne pourrait pas lesintégrer au calcul du résultat final ! De même, on ne peut pas raffiner un calcul de série fait pour obtenir 15 décimales de e en rajoutant des termes jusqu'à une précision de 40 décimales, si le premier calcul n'a pas été effectué avec une précision de 40 décimales (quand bien même on sait que les décimales calculées par le premier calcul au-delà de la 15^{e} ne sont pas celles de e, on a besoin de les connaitre pour la suite !).

Cette remarque vaut également pour la parallélisation.

1.3.4 Précision

À l'inverse, la méthode de Monte-Carlo présente également des handicaps non négligeables. Par exemple, si nous souhaitons utiliser la méthode de Monte-Carlo pour un calcul de π (ce qui est tout-à-fait possible), la précision du calcul pour un cout de calcul T sera en $T^{-1/2}$, soit $O(\log T)$ décimales exactes. Alors que les méthodes non probabilistes permettent d'obtenir assez facilement un nombre de décimales exactes en $O(T^{1/2})$ voire en $O(T^{1-\varepsilon})$... Autant dire que si nous nous étions limités à la méthode de Monte-Carlo, même au prix d'efforts démesurés nous ne connaitrions même pas une vingtaine de décimales exactes, à comparer aux milliards que nous avons aujourd'hui...

Cela dit, pour l'ingénieur il est très (très !) rare d'avoir besoin de descendre à des niveaux de précisions aussi exigeants ; cet écueil ne devrait donc guère vous réfréner d'employer la méthode de Monte-Carlo pour les applications industrielles. Il n'en reste pas moins que parfois la relative lenteur de la méthode de Monte-Carlo fait que d'autres méthodes de calcul concurrentes s'avèrent beaucoup plus appropriées. [5]

1.3.5 Génération aléatoire

Un autre point à garder à l'esprit est que les nombres aléatoires générés par votre machine ne sont en général que *pseudo*-aléatoires... Mais à nouveau le problème ne se poserait que pour des calculs très poussés

1.3.6 Le caractère aléatoire, un problème ?

Le premier reproche qu'on a envie de faire à la méthode de Monte-Carlo est qu'elle ne permet pas d'avoir une certitude absolue sur la validité d'un résultat. Cette limitation ne doit pas vous effrayer, à moins que vous soyez mathématiciens théoriciens ! En effet, du point de vue *pratique*, une probabilité inférieure à 10^{-10} est totalement négligeable (c'est à peu près la probabilité que vous mouriez pendant cette séance de cours d'une rupture d'un anévrisme non détecté, et bien moins que la probabilité que votre patron vienne d'être arrêté pour meurtre sans que vous le sachiez !).

On serait tenté de rétroquer que, d'après ce que j'ai expliqué à la remarque 1.20, on ne peut pas toujours obtenir des estimations à la probabilité de 10^{-10}. Mais cela n'a guère d'importance, dans la mesure où on peut combiner plusierus calculs indépendants pour obtenir une précision extraordinaire. Si ainsi vous lancez 34 simulations indépendantes qui ont chacune au moins 50 % de chances d'être valides, il n'y aura qu'une chance sur 10^{10} que le résultat ne soit dans aucun des intervalles de confiance trouvés ! Cela montre qu'on peut obtenir facilement des niveau de confiance ultra-précis en faisant la *réunion* de plusieurs intervalles de confiance pour des calculs *indépendants*.

5. D'ailleurs, n'oubliez pas que quand vous cherchez à estimer une valeur d'intérêt dans le cadre d'un modèle (par exemple le juste prix d'une option dans le cadre des marchés financiers, ce qui est une application classique de la méthode de Monte-Carlo), le modèle *lui-même* n'est généralement pas exact ou pas parfaitement calibré ! Il est donc absurde est dangereux de donner une précision supérieure à quelques décimales dans ce cas-là, en vertu du fameux adage *garbage in, garbage out*.

1.3.7 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 auquel il convient d'être prudent.

Il existe des moyens d'obtenir des intervalles de confiance non asymptotiques, mais pour cela il faut être capable d'obtenir un contrôle *explicite* sur la loi de probabilité en fonction d'une loi exponentielle. C'est en particulier le cas quand nous savons explicitement que la fonction que nous simulons est *bornée*, et certains des exercices associés à ce cours donnent les formules. Nous n'aborderons pas plus en détail cette question ici, bien qu'elle tout-à-fait intéressante.

1.4 Vers la réduction de la variance

Nous avons vu dans la § 1.2.2 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'efficacité 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 !).

Chapitre 2

Techniques de réduction de la variance

2.1 Généralités

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

Comme nous l'avons dit à la fin du chapitre précédent, les *techniques de réduction de la variance* se proposent, à l'aide de certaines manipulations algériques, 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 2.1. 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.

2.1.2 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 !

2.1.3 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.

2.1.4 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.

2.2 Échantillonnage préférentiel

2.2.1 Principe de l'échantillonnage préférentiel

Définition 2.2. Supposons qu'on cherche à évaluer $\mathbb{E}_P(f)$ par la méthode de Monte-Carlo, $\mathbb{E}_P$ désignant l'espérance associée à une certaine loi P sur l'espace mesurable Ω. Supposons qu'on dispose d'une autre loi Q sur le même espace Ω qu'on sait simuler, et que P a une densité $\big(dP/dQ\big)(\omega)$ par rapport à Q qu'on connait exactement. Alors la *technique d'échantillonnage préférentiel* consiste à observer que $\mathbb{E}_P(f) = \mathbb{E}_Q(dP/dQ \times f)$ et à estimer $\mathbb{E}_P(f)$ par

$$\hat{m}^{\text{pref}} := N^{-1} \sum_{i=1}^{N} \frac{dP}{dQ}(\tilde{\omega}_i) f(\tilde{\omega}_i),$$

où les $\tilde{\omega}_i$ sont simulés i.i.d. *selon la loi* Q.

Comme nous allons le voir dans la suite de la section, l'échantillonnage préférentiel est pertinent lorsque la loi Q donne plus de poids aux ω qui contribuent le plus à l'espérance de f. Mais voyons tout de suite deux exemples pour illustrer ce concept essentiel :

Exemple 2.3 (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 la conclusion que trois facteurs sont impliqués dans un tel risque, facteurs qu'elle aprelle respectivement X, Y et Z, et que ces trois vecteurs sont distribués selon la loi normale

$$(X, Y, Z) \sim (-3, -4, -5) + \mathcal{N} \begin{pmatrix} 1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 1 & 0{,}4 \\ 0{,}3 & 0{,}4 & 1 \end{pmatrix}.$$

L'accident se produit quand les trois facteurs X, Y et Z deviennent *tous les trois* positifs ; il faut donc évaluer $\mathbb{P}((X,Y,Z) \in \mathbb{R}_+^3) =: p$. La norme de certification impose que p soit inférieur à un milliardième.

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 dans l'annexe **??**. Avec un-million simulations, elle obtient le résultat suivant :

```
>> risque_spl(1000000)
Probabilité d'accident : 0 ± 0
```

À première vue, il n'y a donc strictement aucun risque. Mais en fait, réfléchit l'ingénieure, de résultat est dégénéré : il signifie simplement que toutes les simulations ont été négatives ! Mais si le risque véritable est d'un cent-millionnième, ce n'est pas en faisant seulement un-million simulations que je vais pouvoir voir quelque chose... Le problème, c'est que les capacités de calcul de son ordinateur ne permettent guère de dépasser un million de simulations... D'où l'idée d'utiliser une technique d'échantillonnage préférentiel.

Appelons $f = \mathbf{1}_{\{\text{accident}\}}$ l'indicatrice dont l'ingénieure veut évaluer l'espérance et P la loi du phénomène. f étant (X,Y,Z)-mesurable, on peut voir P comme une loi sur les valeurs de (X,Y,Z). Nous abrégeons (2.3) sous la forme « $(X,Y,Z) \sim M + \mathcal{N}(\Sigma)$ ». Quelle loi d'échantillonnage Q allons-nous considérer ? On veut que Q donne un poids assez important aux triplets (X,Y,Z) de $\mathbb{R}_+^3$, sans trop en sous-échantillonner aucun. Une idée est de prendre Q telle que, sous Q, $(X,Y,Z) \sim \mathcal{N}(\Sigma)$: sous cette nouvelle loi les valeurs de (X,Y,Z) seront proches de $(0,0,0)$, donc elles couvriront bien mieux la zone qui nous concerne que sous P. Peut-on calculer dP/dQ ? Oui, assez facilement. En effet, nous connaissons la formule de la densité pour les variables gaussiennes, qui nous dit que sous P,

$$dP\big((X,Y,Z) = (x,y,z) =: \vec{v}\big) = \frac{(2\pi)^{-3/2}}{\det C} \exp\big(-\tfrac{1}{2}(\vec{v}-M)C(\vec{v}-M)^{\mathsf{T}}\big)\, dxdydz,$$

tandis que sous Q, il faut remplacer M par 0 :

$$dQ\big((X,Y,Z) = (x,y,z) =: \vec{v}\big) = \frac{(2\pi)^{-3/2}}{\det C} \exp\big(-\tfrac{1}{2}\vec{v}C\vec{v}^{\mathsf{T}}\big)\, dxdydz\,;$$

de sorte qu'en prenant le quotient, on trouve

$$\begin{aligned}\frac{dP}{dQ}\big((X,Y,Z) = (x,y,z) =: \vec{v}\big) &= \exp\big(-\tfrac{1}{2}(\vec{v}-M)C^{-1}(\vec{v}-M)^{\mathsf{T}} + \tfrac{1}{2}\vec{v}C^{-1}\vec{v}^{\mathsf{T}}\big)\\ &= \exp\big(\tfrac{1}{2}\vec{v}C^{-1}M^{\mathsf{T}} + \tfrac{1}{2}\vec{v}^{\mathsf{T}}C^{-1}M - \tfrac{1}{2}MC^{-1}M^{\mathsf{T}}\big) = e^{-MC^{-1}M^{\mathsf{T}}/2}\exp\big(\vec{v}C^{-1}M^{\mathsf{T}}\big).\end{aligned}$$

Cela conduit au code de l'annexe **??**, dont l'exécution donne :

```
>> risque_pref(200000)
Probabilité d'accident : 5.1108e-10 ± 1.4781e-11
```

Cette fois-ci le résultat est tout-à-fait exploitable, comme le confirme la finesse de l'intervalle de confiance en valeur relative. On trouve que les accidents arriveront avec une probabilité d'environ $\frac{1}{2}$ milliardième, ce qui permet donc de certifier la centrale. (Ici le temps de calcul par simulation a été environ 2 fois plus lent dans la version préférentielle à cause de la complexité supplémentaire due au calcul et à la prise en compte de la densité, mais le gain de variance est tellement important qu'on a pu pallier cet inconvénient en prenant 5 fois moins de simulations, tout en obtenant encore un résultat incomparablement meilleur).

Exemple 2.4. 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 P et Q portent sur un espace discret, $\big(dP/dQ\big)(\omega)$ est simplement $P(\{\omega\})/Q(\{\omega\})$. Nous considérons à nouveau la problème du championnat de basket, mais cette fois-ci nous supposons que l'équipe de Nancy est la #16 et a donc très peu de chances de remporter le championnat. Dans ces conditions, l'échantillonnage selon la loi de probabilité "naturelle" (notée P) n'est sans doute pas le plus adapté, car on tombe très rarement sur des victoires de Nancy... Nous proposons alors l'échantillonnage préférentiel suivant, selon une loi de probabilité que nous noterons Q. Sous la probabilité Q, les matchs n'impliquant pas Nancy se déroulent normalement, mais les matchs impliquant Nancy ont leur résultat tiré au sort de façon *équilibrée* entre Nancy et son adversaire. Ainsi, sous la probabilité P, Nancy avait $i^{1/2}$ / $(4 + i^{1/2})$ chances de gagner et 4 / $(4 + i^{1/2})$ chances de perdre, tandis que sous la probabilité Q elle a $1/2$ chances de gagner et $1/2$ chances de perdre : relativement au seul match entre Nancy et l'équipe $\#i$, la densité de P par rapport à Q est donc de $2i^{1/2}$ / $(4 + i^{1/2})$ en cas de victoire et de 8 / $(4 + i^{1/2})$ en cas de défaite. En fait, ce qui nous intéresse est la densité relative entre P et Q sur le tirage global, mais comme dans les deux cas les différents matchs sont indépendants, cette densité sera le produit des densités relativement aux différents matchs (cf. Annexe **??**). Toujours pour 70 000 simulations, j'obtiens le résultat suivant :

```
>> championnat_pref
Probabilité de victoire évaluée pour l'équipe de Nancy :
    0.0035 ± 1.9024e-04
Durée du calcul en s :
    1.6977
```

à comparer à ce qu'on aurait obtenu sans échantillonnage préférentiel (en décommentant ou modifiant les lignes idoines du code de la § A.1.1) :

```
>> championnat
Probabilité de victoire évaluée pour l'équipe de Nancy :
    0.0036 ± 4.4369e-04
Durée du calcul en s :
    1.1254
```

L'efficacité par simulation est donc nettement meilleure ! (d'un facteur supérieur à 5). Certes, le fait d'avoir eu à calculer la densité nous a pris un peu plus de temps, mais au final le gain d'efficacité reste largement bénéfique.

2.2.2 Recherche d'une bonne loi d'échantillonnage préférentiel

La technique d'échantillonnage préférentiel nous laisse le choix de la probabilité d'échantillonnage Q — sous réserve évidemment que nous sachions simuler Q et calculer dQ/dP. Puisque la technique d'échantillonnage repose sur l'espoir qu'échantillonner selon Q sera plus efficace qu'échantillonner selon P, c'est qu'on s'attend à ce que certaines Q soient meilleures que d'autres... Mais lesquelles ? Y a-t-il des heuristiques pour trouver quelles sont les meilleures lois d'échantillonnage 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 2.5. *L'efficacité par simulation de la technique d'échantillonnage préférentiel est égale à l'inverse de* $\big(\mathbb{E}_P(dP/dQ \times f^2) - \mathbb{E}_P(f)^2\big)$.

Démonstration. L'efficacité par simulation est par définition l'inverse de la variance $\mathrm{Var}_Q(dP/dQ \times f)$, qui est égale à $\mathbb{E}_Q\big((dP/dQ)^2 \times f^2\big) - \mathbb{E}_Q(dP/dQ \times f)^2 = \mathbb{E}_P(dP/dQ \times f^2) - \mathbb{E}_P(f)^2$ d'après la propriété caractéristique de la densité dP/dQ. □

Théorème 2.6. *L'efficacité par simulation par la méthode d'échantillonnage préférentiel est maximale quand la densité* dQ/dP de Q par rapport à P *est proportionnelle à* $|f|$.

Démonstration. Notons $\rho := dQ/dP = (dP/dQ)^{-1}$ (on fera comme si les mesures P et Q étaient équivalentes, ce qui justifie cette expression). D'après la proposition 2.5, notre objectif alors est de minimiser $\mathbb{E}_P(\rho^{-1} f^2)$, et ce sous la contrainte que Q soit une mesure de probabilité, càd. 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 ρ 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 λ à fixer, on cherche un ρ^* vérifiant $\rho^* \geqslant 0$ et $\mathbb{E}_P(\rho^*) = 1$ tel que ρ^* 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 $\big(DF(\rho)\big) \cdot h = \mathbb{E}_P\big((-\rho^{-2} f^2 + \lambda) h\big)$, de sorte qu'un point critique ρ^* doit vérifier $-(\rho^*)^{-2} f^2 + \lambda \equiv 0$, d'où $\rho^* \propto |f|$. □

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

Remarque 2.7. 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 2.5, où ce qui est susceptible de faire exploser $\mathbb{E}_P(dP/dQ \times f^2)$ est que dP/dQ soit très grande (autrement dit que dQ/dP 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 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 2.8. Soient P^* et P_* deux mesures de probabilité sur un espace Ω se décomposant comme une union disjointe $\Omega_0 \cup \Omega_1$. On suppose que P^* (qui sera la mesure *sur*-échantillonnée) donne la masse ε^* à Ω_1 et $(1-\varepsilon^*)$ à Ω_0, et que P_* (qui sera la mesure *sous*-échantillonnée) leur donne quant à elle respectivement les masses ε_* et $(1-\varepsilon)_*$; et on suppose que P_* et P^* ont des densités l'une par rapport à l'autre qui sont constantes sur chaque Ω_i. On prend ici $\varepsilon_* \ll \varepsilon^* \ll 1$, de sorte que les mesures P^* et P_* sont pratiquement identiques sur Ω_0 (la densité relative dP^*/dP_* valant $(1-\varepsilon^*)/(1-\varepsilon_*) \simeq 1$), mais très différentes sur Ω_1.

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

$$\mathbb{E}_{P_*}(dP_*/dP^*) - 1 = (1-\varepsilon_*) \times \frac{1-\varepsilon_*}{1-\varepsilon^*} + \varepsilon_* \times \frac{\varepsilon_*}{\varepsilon^*} - 1 \simeq \varepsilon^*.$$

À l'inverse, en échantillonnant avec P_* là où P^* aurait été plus adaptée, on tombe sur une efficacité par simulation inverse de

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

ce qui est beaucoup plus grand (et donc beaucoup moins bon) que ε^* !

Remarque 2.9. 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.

2.3 Conditionnement

Définition 2.10. Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo, f étant une v.a. L^2 définie sur l'espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$, et supposons qu'il

y ait une sous-tribu $\mathcal{B}$ de $\mathcal{A}$ telle qu'on sache calculer $\mathbb{E}(f|\mathcal{B}) =: f^{\mathcal{B}}$. Alors la *méthode de conditionnement* consiste à observer que $\mathbb{E}(f) = \mathbb{E}(f^{\mathcal{B}})$ et à estimer $\mathbb{E}(f)$ par l'estimateur

$$\hat{m}^{\text{cond}} := N^{-1} \sum_{i=1}^{N} f^{\mathcal{B}}(\omega_i).$$

Remarque 2.11. On peut résumer la philosophie du conditionnement ainsi : ne pas utiliser Monte-Carlo quand on a accès à un résultat exact !

Exemple 2.12. Reprenons l'exemple de la loterie que nous avions vu dans la §**??**. De la façon dont nous procédions, l'aléa du résultat intervenait en douze étapes : tirage du premier numéro sorti, puis du deuxième, etc. Conditionner, c'est en fait n'utiliser qu'une partie de l'information sur cet aléa ; en l'occurrence, regardons ce qui se passe quand on conditionne par rapport aux onze premiers tirages ; c'est-à-dire que plutôt que d'attendre le tirage du tout dernier numéro, la joueuse calcule son espérance de gain sachant les onze premiers tirés. Calculer cette espérance est facile, car il suffit de faire la disjonction de treize cas dont la probabilité de chacun est connue exactement : que le dernier numéro tiré soit le pari à 12 points de la joueuse (probabilité 1/85 si ce numéro n'est pas encore sorti, 0 sinon), que le dernier numéro soit le pari à 11 points (idem), ..., que le dernier numéro soit le pari à 1 point, et que ce ne soit aucun des numéros pariés (probabilité de 1 − nombre_de_paris_sortis/85). Évidemment, calculer cela va prendre plus de temps que juste tirer la dernière boule et calculer le score, mais dans la mesure où on a totalement tué l'aléa dû à la dernière boule, on peut quand même espérer y gagner !

En pratique, j'ai obtenu les résultats suivants, avec le code de l'annexe A.2.3 :

Théorème 2.13. *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 $f^{\mathcal{B}}$.

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

Remarque 2.14. Non seulement la variance par simulation est réduite quand on utilise le conditionnement, mais en général le cout de calcul par simulation va *aussi* diminuer ! En effet, il y aura besoin de pousser la simulation moins loin pour savoir où on tombe dans la tribu $\mathcal{B}$ que pour savoir où on tombe dans la tribu $\mathcal{A}$, vu que la tribu $\mathcal{B}$ est plus grossière. La seule chose qui pourrait éventuellement prendre plus de temps est l'évaluation de f, mais elle-ci représente quasiment toujours une fraction négligeable du cout de calcul. En conclusion, il ne faut jamais hésiter à implémenter une technique de réduction de variance par conditionnement quand on en voit la possibilité.

2.4 Stratification

2.4.1 Stratification avec échantillonnage uniforme

Théorème 2.15. *Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo, et que l'espace Ω sur lequel est définie la probabilité $\mathbb{P}$ est partitionné en différentes « strates » $A_1, \dots, A_k$ 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

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

a une meilleure efficacité (par simulation) que l'estimateur "simple" (1.5).

Démonstration. Notons $\mathbb{P}_k$ la loi conditionnelle de $\mathbb{P}$ sous A_k (càd. $\mathbb{P}_k(B) := p_k^{-1}\,\mathbb{P}(B \cap A_k)$), 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 (2.15) comme $\hat{m}_{\text{strat}} = \sum_k p_k \hat{m}_k$, où $\hat{m}_k$ est l'estimateur de Monte-Carlo "ordinaire" de m_k, réalisé avec un nombre 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 $\hat{m}_k$ a une variance égale à $n_k^{-1} \operatorname{Var}_{\mathbb{P}_k}(f)$, et les différents $\hat{m}_k$ sont indépendants, donc la variance globale de $\hat{m}^{\text{strat}}$ vaut

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

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

$$\begin{aligned}\operatorname{Var}(f) = \mathbb{E}(f^2) - \mathbb{E}(f)^2 &= \sum_k p_k\,\mathbb{E}_k(f^2) - \Big(\sum_k p_k\,\mathbb{E}(f_k)\Big)^2 \\ &= \sum_k p_k \operatorname{Var}_{\mathbb{P}_k}(f) + \Big[\sum_k p_k\,\mathbb{E}_k(f)^2 - \Big(\sum_k p_k\,\mathbb{E}(f_k)\Big)^2\Big]. \quad (2.1)\end{aligned}$$

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 $\sum_k p_k = 1$). □

Exemple 2.16 (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 1 024 électeurs, notre institut de sondage a obtenu les résultats suivants :

Réponse	Nombre	Pourcentage
Pour	455	44,43 %
Blanc	135	13,19 %
Contre	434	42,38 %

D'après ces résultats, la méthode de Monte-Carlo lui permettra d'annoncer comme estimateur une victoire du « oui » par une avance de 2,05 % des inscrits, avec une marge d'erreur à 2 sigmas sur cette avance de ±2,91% — ce qui n'est pas assez pour conclure de façon catégorique...

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 :

Département	Sondés	Pour	(% pour)	Blanc	(% blanc)	Contre	(% contre)
54	309	96	31,07	69	23,33	144	45,60
55	100	39	39,00	9	9,00	52	52,00
57	430	304	70,70	39	9,07	87	20,23
88	185	16	8,65	18	9,73	151	81,63
Total	1024	455	44,43	135	13,19	434	42,38

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 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 :

Département	Résultat local	Proportion	Contribution
54	-14.53 ± 9.87	0,3	4.36 ± 2.96
55	-13.00 ± 18.90	0,1	...
57	...	0,4	
88		0,2	
Total	$+2.05 \pm 2.91$	1	$+0.53 \pm 1.26$

Pour obtenir la dernière case de la ligne « Total », on a ajouté les estimateurs. Concernant la marge d'erreur, étant donné que les différentes estimations sont indépendantes, il faut ajouter les *variances* et non pas simplement les intervalles : puisque

la largeur de l'intervalle de confiance est proportionnelle à l'écart-type, on obtient une largeur pour l'intervalle global qui est la racine de la somme des carrés des longueurs, soit 1.26, nettement plus petite que la somme des longueurs qui serait 4.55.

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, il pronostique un résultat nettement plus serré.
– La précision de l'estimateur stratifié est meilleure que celle de l'estimateur brut ! L'utilisation intelligente des données sur les départements nous a donc permis d'améliorer notre estimation.

2.5 Variables de contrôle

2.5.1 Version rudimentaire

Définition 2.17. Supposons qu'on cherche à évaluer $\mathbb{E}(f)$ par la méthode de Monte-Carlo, et qu'on disposne d'une variable aléatoire g de classe L^2 sur Ω 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

$$\hat{m}^g := N^{-1} \sum_{i=1}^{N} \big(f(\omega_i) - g(\omega_i)\big) + \mathbb{E}(g).$$

Proposition 2.18. *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 2.19. 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 $\mathcal{N}(1)$; autrement dit, pour $(S_j)_{1\leqslant j\leqslant 60}$ des v.a. i.i.d. $\mathcal{N}(1)$, on considère la suite finie $(X_i)_{0\leqslant i\leqslant 60}$ définie par $X_j = \sum_{j=1}^{i} S_j$. 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 ω, on pose $X_*(\omega) := \max_{0\leqslant i\leqslant 60} X_i(\omega)$. 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, $Y \sim \mathcal{N}(60)$, donc, notant $\sigma := \sqrt{60}$,

$$\begin{aligned}\mathbb{E}(Y) = \int_{-\infty}^{\infty} \max\{x,0\} \frac{e^{-x^2/2\sigma^2}}{\sqrt{2\pi}\sigma}\, dx &= \frac{1}{\sqrt{2\pi}\sigma} \int_0^{\infty} x e^{-x^2/2\sigma^2} dx \\ &= \frac{1}{\sqrt{2\pi}\sigma} \big[-\sigma^2 e^{-x^2/2\sigma^2}\big]_0^{\infty} = \frac{\sigma}{\sqrt{2\pi}} = 3{,}090\ 193... \quad (2.2)\end{aligned}$$

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

L'annexe A.5 montre comment implémenter l'utilisation de cette variable de contrôle. Sans variable de contrôle, on obtient le résultat suivant :

```
>> tic; mathfi_naif(100000); toc;
Estimation de E(X_*) : 5.62716 ± 0.02893
Elapsed time is 1.086699 seconds.
```

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

```
>> tic; mathfi_ctrl0(100000); toc;
Estimation de E(X_*) : 5.61888 ± 0.01427
Elapsed time is 0.998897 seconds.
```

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.

2.5.2 Version paramétrée

Si on dispose d'une variable de contrôle g, on peut aussi se servir de λg 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 λ.

Théorème 2.20. *La valeur de λ qui minimise $\mathrm{Var}(f - \lambda g)$ est*

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

Notant $r := \mathrm{Cov}(f,g) \,/\, \mathrm{Var}(f)^{1/2}\,\mathrm{Var}(g)^{1/2}$ 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 λ dont la minimisation est élémentaire : le minimum de $\mathrm{Var}(f - \lambda g)$ est atteint en λ^* 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 2.21. 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 de l'annexe **??**. L'exécution donne :

```
>> tic; mathfi_ctrl1(100000); toc;
Estimation de E(X_*) : 5.61753 ± 0.01404
Elapsed time is 1.029657 seconds.
```

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 λ optimal vaut à peu près 0,9, de sort que la version simple (correspondant à $\lambda = 1$) était déjà pratiquement optimale.

2.5.3 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 (2.17) avec $\lambda_1 g_1 + \dots + \lambda_K g_K$, $(\lambda_1, \dots, \lambda_K)$ étant un jeu de K paramètres à optimiser.

Théorème 2.22. *La valeur de* $(\lambda_1, \dots, \lambda_K) =: \vec{\lambda}$ *qui minimise* $\mathrm{Var}(f - \lambda_1 g_1 - \dots - \lambda_K g_K)$ *est donnée par*

$$\vec{\lambda}^* = R_{fg} \Sigma_g^{-1},$$

où R_{fg} *est le vecteur* $\big(\mathrm{Cov}(f, g_1), \dots, \mathrm{Cov}(f, g_K)\big)$ *et* Σ_Y *est la matrice de covariance (supposée non dégénérée) de* $(g_1, \dots, g_K)$.

Démonstration. Par bilinéarité,

$$\mathrm{Var}(f - \lambda_1 g_1 - \dots - \lambda_K g_K) = \mathrm{Var}(f) - 2\vec{\lambda} R_{fg}^{\mathsf{T}} + \vec{\lambda} \Sigma_g \vec{\lambda}^{\mathsf{T}},$$

où nous rappelons que Σ_g, en tant que matrice de covariance, est symétrique et positive (définie). Ce polynôme de degré 2 en λ se réduit en

$$\mathrm{Var}(f) + \left\| \vec{\lambda} \Sigma_g^{1/2} - R_{fg} \Sigma^{-1/2} \right\|^2 - R_{fg} \Sigma_g^{-1} R_{fg}^{\mathrm{T}},$$

où $\|v\|^2$ désigne la norme vv^{T} d'un vecteur v de $L^2(\mathbb{R}^K)$. Le seul terme de (2.5.3) dépendant de $\vec{\lambda}$ étant alors le carré de la norme de $\vec{\lambda}\Sigma_g^{1/2} - R_{fg}\Sigma^{-1/2}$, l'expression est minimale quand ce vecteur s'annule, ce qui donne bien $\vec{\lambda}^* = R_{fg}\Sigma_g^{-1}$. □

Exemple 2.23. Reprenons encore une fois l'exemple des mathématiques financières. Je prends trois variables de contrôle : $X_{20} \wedge 0 := Y_1$, $X_{40} \wedge 0 := Y_2$ et $X_{60} \wedge 0 := Y_3$, qui ont pour espérances respectives $\sqrt{20\,/\,2\pi}$, $\sqrt{40\,/\,2\pi}$ et $\sqrt{60\,/\,2\pi}$. Cela conduit au programme de l'annexe **??**, dans laquelle nous avons utilisé le formalisme vectoriel de MATLAB pour traiter Y_1, Y_2 et Y_3 avec une seule ligne de code à chaque fois. L'exécution donne :

```
>> tic;mathfi_ctrl3(100000);toc;
Estimation de E(X_*) : 5.62414 ± 0.01012
Elapsed time is 1.827719 seconds.
```

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

2.6 Variables antithétiques

Théorème 2.24. *Supposons que nous cherchons à évaluer* $\mathbb{E}(f)$ *par la méthode de Monte-Carlo. Notant* Ω *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* $\mathbb{P}$ *(càd. que la mesure-image de* $\mathbb{P}$ *par* γ *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

$$\hat{m}^{\text{anti}} := (2N)^{-1} \sum_{i=1}^{N} \big(f(\omega_i) + f(\gamma(\omega_i)) \big)$$

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 $\frac{1}{4}\big(\mathrm{Var}(f)+\mathrm{Var}(f\circ\gamma)+\mathrm{Cov}(f,f\circ\gamma)\big) = \frac{1}{2}\mathrm{Var}(f) + \frac{1}{4}\mathrm{Cov}(f,f\circ\gamma) \leqslant \frac{1}{2}\mathrm{Var}(f)$ 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 2.25. 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 dans l'annexe A.5.5, qui donne à l'exécution :

```
>> tic;mathfi_anti(100000);toc;
Estimation de E(X_*) : 5.62541 ± 0.01627
Elapsed time is 0.615886 seconds.
```

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.

Annexe A

Annexe : Codes MATLAB

A.1 Le championnat de basket

A.1.1 Méthode de Monte-Carlo basique : `championnat.m`

```
function championnat()

% Décommenter la ligne suivante pour connaitre le temps de calcul.
%tic;

% Le nombre de simulations.
N = 70000;
% Nombre d'équipes impliquées dans le championnat.
nequipes = 16;
% Numéro de l'équipe de Nancy. Il suffit de modifier cette ligne pour
% traiter le cas où Nancy est l'équipe #16.
Nancy = 3;

% bilan est le nombre de simulations où l'équipe de Nancy a été classé
% championne.
bilan = 0;
% La boucle de Monte-Carlo.
for i = 0 : N-1
    % victoires(a) est le nombre de matches remportés par l'équipe #a.
    victoires = zeros(1,nequipes);
    for a = 1 : nequipes
        % On fait jouer chaque équipe contre l'autre une fois.
        for b = a+1 : nequipes
            % Probabilité de défaite de a.
            if rand < 1/(1+(b/a)^(1/2))
                victoires(b) = victoires(b) + 1;
            else
                victoires(a) = victoires(a) + 1;
            end
        end
    end
```

```
    % Une méthode pour départager uniformément les ex-æquo est d'ajouter
    % une partie fractionnaire aléatoire à chaque score.
    victoires = victoires + rand(1,nequipes);
    [~,championne] = max(victoires);
    if championne == Nancy
        bilan = bilan + 1;
    end
end
probabilite = bilan / N;
disp ('Probabilité de victoire évaluée pour l''équipe de Nancy :');
disp (probabilite);

% Décommenter les lignes suivantes pour avoir l'intervalle de confiance (à
% 5 p.c.). Noter que dans ce cas particulier de la méthode de rejet où on
% s'intéresse à une variable de Bernoulli, l'écart-type est s'exprime
% directement en fonction de la moyenne.
%ecarttype = sqrt(probabilite*(1-probabilite));
%disp ('±'); disp (1.96 * ecarttype/sqrt(N));

% Décommenter les lignes suivantes pour connaitre le temps de calcul.
%t = toc;
%disp ('Durée du calcul en s :');
%disp (t);
end
```

A.1.2 Avec échantillonnage préférentiel : `championnat_pref.m`

```
function championnat_pref()
% Variante de la fonction "championnat" utilisant l'échantillonnage
% préférentiel pour évaluer la probabilité de victoire de l'équipe la moins
% forte. Je ne commente que les différences

tic;
N = 70000;
nequipes = 16;
% Cette fois-ci c'est l'équipe #16 qui nous intéresse.
Nancy = 16;

% bilan est toujours le nombre de simulations où l'équipe qui nous
% intéresse a été classée championne, mais cette fois-ci pondéré par la
% densité de la vraie loi par rapport à la loi d'échantillonnage.
bilan = 0;
% Pour donner un intervalle de confiance et évaluer l'efficacité, il va
% aussi nous falloir connaitre la somme des carrés des quantités simulées.
sommecarres = 0;

for i = 0 : N-1
    % Puisque nous n'échantillonnons pas sous la véritable loi, il nous
```

```
% faut aussi calculer la densité relative de la véritable loi. Celle-ci
% va s'obtenir comme un produit de densités partielles indépendantes
% entre elles, aussi nous l'initialisons à 1.
densite = 1;
victoires = zeros(1,nequipes);
% On commence par simuler les résultats de l'équipe qui nous intéresse.
% Ceux-ci étant tirés selon la loi uniforme, qui n'est pas la vraie
% loi, il nous faudra en tenir compte pour la densité.
for b = 1 : nequipes
    % Il faut penser à ne pas simuler de match de Nancy contre
    % elle-même...
    if b ~= Nancy
        % Notre simulation donne 50 % de probabilité de victoire à
        % Nancy pour chaque match
        if rand < 1/2
            % En cas de victoire...
            victoires(Nancy) = victoires(Nancy) + 1;
            % On modifie la densité pour tenir compte de
            % l'échantillonnage préférentiel.
            densite = densite * 2/(1+(Nancy/b)^(1/2));
        else
            % En cas de défaite...
            victoires(b) = victoires(b) + 1;
            densite = densite * 2/(1+(b/Nancy)^(1/2));
        end
    end
end

% Maintenant on simule les autres matchs, cette fois-ci directement
% sous la vraie probabilité
for a = 1 : nequipes
    for b = a+1 : nequipes
        % N'oublions pas de ne pas refaire les simulations impliquant
        % Nancy...
        if (a ~= Nancy & b ~= Nancy)
        if rand < 1/(1+(b/a)^(1/2))
            victoires(b) = victoires(b) + 1;
        else
            victoires(a) = victoires(a) + 1;
        end
        end
    end
end
victoires = victoires + rand(1,nequipes);
[~,championne] = max(victoires);
    % Maintenant, il va falloir tenir compte de la pondération dans le
    % bilan !
    contribution = (championne == Nancy) * densite;
```

```
        bilan = bilan + contribution;
        sommecarres = sommecarres + contribution * contribution;
end

probabilite = bilan / N;
ecarttype = sqrt (sommecarres/N - probabilite*probabilite);
disp ('Probabilité de victoire évaluée pour l''équipe de Nancy :');
disp (probabilite);
disp ('±');
disp (1.96*ecarttype/sqrt(N));

t = toc;
disp ('Durée du calcul en s :')
disp (t);
end
```

A.2 La loterie

A.2.1 Méthode de Monte-Carlo basique : `loterie.m`

```
function loterie

% Le nombre de simulations.
N = 700000;
% Nombre de numéros qu'il y a en jeu.
Nnumeros = 36;
% Nombre de boules tirées (et de paris à faire).
Ntirages = 12;
% total est le total des gains au cours des simulations.
total = 0;

% La boucle de Monte-Carlo.
for i = 0 : N-1

    % Comme tous les numéros sont interchangeables, tous les remplissages
    % de la grille sont équivalents ; on va donc supposer que notre joueuse
    % a fait un pari à 1 point sur le no 1, à 2 points sur le numéro 2,
    % ..., et à 12 points sur le numéro 12

    % On tire au sort les numéros qui sortent. "numerostires" est la liste
    % des numéros tirés, qu'on va remplir successivement.
    numerostires = zeros(1,Ntirages);
    % "boules" est la liste des boules présentes dans l'urne.
    boules = 1 : Nnumeros;
    % On maintient aussi le nombre de boules restant dans l'urne, appelé
    % "nombredeboules", ainsi que le nombre de boules déjà tirées, appelé
    % "nombretirees".
    nombredeboules = Nnumeros;
```

```
    nombretirees = 0;

    % On tire successivement 12 boules.
    for j = 1 : Ntirages
        % "laboule" est la position de la boule tirée au sein de la liste des boules de l
        laboule = randi(nombredeboules);
        % On regarde le numéro correspondant et on l'ajoute à la liste des numéros tirés.
        numerostires(nombretirees+1) = boules(laboule);
        % On retire la boule de la liste des boules à tirer.
        boules (laboule) = boules (nombredeboules);
        nombredeboules = nombredeboules - 1;
        nombretirees = nombretirees + 1;
    end

    score = sum (numerostires .* (numerostires <= Ntirages));
    gain = 10 * exp((score - 39)/8) * (score >= 39);
    total = total + gain;

end

moyenne = total / N;
disp ('Gain moyen évalué :');
disp (moyenne);

end
```

A.2.2 Avec les intervalles de confiance : `loterie_IC.m`

```
function loterie_IC

N = 100000;
total = 0;
totalcarres = 0; % On maintient aussi le total des carrés cette fois-ci.

for i = 0 : N-1

    % La partie simulation de la boucle de Monte-Carlo reste inchangée.
    numerostires = zeros(1,12);
    boules = 1 : 48;
    nombredeboules = 48;
    nombretirees = 0;
    for j = 1 : 12
        laboule = randi(nombredeboules);
        numerostires(nombretirees+1) = boules(laboule);
        boules (laboule) = boules (nombredeboules);
        nombredeboules = nombredeboules - 1;
        nombretirees = nombretirees + 1;
    end
```

```
    score = sum (numerostires .* (numerostires <= 12));
    if score >= 33
        gain = floor(12*(7/6)^(score-33));
    else
        gain = 0;
    end

    total = total + gain;
    % Il faut aussi maintenir le total des carrés.
    totalcarres = totalcarres + gain*gain;
end

moyenne = total / N;
% On estime l'écart-type de la quantité simulée par l'écart-type empirique.
ecarttype = sqrt(totalcarres / N - moyenne * moyenne);
disp ('Intervalle de confiance pour le gain moyen :');
% Un intervalle de confiance à 99,5 % correspond à 2,81 écarts-types.
disp ([num2str(moyenne),' ± ',num2str(2.81*ecarttype/sqrt(N))]);
end
```

A.2.3 Avec le conditionnement : `loterie_cond.m`

```
function loterie_cond

% La fonction "loterie_cond" effectue le même travail que "loterie_IC",
% mais en utilisant une technique de conditonnement. Je ne commente que les
% différences.

N = 700000;
Nnumeros = 36;
Ntirages = 12;
total = 0;
totalcarres = 0;

for i = 0 : N-1

    % Cette fois-ci on ne va réellement tirer que onze numéros.
    numerostires = zeros(1,Ntirages-1);
    boules = 1 : Nnumeros;
    nombredeboules = Nnumeros;
    nombretirees = 0;
    for j = 1 : Ntirages-1
        laboule = randi(nombredeboules);
        numerostires(nombretirees+1) = boules(laboule);
        boules (laboule) = boules (nombredeboules);
        nombredeboules = nombredeboules - 1;
        nombretirees = nombretirees + 1;
    end
```

```
    % Pour des raisons techniques, il va nous être pratique ici de
    % maintenir la liste des numéros sortis *parmi ceux que notre joueuse a
    % coché*. Nous codons cela sous la forme d'un tableau de 12 variables
    % booléennes valant ‘true’ en position #i si le numéro coché i est
    % sorti et ‘false’ sinon. Ce tableau est appelé "cochesetsortis".
    cochesetsortis = false * ones(1,12);
    for j = 1 : Ntirages-1
        if numerostires(j) <= Ntirages
            cochesetsortis(numerostires(j)) = true;
        end
    end

    % À l'issue de ces 11 tirages, nous obtenons un score partiel ;
    % maintenant, ce qui va nous intéresser est l'espérance du gain
    % conditionnellement à ce score partiel.
    scorepartiel = cochesetsortis * (1:12)';
    % Nous appelons "contribution" l'espérance conditionnelle. Nous
    % obtenons celle-ci simplement en comptant séparément les nouveaux cas
    % possibles.
    contribution = 0;
    for j = 1 : Ntirages
        if ~cochesetsortis(j)
            score = scorepartiel + j;
            if score >= 39
            contribution = contribution + floor(exp((score-39)/8));
            end
        end
    end
    % Chaque élément précédemment compté avait en fait une probabilité de 1
    % sur 25 ; il faut donc diviser "contribution" par 25 pour avoir la
    % véritable contribution.
    contribution = contribution / 25;

    total = total + contribution;
    totalcarres = totalcarres + contribution*contribution;
end

moyenne = total / N;
ecarttype = sqrt(totalcarres / N - moyenne * moyenne);
disp ('Intervalle de confiance pour le gain moyen :');
disp ([num2str(moyenne),' ± ',num2str(2.81*ecarttype/sqrt(N))]);

end
```

A.3 Le lapin de Douady

A.3.1 Calcul de l'aire du lapin : `lapin.m`

```
function lapin

% Calcul de la constante c, par la méthode de Newton.
c = -.1+.7i; % On initialise c près de la bonne valeur.
% Rappelons qu'on cherche à annuler le polynôme c³+2c²+c+1.
ancienneerreur = +Inf;
while true
    erreur = c*c*c + 2*c*c + c + 1;
    % On n'est pas sûr que l'erreur va arriver à exactement zéro ; c'est
    % pourquoi il faut prévoir une sortie de la boucle dès que l'erreur ne
    % décroit plus.
    if (abs(erreur) >= ancienneerreur)
        break;
    end
    derivee = 3*c*c + 4*c + 1;
    c = c - erreur / derivee;
    ancienneerreur = abs(erreur);
end

N = 1000000;
total = 0; % Total des résultats des simulations.

for i = 1:N
    % On tire z uniformément sur le carré.
    z = (-2+rand*4) + (-2+rand*4)*1i;
    % On lance la boucle des itérations de z jusqu'à pouvoir déterminer si
    % c'est un point du lapin ou non.
    while true
        if (abs(z) > 2.) % Le z initial n'appartenait pas au lapin.
            contribution = 0;
            break;
        elseif (abs(z) < .25) % Le z initial appartenait au lapin.
            contribution = 16;
            break;
        else
            z = z*z + c;
        end
    end
    total = total + contribution;
end

moyenne = total / N;
disp('Aire estimée du lapin de Douady :');
disp(moyenne);
end
```

A.4 SOS analyse complexe

A.4.1 Calcul de l'intégrale : `integrale.m`

```
function integrale
N = 1000000;
total = 0;
for i = 1 : N
    % On tire x suivant la loi de Pareto.
    x = 1 / rand - 1;
    % On calcule la contribution au total des simulations.
    contribution = (1+x)^2 * sin(x) / ((x^2 + 1)*x);
    total = total + contribution;
end

moyenne = total / N;
disp ('Évaluation de I :');
disp (moyenne);
end
```

A.5 Mathématiques financières

A.5.1 Algorithme "naïf" : `mathfi_naif.m`

```
% La fonction "mathfi_naif" évalue par la méthode de Monte-Carlo
% l'espérance du maximum d'une marche de 60 pas sur R issue de 0
% d'incréments N(1), sans utiliser de technique de réduction de variance
% particulière. "N" est le nombre de simulations demandées.
function mathfi_naif(N)
% On introduit le raccourci T pour le nombre de pas de temps.
T = 60;
% "somme" et "somme2" contiendront respectivement la somme des résultats
% simulés et la somme de leurs carrés.
somme = 0;
somme2 = 0;
for i = 1:N
    % On simule la trajectoire, resprésentée par le 61-vecteur "X".
    % (Attention, ce que nous appelons X_t dans le polycopié correspond
    % dans le programme à X(t+1), puisque sous MATLAB les indices
    % commencent à 1). Il sera plus rapide de pré-allouer la mémoire pour
    % X, en initialisant le vecteur avec des zéros.
    X = zeros(1,T+1);
    % Boucle pour remplir le vecteur X. "Xt" est la valeur courante de X,
    % et "t" le jour auquel on calcule Xt.
    %
    Xt = 0;
    for t = 1:T
    Xt = Xt + randn(1);
```

```
    X(t+1) = Xt;
    end
    % "Xmax".
    Xmax = max(X);
    % On met à jour somme et somme2.
    somme = somme + Xmax;
    somme2 = somme2 + Xmax * Xmax;
end
% "m" est l'estimateur de Monte-Carlo.
m = somme / N;
% "ectype" est l'écart-type (estimé) de X_*.
ectype = sqrt (somme2/N - m*m);
% "erreur" est le rayon de l'intervalle de confiance à 5 %.
erreur = 1.96 * ectype / sqrt(N);
% On affiche le résultat.
fprintf('Estimation de E(X_*) : %.5f ± %.5f\n', m, erreur);
% Le programme s'arrête.
return
end
```

A.5.2 Utilisation "rudimentaire" d'une variable de contrôle : mathfi_ctrl0.m

```
% "mathfi_ctrl0" effectue le même travail que "mathfi_naif", mais en
% utilisant la variable de contrôle Y. Je ne commente que les différences
% avec "mathfi_naif".
function mathfi_ctrl0(N)
T = 60;
% Attention : cette fois-ci les variables "somme" et "somme2" ne se
% réfèreront pas à la simulation de "Xmax", mais à celle de Xmax-Y, où "Y"
% est la variable de contrôle.
somme = 0;
somme2 = 0;
for i = 1:N
    X = zeros(1,T+1);
    Xt = 0;
    for t = 1:T
        Xt = Xt + randn(1);
        X(t+1) = Xt;
    end
    Xmax = max(X);
    % On calcule la variable de contrôle.
    Y = max(X(T+1),0);
    % La quantité à laquelle on va appliquer la méthode de Monte-Carlo à
    % proprement parler est la différence Xmax-Y, notée "valeur".
    valeur = Xmax - Y;
    somme = somme + valeur;
    somme2 = somme2 + valeur*valeur;
```

```
end
% On évalue l'estimateur et la marge d'erreur pour E(valeur).
m = somme / N;
ectype = sqrt (somme2/N - m*m);
erreur = 1.96 * ectype / sqrt(N);
% On en déduit l'estimateur pour E(Xmax) = E(valeur) + E(Y), sachant qu'on
% sait calculer exactement E(Y), appelée ici "EY". Noter que l'addition de
% EY ne modifie pas la marge d'erreur.
EY = sqrt(T / (2*pi));
mXmax = m + EY;
erreurXmax = erreur;
fprintf('Estimation de E(X_*) : %.5f ± %.5f\n', mXmax, erreurXmax);
return
end
```

A.5.3 Utilisation de la variable de contrôle en version paramétrée : `mathfi_ctrl1.m`

```
% "mathfi_ctrl1" utilise la même variable de contrôle que "mathfi_ctrl0",
% mais en version paramétrée. Je ne commente que les différences avec
% "mathfi_ctrl0".
function mathfi_ctrl1(N)
T = 60;
somme = 0;
somme2 = 0;
% Cette fois-ci, il faut aussi maintenir la somme des simulations de Y et
% des Y^2 ("sommeY" et "sommeY2"), ainsi que la somme des produits Xmax * Y
% ("sommeprod").
sommeY = 0;
sommeY2 = 0;
sommeprod = 0;
for i = 1:N
    X = zeros(1,T+1);
    Xt = 0;
    for t = 1:T
        Xt = Xt + randn(1);
        X(t+1) = Xt;
    end
    Xmax = max(X);
    Y = max(X(T+1),0);
    % Pour l'instant on ne sait pas encore quelle va être la valeur dont on
    % calcule l'espérance par Monte-Carlo, donc on maintient à jour toutes
    % les sommes introduites précédemment.
    somme = somme + Xmax;
    somme2 = somme2 + Xmax*Xmax;
    sommeY = sommeY + Y;
    sommeY2 = sommeY2 + Y*Y;
    sommeprod = sommeprod + Xmax*Y;
```

```
end
% On évalue la moyenne covariance "covariance" entre Xmax et Y, ainsi que
% la variance "varY" de Y. On en profite aussi pour calculer la variance
% "varXmax" de Xmax, qui nous servira plus loin.
moyenneXmax = somme / N;
moyenneY = sommeY / N;
covariance = sommeprod/N - moyenneXmax * moyenneY;
varY = sommeY2/N - moyenneY*moyenneY;
varXmax = somme2/N - moyenneXmax*moyenneXmax;
% On cherche maintenant le paramètre "lambda" optimal (ou du moins une
% estimation de celui-ci) pour lequel considérer la quantité d'intérêt
% Xmax-lambda*Y. On utilise pour ce faire la formule du cours.
lambda = covariance / varY;
% Maintenant, il s'agit de savoir quelles ont été l'espérance empirique "m"
% et l'écart-type "ectype" de la quantité d'intérêt Xmax-lambda*Y - toute
% l'astuce étant qu'on peut les calculées à partir dans quantités comptées
% dans la boucle de Monte-Carlo. m est l'estimateur de l'espérance de la
% quantité d'intérêt, dont on calcule la marge d'erreur à partir de ectype.
m = moyenneXmax - lambda * moyenneY;
ectype = sqrt (varXmax - 2*lambda*covariance + lambda*lambda*varY);
erreur = 1.96 * ectype / sqrt(N);
% On en déduit l'estimateur pour E(Xmax) = E(valeur) + lambda*E(Y). Comme
% précédemment, l'addition de EY ne modifie pas la marge d'erreur.
EY = sqrt(T / (2*pi));
mXmax = m + lambda*EY;
erreurXmax = erreur;
fprintf('Estimation de E(X_*) : %.5f ± %.5f\n', mXmax, erreurXmax);
return
end
```

A.5.4 Utilisation de trois variables de contrôle : `mathfi_ctrl3.m`

```
% "mathfi_ctrl3" fonctionne comme "mathfi_ctrl_1", mais en utilisant trois
% variables de contrôle Y_1, Y_2 et Y_3. Nous utilisons le formalisme
% vectoriel de MATLAB pour traiter les trois variables de contrôle
% collectivement. Je ne commente que les différences avec "mathfi_ctrl1".
function mathfi_ctrl3(N)
T = 60;
somme = 0;
somme2 = 0;
% Attention, "sommeY" est maintenant une quantité vectorielle, dont les
% entrées correspondent respectivement aux sommes des Y_1, des Y_2 et des
% Y_3.
sommeY = zeros(1,3);
% Attention, "sommeY2" est maintenant une quantité matricielle :
% "sommeY2(i,j)" contient la somme des produits Y_i * Y_j.
sommeY2 = zeros(3,3);
% Attention, "sommeprod" est maintenant une quantité vectorielle :
```

```
% "sommeprod(i)" contient la somme des produits X_* * Y_i.
sommeprod = zeros(1,3);
for i = 1:N
    X = zeros(1,T+1);
    Xt = 0;
    for t = 1:T
        Xt = Xt + randn(1);
        X(t+1) = Xt;
    end
    Xmax = max(X);
    % On calcule cette fois-ci trois variables de contrôle, regroupées dans
    % le vecteur "Y".
    Y = [max(X(21),0), max(X(41),0), max(X(61),0)];
    % Pour l'instant on ne sait pas encore quelle va être la valeur dont on
    % calcule l'espérance par Monte-Carlo, donc on maintient à jour toutes
    % les sommes introduites précédemment.
    somme = somme + Xmax;
    somme2 = somme2 + Xmax*Xmax;
    sommeY = sommeY + Y;
    % Noter le formalisme matriciel pour la mise à jour de sommeY2 (la
    % prime correspond à la transposition).
    sommeY2 = sommeY2 + Y'*Y;
    sommeprod = sommeprod + Xmax*Y;
end
moyenneXmax = somme / N;
moyenneY = sommeY / N;
covariance = sommeprod/N - moyenneXmax * moyenneY;
% Noter le formalisme matriciel pour le calcul de varY.
varY = sommeY2/N - moyenneY'*moyenneY;
varXmax = somme2/N - moyenneXmax*moyenneXmax;
% La paramètre "lambda" est maintenant vectoriel. Sous MATLAB, la commande
% "A / B", quand elle est appliquée à des matrices, calcule le produit A *
% B^{-1} (à l'inverse, "A \ B" calcule B^{-1} * A).
lambda = covariance / varY;
% Notre quantité d'intérêt est maintenant X_* - \lambda_1*Y_1 -
% \lambda_2*Y_2 - lambda_3*Y_3, soit, en formalisme vectoriel, X_* - lambda
% * Y'. À part le formalisme vectoriel auquel il faut faire attention, la
% façon de calculer l'espérance empirique et l'écart-type de X sont les
% mêmes que pour "mathfi_ctrl1".
m = moyenneXmax - lambda * moyenneY';
ectype = sqrt (varXmax - 2*lambda*covariance' + lambda*varY*lambda');
erreur = 1.96 * ectype / sqrt(N);
% "EY" est maintenant vectoriel.
EY = [sqrt(20/(2*pi)), sqrt(40/(2*pi)), sqrt(60/(2*pi))];
% Noter le formalisme vectoriel dans la ligne suivante.
mXmax = m + lambda*EY';
erreurXmax = erreur;
fprintf('Estimation de E(X_*) : %.5f ± %.5f\n', mXmax, erreurXmax);
```

```
return
end
```

A.5.5 Utilisation des variables antithétiques : mathfi_anti.m

```
% La fonction "mathfi_anti" améliore "mathfi_naif" en y implémentant la
% méthode des variables antithétiques. N est le nombre de simulations
% demandées. Je ne commente que les différences par rapport à
% "mathfi_naif".
function mathfi_anti(N)
% Pour commencer, comme les simulations vont en fait être effectuées par
% deux, le ‘‘vrai’’ N doit être divisé par deux par rapport à celui indiqué
% en paramètre.
N = ceil(N/2);
T = 60;
somme = 0;
somme2 = 0;
for i = 1:N
    % Il y a maintenant deux trajectoires à simuler, que nous appelons
    % respectivement "X" et "Xanti". Le tirage des gaussiennes qui servent
    % à simuler "Xanti" est inversé par rapport à celui de "X". (Noter
    % qu'ici la correspondance entre la trajectoire de X et celle de X anti
    % est triviale, mais la méthode de retournement des gaussiennes par
    % laquelle je suis passé a le mérite d'être plus générale).
    X = zeros(1,T+1);
    Xanti = zeros(1,T+1);
    Xt = 0;
    Xantit = 0;
    for t = 1:T
        r = randn(1);
        Xt = Xt + r;
        Xantit = Xantit - r;
        X(t+1) = Xt;
        Xanti(t+1) = Xantit;
    end
    % On évalue "Xmax" et "Xantimax" qui est la valeur de Xmax pour Xanti.
    Xmax = max(X);
    Xantimax = max(Xanti);
    % La valeur dont nous allons calculer l'espérance par Monte-Carlo est
    % celle de Xmax + Xantimax ; c'est par rapport à elle que nous
    % maintenons "somme" et "somme2".
    valeur = Xmax + Xantimax;
    somme = somme + valeur;
    somme2 = somme2 + valeur*valeur;
end
% "mvaleur", "ectype" et "errvaleur" sont les quantités de l'estimation de
% Monte-Carlo pour "valeur".
mvaleur = somme / N;
```

```
ectype = sqrt (somme2/N - mvaleur*mvaleur);
errvaleur = 1.96 * ectype / sqrt(N);
% L'estimation qui nous intéresse est celle de l'espérance de X_*, qui est
% la moitié de l'espérance de "valeur" : pour obtenir son estimation et son
% intervalle de confiance, il faut tout diviser par 2.
m = mvaleur / 2;
erreur = errvaleur / 2;
fprintf('Estimation de E(X_*) : %.5f ± %.5f\n', m, erreur);
return
end
```