MÉTHODES DE MONTE-CARLO
CORRIGÉ DE l'EXERCICE 9

par Rémi Peyre

EXERCICE 9 — Entraînement à la stratification

Dans cet exercice, on cherche à évaluer l'intégrale

$$I := \int_0^\infty \frac{\sin(2\pi/x)}{x^{1/4}}\,dx \tag{1}$$

par la méthode de Monte-Carlo.

1. Démontrer que l'intégrale (1) est absolument convergente.

Corrigé. La fonction à intégrer est continue sur $(0, +\infty)$; il y a donc juste à vérifier sa convergence aux extrêmités. Quand $x \to 0$, la fonction à intégrer peut être majorée en valeur absolue par $x^{1/4}$, qui est bien intégrable ; quand $x \to +\infty$, un développement limité de $\sin(t)$ quand $t \to 0$ montre que la fonction est équivalente à $x^{-5/4}$, qui est bien intégrable également. ✓

2. Proposer une méthode de Monte-Carlo pour évaluer I, et l'implémenter.

Corrigé. La quantité à intégrer ne s'écrivant pas natuellement comme une espérance, il faut choisir une loi d'échantillonnage pour appliquer la méthode de Monte-Carlo. Pour que la méthode converge, il suffit de s'assurer que cette loi d'échantillonnage ait une densité positive par rapport à la mesure de Lebsegue sur tout l'intervalle $(0, +\infty)$. Pour que la convergence soit efficace, il y a un critère L^2 à vérifier qui impose de choisir une densité qui « ressemble » suffisamment à la fonction à intégrer, en faisant en sorte que la densité ne soit beaucoup plus petite que la fonction qu'aussi rarement que possible. Je suggère de prendre comme loi d'échantillonnage $\mathcal{P}areto(\frac{1}{4}) - 1$, dont la densité est $m(x) = \frac{1}{4}(1+x)^{-5/4}$: on vérifie que, notant m cette densité et f la fonction à intégrer, on a bien $\mathbb{E}_m[(f/m)^2] < \infty$.

Cela conduit au code suivant :

```
function naif(N)
% La fonction "naif" calcule I par la méthode de Monte-Carlo avec une loi
% d'échantillonnage Pareto(1/4)-1.

somme = 0;
sommecarres = 0;
tic
for k=1:N
    % On simule x suivant la loi Pareto(1/4)-1.
    x = rand^(-4)-1;
    % y est la quantité dont on va prendre l'espérance. Notez qu'une
    % variable "pi" est préprogrammée par défaut dans MATLAB comme
    % valant... pi :-)
    y = sin(2*pi/x)/x^(1/4) / (1/4*(1+x)^(-5/4));
    somme = somme + y;
    sommecarres = sommecarres + y*y;
end
t = toc;
moyenne = somme/N;
variance = sommecarres/N - moyenne*moyenne;
```

```
disp('Intervalle de confiance à 2 sigmas :');
disp(moyenne+sqrt(variance/N)*[-2,2]);
disp('Efficacité :');
disp(N/t/variance);
end
```

✓

On pose

$$I_1 := \int_0^1 \frac{\sin(2\pi/x)}{x^{1/4}}\,dx\,; \qquad I_2 := \int_1^\infty \frac{\sin(2\pi/x)}{x^{1/4}}\,dx. \tag{2}$$

3. Proposer deux méthodes de Monte-Carlo *judicieuses* pour évaluer I_1 et I_2. Implémenter ces méthodes et évaluer leurs efficacités.

Corrigé. Par les mêmes arguments que dans la question précédente pour choisir la loi d'échantillonnage, je propose ici de prendre respectivement les lois d'échantillonnage $\beta(\frac{3}{4},1)$ (dont la densité, sur $(0,1)$, est $\frac{3}{4}x^{-1/4}$) et $\mathcal{P}areto(\frac{1}{4})$ (dont la densité, sur $(1,+\infty)$, est $\frac{1}{4}x^{-5/4}$). Je calcule les efficacités avec le code suivant :

```
function efficacites
% Cette fonction évalue les efficacités respectives du calcul des
% intégrales I_1 et I_2.

% Je fais mes évaluations avec 8000 simulations.
N = 8000;
% On commence par I_1.
somme = 0;
sommecarres = 0;
tic;
for k=1:N
    % On simule x suivant la loi bêta(3/4,1).
    x = rand^(4/3);
    % y est la quantité dont on prendrait l'espérance.
    y = sin(2*pi/x)/x^(1/4) / (3/4*x^(-1/4));
    somme = somme + y;
    sommecarres = sommecarres + y*y;
end
t = toc;
moyenne = somme/N;
variance = sommecarres/N - moyenne*moyenne;
disp('CALCUL DE L''INTÉGRALE I_1')
disp('Efficacité :');
disp(N/t/variance);
disp('Nombre de simulations par unité de temps :');
disp(N/t);
% Et maintenant, I_2.
somme = 0;
sommecarres = 0;
tic;
for k=1:N
    % On simule x suivant la loi Pareto(1/4).
    x = rand^(-4);
    % y est la quantité dont on prendrait l'espérance.
    y = sin(2*pi/x)/x^(1/4) / (1/4*x^(-5/4));
```

```
    somme = somme + y;
    sommecarres = sommecarres + y*y;
end
t = toc;
moyenne = somme/N;
variance = sommecarres/N - moyenne*moyenne;
disp('CALCUL DE L''INTÉGRALE I_2')
disp('Efficacité :');
disp(N/t/variance);
disp('Nombre de simulations par unité de temps :');
disp(N/t);
end
```

Pour l'intégrale I_1, je trouve une efficacité d'environ $7{,}5 \cdot 10^5\ \mathrm{s}^{-1}$, contre environ $7{,}5 \cdot 10^3\ \mathrm{s}^{-1}$ pour I_2. ✓

4. En déduire une technique de stratification (et plus précisément de stratification *a priori*) pour évaluer I. Optimiser l'échantillonnage interstrates et implémenter la méthode.

Corrigé. La méthode de stratification, en l'occurrence, consiste simplement en écrire $I = I_1 + I_2$ et à évaluer séparément les deux intégrales. Comment répartir le temps de calcul entre les deux parties ? Nous savons que, dans une situation de stratification *a priori* comme ici, le temps à consacrer à un calcul d'efficacité $\mathcal{E}\!f\!f$ doit être proportionnel à $\mathcal{E}\!f\!f^{-1/2}$. Ici le ratio entre les deux efficacités est d'environ 100, donc il faut consacrer $100^{1/2} = 10$ fois plus de temps à l'évaluation de I_2 qu'à celle de I_1. Pour convertir cela en nombre de simulations, l'évaluation que j'ai faite me disait aussi que le temps de calcul par simulation est environ 1,4 fois plus rapide pour I_2 que pour I_1 ($8{,}5 \cdot 10^5$ simulations par seconde contre $6 \cdot 10^5$), de sorte qu'il faut consacrer $1{,}4 \times 10 = 14$ fois plus de simulations à I_2 qu'à I_1.

Cela aboutit au programme suivant (noter le calcul de la variance globale comme somme des deux variances) :

```
function stratifie(N)
% "stratifie" évalue l'intégrale I en utilisant la technique
% d'échantillonnage stratifié. N est le nombre total de simulations.

% On répartit les simulations entre les deux intégrales.
N1 = ceil(N/15);
N2 = N-N1;
% Ici les tic et toc vont recouvrir l'ensemble des deux boucles.
tic
% On commence par I_1.
somme = 0;
sommecarres = 0;
for k=1:N1
    x = rand^(4/3);
    y = sin(2*pi/x)/x^(1/4) / (3/4*x^(-1/4));
    somme = somme + y;
    sommecarres = sommecarres + y*y;
end
moyenne1 = somme/N1;
% Ici ce que j'appelle "variance1" n'est pas la variance de la fonction
% dont on évalue l'intégrale, mais la variance de /l'estimateur/ de I1
% obtenu.
```

```
variance1 = (sommecarres/N1 - moyenne1*moyenne1) / N1;
% Et maintenant, I_2.
somme = 0;
sommecarres = 0;
for k=1:N2
    x = rand^(-4);
    y = sin(2*pi/x)/x^(1/4) / (1/4*x^(-5/4));
    %disp(y);
    somme = somme + y;
    sommecarres = sommecarres + y*y;
end
moyenne2 = somme/N2;
% Même remarque pour "variance2" que pour "variance1".
variance2 = (sommecarres/N2 - moyenne2*moyenne2) / N2;
t = toc;
% La variance est simplement la somme des deux variances.
variance = variance1 + variance2;
disp('Intervalle de confiance à 2 sigmas pour I :');
disp(moyenne1+moyenne2+sqrt(variance)*[-2,2]);
disp('Efficacité :');
disp(1/variance/t);
end
```

Notez qu'à l'exécution, j'obtiens chez moi une amélioration de l'efficacité négligeable ($\simeq +7$ %)... C'est que mon exemple pour l'exercice était mal choisi : ça m'apprendra à ne pas l'avoir testé avant de la proposer ☹. (Après réflexion, je pense que l'exercice aurait mieux marché en coupant l'intégrale en deux au niveau de l'abscisse 4 plutôt que 1. ✓