TRAVAUX PRATIQUES N° 1. — PROCESSUS DE LÉVY RÉELS

1. **Introduction théorique**

Un processus réel $X = (X_t)_{t\geqslant 0}$ est un processus à accroissements indépendants si et seulement si pour tous $0 = t_0 < t_1 < \cdots < t_n$, X_0, $X_{t_1} - X_0$, …, $X_{t_n} - X_{t_{n-1}}$ sont des variables aléatoires réelles indépendantes. Les accroissements sont homogènes si et seulement si la loi de $X_t - X_s$ ne dépend que de $t-s$, on la note alors μ_{t-s}. Les lois $(\mu_t)_{t\geqslant 0}$ forment alors un semi-groupe de convolution : $\mu_0 = \delta_{\{0\}}$ et $\mu_s * \mu_t = \mu_{s+t}$ pour tous $s, t \geqslant 0$. Lorsque le semi-groupe $(\mu_t)_{t\geqslant 0}$ satisfait de plus la condition de continuité $\mu_t \to \mu_0$ quand t tend vers 0 — ce qui équivaut à la continuité à droite du semi-groupe —, le processus X est alors qualifié de processus de Lévy.

Les processus à accroissements indépendants homogènes satisfont deux type d'homogénéité : l'homogénéité temporelle (invariance par translation dans le temps : ce sont des processus de Markov dans leur filtration naturelle), et l'homogénéité spatiale (invariance par translation dans l'espace). Ces deux propriétés les caractérisent. Le semi-groupe de transition $(P_t)_{t\geqslant 0}$ est alors lié au semi-groupe de convolution par

$$P_t(x, B) = \mathbb{P}^x\{X_t \in B\} = \mathbb{P}^x\{X_t - x \in B - x\} = \mu_t(B - x).$$

La composition des noyaux

$$\begin{aligned}(P_s \circ P_t)(x, B) &= \int_{\mathbb{R}} P_s(x, \mathrm{d}y) P_t(y, B) = \int_{\mathbb{R}} \mu_s(\mathrm{d}y - x)\mu_t(B - y) \\ &= \int_{\mathbb{R}} \mu_s(\mathrm{d}y)\mu_t\big((B - x) - y\big) = (\mu_s * \mu_t)(B - x) = P_{s+t}(x, B), \qquad x \in \mathbb{R},\ B \in \mathcal{B}(\mathbb{R}),\end{aligned}$$

correspond bien à la convolution des mesures de probabilité dans ce cas.

Les semi-groupes de convolution de mesures de probabilité sur $\mathbb{R}$ (et $\mathbb{R}^n$) continus en 0, c'est-à-dire continus à droite, ont été caractérisés par Paul Lévy, en particulier le semi-groupe $(\varphi_t)_{t\geqslant 0}$ de leurs fonctions caractéristiques vérifie $\varphi_t(\theta) = \mathbb{E}^0\big[\mathrm{e}^{\mathrm{i}\theta X_t}\big] = \mathrm{e}^{tg(\theta)}$ où

$$g(\theta) = \mathrm{i}a\theta - \frac{1}{2}\sigma^2\theta^2 + \int_{|x|\geqslant m} \big(\mathrm{e}^{\mathrm{i}\theta x} - 1\big)\, \nu(\mathrm{d}x) + \int_{|x|<m} \big(\mathrm{e}^{\mathrm{i}\theta x} - 1 - \mathrm{i}\theta x\big)\, \nu(\mathrm{d}x)$$

qui est la formule de Lévy–Khinchine (a est le coefficient de dérive [*drift*], σ le coefficient de diffusion, et ν la mesure de Lévy associés au processus). Parmi eux, nous avons (entre autres) :

- le semi-groupe brownien, $\mu_t(\mathrm{d}x) = \mathrm{e}^{-x^2/2t}\, \dfrac{\mathrm{d}x}{\sqrt{2\pi t}}$, $t \geqslant 0$;
- le semi-groupe de Poisson d'intensité $\lambda > 0$, $\mu_t(\mathrm{d}x) = \displaystyle\sum_{n=0}^{\infty} \mathrm{e}^{-\lambda t}\, \frac{(\lambda t)^n}{n\,!}\, \delta_{\{n\}}(\mathrm{d}x)$, $t \geqslant 0$.

Si nous savons que le mouvement brownien admet des versions à trajectoires continues, un processus de Poisson est essentiellement un processus de comptage et ses trajectoires sont en escalier avec sauts d'amplitude 1. La condition de continuité à droite ne garantit que l'existence de versions à trajectoires continues à droite et limitées à gauche (càdlàg).

⚄ À quelle condition un processus déterministe est-il un processus de Lévy ? Et si on ote la condition de continuité à droite ?

⚄ Constater que les combinaisons linéaires de processus de Lévy indépendants sont des processus de Lévy (on se dispensera de vérifier la condition de continuité, qui se traduit au niveau des processus par la continuité à droite en probabilité). Retrouver cette propriété au niveau des semi-groupes de convolution.

⚄ Reprendre son cours de L3 ou de M1 pour citer d'autres semi-groupes de convolution classiques (deux au moins).

2. Simulation de processus à accroissements indépendants homogènes

Fixons un horizon temporel $T = 1$ par exemple. Nous souhaitons représenter une trajectoire d'un processus de Lévy X (ou plus généralement, d'un processus à accroissements indépendants) sur $[0, T]$. Pour ce faire, nous divisons $[0, T]$ en n intervalles de longueur $\Delta t = T/n$, et tirons successivement les n accroissements correspondants. La trajectoire s'obtient alors en cumulant ces accroissements (on peut supposer $X_0 = 0$ puisqu'il y a homogénéité spatiale). Voici un exemple de code pour commencer :

```
// code scilab
T = 1; n = 1000;
dB = grand(n, 1, "nor", 0, sqrt(T/n));
time = (T/n)*[0:n]'; // vecteur colonne [0; T/n; ...; T*(n-1)/n; T]
B = cumsum([0; dB]); // trajectoire issue de 0
plot(time, B);
```

⚄ Simuler une trajectoire brownienne sur $[0, T]$ et la représenter par une animation au cours du temps. Une fois cette première trajectoire achevée, stocker la valeur terminale dans un tableau et recommencer la procédure. Après $N = 100$ trajectoires simulées, tracer un histogramme des N valeurs terminales obtenues.

Compte tenu du fait que l'ensemble de cette procédure a requis $n \times N$ appels au générateur aléatoire `grand()` (*Mersenne Twister ?*), l'histogramme vous paraît-il honnête ?

Remarque (explications empruntées). — Chaque image du tracé sera séparée de la suivante par 20 unités de temps, l'animation est obtenue en effaçant l'image précédente avant de dessiner l'image présente, une petite pause permet de séparer les différentes images. L'utilisation d'une boucle est nécessaire ici. Un bon effet d'animation est obtenu en « dessinant » dans le tampon vidéo puis en l'affichant en une seule fois. Les fonctions SCILAB pour gérer le tampon vidéo sont :

- Les dessins sont tous redirigés dans le tampon vidéo après que la commande d'activation `xset('pixmap',1)` a été exécutée.
- Le tampon vidéo est effacé par la commande `xbasc(`⟨*n° de figure*⟩`)`.
- Le tampon vidéo est affiché par la commande `xset('wshow')`.
- L'animation est rythmée par la commande `xpause(`⟨*délai en µs*⟩`)`.
- Le tampon vidéo est désactivé par la commande `xset('pixmap',0)`.

⚄ La commande `grand(`⟨*m*⟩`,`⟨*n*⟩`,`⟨*loi*⟩`,`⟨*paramètres*⟩`)` reconnaît la loi de Poisson (`"poi"`) de paramètre λ. reprendre l'activité précédente avec le processus de Poisson d'intensité $\lambda = 1$.

Contrairement à ce qui peut être fait pour générer une des nombres suivant une loi normale, le cas poissonnien peut être lourdement calculatoire et notamment faire des appels répétés au générateur aléatoire (*voir* la loi des événements rares). Qu'en est-il avec le générateur

`grand()` de SCILAB — ou sa bibliothèque ? (Réponse inconnue de l'auteur qui soupçonne cependant qu'elle se trouve dans les solutions données plus tard.)

⚄ La famille des lois de Cauchy semble généralement oubliée des programmeurs. Rappelons que loi de Cauchy de paramètre $a > 0$ est la loi de probabilité absolument continue par rapport à la mesure de Lebesgue admettant la fonction de densité suivante

$$x \in \mathbb{R} \longrightarrow \frac{1}{\pi} \times \frac{a}{a^2 + x^2}.$$

Le calcul de sa fonction caractéristique n'est pas des plus faciles (méthode d'analyse complexe), et donne $\varphi_a(\theta) = \exp(-a|x|)$. Il est alors clair qu'en étendant cette famille de mesures de probabilité par $\delta_{\{0\}}$ pour $a = 0$, on obtient un semi-groupe de convolution continu à droite.

⚄ Combler les lacunes de `grand()` en définissant son propre générateur de lois de Cauchy. Le mettre en œuvre pour réaliser les simulations associées.

3. **Retour sur le cas poissonnien**

Certains auteurs présentent les lois de Poisson (Siméon-Denis) comme apparaissant à travers le résultat asymptotique connu sous le nom de « loi des événements rares » ou « loi des petits nombres ». Celui-ci affirme que si $(p_n)_{n\geqslant 0}$ est une suite de réels positifs (inférieurs ou égaux à 1) tels que $n \times p_n \to \lambda \geqslant 0$ lorsque n tend vers l'infini, alors la suite des lois binomiales $\mathcal{B}(n, p_n)$ converge vers la loi de Poisson $\mathcal{P}(\lambda)$ de paramètre λ. Ce résultat permet, pour $\lambda \geqslant 0$ fixé, de simuler de manière approchée le tirage selon la loi de Poisson de paramètre λ à l'aide d'un schéma de Bernoulli (Jacob) de paramètres n et λ/n.

⚄ Considérer chaque pas de temps comme une épreuve du schéma de Bernoulli destiné à approcher la loi de Poisson de paramètre $\lambda = 1$ au temps $T = 1$. Représenter la chronologie du nombre de succès et donc une trajectoire approchée (?) d'un tel processus de Poisson.

Un processus de Poisson $N = (N_t)_{t\geqslant 0}$ d'intensité λ évolue uniquement par saut de hauteur 1. Les durées des intervalles de temps successifs entre deux sauts sont des variables aléatoires $(T_k)_{k\geqslant 1}$ indépendantes et identiquement distribuées de loi la loi exponentielle de paramètre (d'intensité) λ. Les temps T_k sont souvent appelés temps de « record » du processus de Poisson considéré : ce sont les durées qu'il faut attendre pour enregistrer quelque chose. En posant, la date du n-ième saut est $S_n = \sum_{k=1}^{n} T_k$, et on a $T_n = S_n - S_{n-1}$ en convenant que $S_0 = 0$ (la loi de S_n est une loi Gamma de paramètres n et λ). On a alors $N_t = \inf\{n \geqslant 0 : S_n > t\} = \sup\{n \geqslant 0 : S_n \leqslant t\}$.

⚄ Rappeler comment on génère un nombre selon la loi exponentielle $\mathcal{E}(\lambda)$ à partir d'une entrée uniforme sur $[0, 1[$.

⚄ Se servir de la description précédente pour définir un générateur aléatoire des lois de Poisson `Poissonrand(`⟨*lambda*⟩`)`.

⚄ Pour $\lambda = 1$ et $T = 1$, effectuer la simulation des trajectoires par tirages successifs des temps de record. Comparer la distribution observée de la variable terminale avec la loi de Poisson de paramètre 1 (on pourra créer le vecteur des « probabilités théoriques » en lui donnant pour taille celle qui correspond au maximum observé et prendra garde à ce que le diagramme de cette sous-probabilité ne soit pas renormalisé).

Remarque. — L'utilisation des temps de record semble plus économique et fiable que celle de la loi des événements rares. Certaines situations cependant — en particulier dès que l'homogénéité spatiale ou temporelle est rompue — amènent à choisir une approche de type Bernoulli. Mais il ne s'agit plus de processus de Lévy.

4. Retour sur le cas brownien

Si la loi des événements rares permet d'approcher un processus de Poisson via un schéma de Bernoulli, une version étendue du théorème central limite permet une démarche semblable pour le mouvement brownien.

Le « théorème d'invariance de Donsker » affirme qu'on peut approcher le mouvement brownien (unidimensionnel) par toute marche aléatoire raisonnable convenablement normalisée. Sans rentrer dans les détails : Soit $(\xi_n)_{n\geqslant 0}$ une suite de variables aléatoires réelles indépendantes, identiquement distribuées de moyenne nulle et de variance 1. En posant, pour $k \geqslant 0$, $X^{(n)}_{k/n} = \sum_{\ell=1}^{k} \xi_\ell/\sqrt{n}$ et en prolongeant cette suite de manière continue et affine par morceaux de la manière la plus simple qu'il soit, on obtient un processus $X^{(n)} = (X^{(n)}_t)_{t\geqslant 0}$ approchant le mouvement brownien standard.

⚄ Mettre en œuvre ce schéma d'approximation en partant de la marche aléatoire simple symétrique sur $\mathbb{Z}$. Comparer les lois terminales.

⚄ Dans toutes les simulation précédentes, nous n'avons finalement que comparé les lois terminales observées avec les lois exactes, et ce simplement graphiquement (un test de Kolmogorov–Smirnov serait envisageable après implémentation en Scilab des fonctions de répartition de référence de ce test). Que peut-on dire de l'indépendance des accroissements sans effectuer le moindre test, ou en supposant qu'ils ont été menés avec succès par d'autres ?

L'utilisation des temps de record d'un processus de Poisson permet de décrire les trajectoires de celui-ci à tout instant. Pour les simulations du mouvement brownien que nous avons faite, nous ignorons ce qu'il a pu se passer entre deux instants successifs de la simulation : nous avons les valeurs de B_t et de $B_{t+\Delta t}$ mais pas de B_s pour $s \in]t, t+\Delta t[$. Il faudrait pouvoir recoller un morceau de trajectoire brownienne entre ces deux instants, c'est-à-dire jeter un *pont brownien* entre B_t et de $B_{t+\Delta t}$ aux dates t et $t+\Delta t$.

On appelle pont brownien (*Brownian bridge*) « le processus » $\beta = (\beta_t)_{t\in[0,1]}$ obtenu en conditionnant le mouvement brownien $(B_t)_{t\in[0,1]}$, issu de 0, à valoir 0 au temps 1. La théorie du conditionnement des vecteurs gaussiens (infini-dimensionnels) justifie l'existence d'un tel objet et mieux encore en donne une réalisation extrêmement simple : il suffit de prendre le processus défini par $\beta_t = B_t - tB_1$ (ce qui est très exceptionnel, voire unique en son genre).

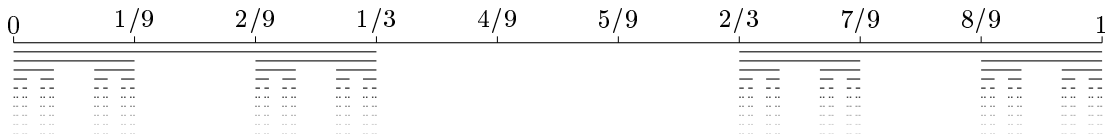
⚄ Sans nécessairement l'implémenter, réfléchir à un schéma de simulation permettant de zoomer entre deux subdivisions, autrement dit, de raffiner une simulation existante sans la reprendre depuis le début.

Remarque. — La génération des accroissements browniens directement à partir des lois normales est très efficace pour « voir le brownien lui-même » dans un milieu spatial homogène. L'approche par un schéma de Bernoulli peut être pertinente lorsqu'on considère des diffusions, en quelque sorte des mouvements browniens inhomogènes (*voir* équations différentielles stochastiques).

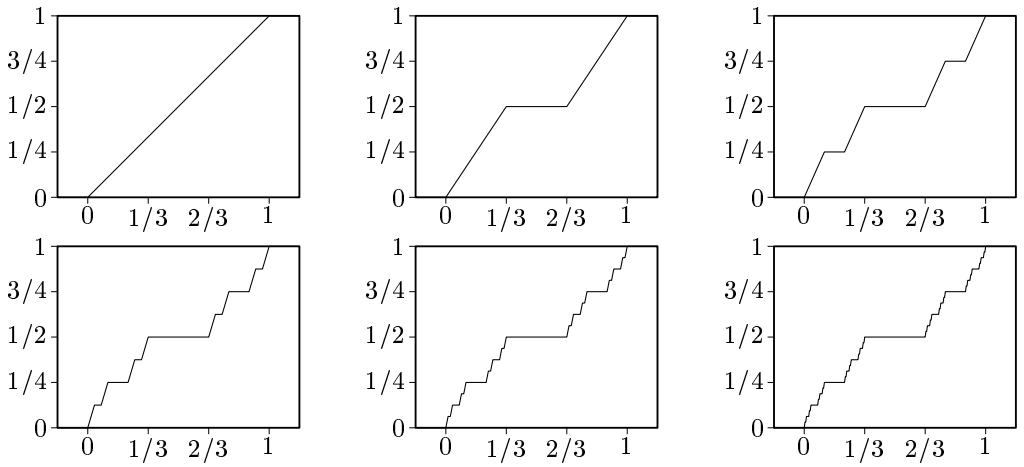
Bonus : la fonction de Cantor

La fonction de Cantor est généralement présentée en cours d'intégration de L3 conjointement à l'ensemble triadique C de Cantor. Ce dernier est obtenu en partant de l'intervalle $C_0 = [0,1]$ à qui on enlève dans un premier temps l'intervalle ouvert $]1/3, 2/3[$ pour obtenir C_1 ; puis des intervalles restants, on enlève les tiers ouverts centraux pour obtenir C_2, etc. On obtient ainsi une suite décroissante de compacts $(C_n)_{n\geqslant 0}$ de longueur (de mesure de Lebesgue) respective $(2/3)^n$. L'ensemble $C = \bigcap_{n\geqslant 0} C_n$ est alors un compact non vide de mesure

nulle. Il est sans point isolé et de cardinal continu.



Dans le même esprit, on construit une suite de fonctions continues, croissantes affines par morceaux valant 0 en 0 et 1 en 1, et convergeant uniformément sur $\mathbb{R}$ vers une fonction continue F, croissante valant 0 en 0 et 1 en 1 et constante sur les composantes connexes de C^c. Les graphes suivants précisent la construction de F.



Quant au code MetaPost suivant, il met l'accent la construction récursive de ces figures.

```
% code metapost
def cantorgraph(expr u, v, depth) =
    if depth > 0:
        cantorgraph(u,(xpart(1/3[u,v]),ypart(1/2[u,v])),depth-1)
        --cantorgraph((xpart(2/3[u,v]),ypart(1/2[u,v])),v,depth-1)
    else: u--v fi
enddef;

n := 5;
for i = 0 upto n:
    beginfig(thisfig);
        setrange(-1/6,7/6,8/3cm,0,1,2cm);
        pickup rule.nib;
        gdraw (hmin,0)--cantorgraph((0,0),(1,1),i)--(hmax,1);
        yticks.lft(hmin) "0", "1/4", "1/2","3/4", "1";
        xticks.bot(vmin) "0", "1/3", "2/3", "1";
    endfig;
endfor
```

1° *Loi de variable aléatoire.* — La fonction F est une fonction de répartition telle que $F(0) = 0$ et $F(1) = 1$. Il existe donc une unique mesure de probabilité μ sur $[0, 1]$ dont F est la fonction de répartition. Soit $X : (\Omega, \mathcal{A}, \mathbb{P}) \to [0, 1]$ une variable aléatoire de loi μ.

⚄ Déterminer l'espérance de X par des arguments élémentaires mais rigoureux.

⚄ Mettre en place en Scilab une fonction permettant de calculer de manière approchée $\mathbb{E}[f(X)]$ pour toute fonction $f : [0, 1] \to \mathbb{R}$ continue. On pourra pour cela approcher X par des variables aléatoires de loi uniforme sur des ensembles finis à préciser.

⚄ Approcher $\mathbb{E}[X^2]$ ainsi que la variance de X. Leur détermination théorique vous semble-t-elle accessible ? Si oui, comment ?

Correction. — ⚄ La loi de X est symétrique par rapport à $1/2$, comme X est bornée, elle est intégrable et alors $\mathbb{E}[X] = 1/2$.

⚄ Pour $n \geqslant 0$, posons $X_n^-(\omega) = k/3^n$ et $X_n^+(\omega) = (k+1)/3^n$ si $X(\omega) \in [k/3^n, (k+1)/3^n]$, $k = 0, \dots, 3^n - 1$. Les suites de variables aléatoires $(X_n^-)_{n\geqslant 0}$ et $(X_n^+)_{n\geqslant 0}$ convergent uniformément vers X ($\|X_n^\pm - X\|_\infty \leqslant 1/3^n$) et on constate qu'elles sont, pour $n \geqslant 0$ donné, uniformément réparties respectivement sur les extrémités gauches et droites des 2^n intervalles de C_n. Le calcul approché des espérances est alors immédiat :

```
% code metapost
vardef f expr x = x**2 enddef;

% remarquer que la normalisation n'appara\^it qu'\`a la fin du calcul

def EX expr depth = EXapprox := EXsub(0,1,depth)/(2**depth); message
    "E[f(X)] = " & decimal(EXapprox); enddef;

vardef EXsub(expr u,v,depth) = if depth >0:
    EXsub(u,1/3[u,v],depth-1)+EXsub(2/3[u,v],v,depth-1)
    else: f(u) fi% ou f(v)
enddef;

EX 8;
```

⚄ L'espérance de $(X_n^-)^2$ s'écrit sous la forme d'une série :

$$\mathbb{E}\big[(X_n^-)^2\big] = \frac{1}{2^n} \sum_k \Big(\frac{k}{3^n}\Big)^2$$

ou si on sait que X peut s'écrire $\sum_{n\geqslant 1} \xi_n/3^n$ avec $(\xi_n)_{n\geqslant 1}$ variables aléatoires indépendantes uniformément distribuées sur $\{0, 2\}$, on a

$$\mathbb{E}\big[X^2\big] = \sum_{m,n\geqslant 1} \frac{\mathbb{E}[\xi_m \xi_n]}{3^{m+n}} = \sum_{m\geqslant 1} \frac{2}{(9)^m} + 2 \sum_{1\leqslant m<n} \frac{1}{3^m 3^n} = \frac{2}{8} + \frac{2}{8} \times \frac{1}{2} = \frac{3}{8}\,.$$

En précisant tout ça proprement, ça devrait être jouable avec un peu d'effort. Le calcul numérique indique que $\mathbb{E}[X^2] = 3/8$ (comme prédit), donc $\mathrm{Var}(X) = 3/8 - 1/4 = 1/8$. Notons que si U est de loi uniforme sur $[0, 1]$, $\mathbb{E}[U] = 1/2$, $\mathrm{Var}(U) = 1/3 - 1/4 = 1/12$, ce qui indique bien que la mesure de Cantor est plus dispersée autour de sa valeur moyenne que ne l'est la loi uniforme.

2° *Processus à variation finie.* — Considérons maintenant la fonction F comme un processus croissant (déterministe). C'est une fonction continue dont la dérivée est définie et nulle presque partout, pourtant en général,

$$\mu\big(\mathbb{1}_{]a,t]} g\big) =: \int_a^t g(s)\,\mathrm{d}F(s) \neq \int_a^t g(s) F'(s)\,\mathrm{d}s = 0$$

où F' est une fonction nulle en tout point où F est dérivable. (On notera la convention concernant les bornes d'intégration. Elle est nécessaire pour avoir une relation Chasles lorsque la mesure μ comporte une partie discrète, notamment ici si $\mu\{a\} > 0$, ce qui n'est cependant pas le cas ici.)

⚄ Définir avec SCILAB la fonction $t \in \mathbb{R} \mapsto F(t)$. L'utiliser cette définition dans l'implémentation d'un calcul approché par des sommes de Riemann

$$\int_0^t g(s)\,\mathrm{d}F(s) \approx \sum_{0=t_0<\cdots<t_n=t} g(t_k)\big(F(t_{k+1}) - F(t_k)\big)$$

prises sur des subdivisions régulières.

⚄ Écrire une fonction SCILAB permettant de calculer ces approximations d'intégrales de manière récursive.

⚄ Mettre en œuvre les deux méthodes pour $g(s) = s$ et $g(s) = s^2$ avec $t = 1/4,\ 1/2,\ 3/4,\ 1$ (les valeurs exactes ne sont pas demandées).

⚄ Quelle méthode semble la plus performante ? Vous semble-t-il que l'on puisse aisément généraliser cette dernière méthode à toute fonction de répartition ? voire plus généralement à toutes fonctions monotones ou différences de telles fonctions (fonctions à variation finie) ?

Correction. — ???

Remarque. — L'évocation de la fonction de Cantor est une forme d'introduction au temps local brownien dont il devrait être question dans une fiche de travaux dirigé prochaine (on l'espère).

Références

[1] BERTOIN (Jean), *Lévy processes*, Cambridge University Press.

[2] CHANCELIER (J.-P.), DELEBECQUE (F.), GOMEZ (C.), GOURSAT (M.), NIKOUKHAH (R.), STEER (S.), *Introduction à Scilab, deuxième édition*, Springer-Verlag France (2007).

[3] DELLACHERIE (C.), MEYER (P.-A.), *Probabilités et potentiel*, Chapitres XII à XVI, Hermann (1987).

Université de Poitiers
Département de Mathématiques

3M21–Processus à temps continu
Master MMAS 2, année 2009–10

TRAVAUX PRATIQUES N° 1. — PROCESSUS DE LÉVY RÉELS, COMPLÉMENTS PARTIELS

Nous reprenons la fiche de travaux pratiques n° 1 en précisant pas à pas les activités qui y sont proposées. Ces deux fiches sont donc à suivre simultanément, en particulier ce serait un non-sens que de considérer celle-ci comme auto-suffisante.

1. **Introduction théorique**

Ce qui suit ne sert qu'à conserver une trace écrite de ce qui a été vu la fois précédente.

Nous avons vu qu'un processus déterministe à valeurs réelles $(x(t))_{t\geqslant 0}$ est à accroissements indépendants (les constantes étant des variables aléatoires indépendantes de tout ce qu'on veut), et qu'il est homogène si pour tout $0 \leqslant s \leqslant t$, $x(t) - x(s) = x(t-s) - x(0)$. En notant $\Delta x(t) = x(t) - x(0)$, $t \geqslant 0$, on constate que $\Delta x(t)$ est $\mathbb{Q}_+$-homogène (si on pose $\Delta x(t) = -\Delta x(-t)$ pour $t \leqslant 0$, alors Δx est $\mathbb{Q}$-linéaire). La condition de continuité à droite du semi-groupe de convolution se traduit ici par la continuité à droite de Δx et assure, en passant à la limite, que Δx est $\mathbb{R}_+$-homogène : $\Delta x(t) = t \times \Delta x(1)$ pour tout $t \geqslant 0$, soit $x(t) = x(0) + t \times \Delta x(1)$. Sans hypothèse de régularité aucune, on peut *choisir* des solutions très irrégulières à ce problème, il suffit de considérer $\mathbb{R}$ comme un $\mathbb{Q}$-espace vectoriel et *choisir* $\Delta x : \mathbb{R} \to \mathbb{R}$ une application $\mathbb{Q}$-linéaire différente d'une homothétie.

Évidemment, nous omettons quelques détails sur lesquels le lecteur pourra revenir pour lui-même.

Il est évident que si $X = (X_t)_{t\geqslant 0}$ est un processus à accroissements indépendants, homogènes, dont le semi-groupe de convolution vérifie la propriété de continuité à droite en 0, il en est de même de $aX = (aX_t)_{t\geqslant 0}$ pour toute constante $a \in \mathbb{R}$ (l'écrire proprement).

De même, si X et Y sont deux processus *indépendants* (bien comprendre la force de cette condition), à accroissements indépendants, alors $X + Y$ est un processus à accroissements indépendants, à accroissements homogènes, alors $X + Y$ est à accroissements homogènes, vérifiant la condition de continuité à droite en $t = 0$, alors il en est de même pour $X + Y$ (on pourra noter que la condition de continuité à droite en 0 des semi-groupe équivaut à la continuité à droite en probabilité des processus).

Quelques semi-groupes de convolution :

- $(\delta_{v\times t})_{t\geqslant 0}$ le semi-groupe de translation uniforme à vitesse $v \in \mathbb{R}$ (mouvement rectiligne uniforme).
- $(\mathcal{N}(v\times t, \sigma^2\times t))_{t\geqslant 0}$ le semi-groupe du mouvement brownien avec dérive (*drift*) de variance $\sigma^2 \geqslant 0$ (non standard).
- $(\mathcal{P}(\lambda \times t))_{t\geqslant 0}$ le semi-groupe du processus de Poisson d'intensité $\lambda \geqslant 0$.
- $(\mathcal{C}auchy(a \times t))_{t\geqslant 0}$ le semi-groupe de Cauchy de paramètre $a \geqslant 0$.
- $(\Gamma(\lambda, a\times t))_{t\geqslant 0}$ le semi-groupe des lois Gamma de paramètres $\lambda \geqslant 0$ et $a\times t \geqslant 0$. Rappelons qu'une loi Gamma de paramètres $\lambda > 0$ et $k > 0$ est la loi absolument continue sur $\mathbb{R}$ admettant pour densité

$$p(x) = \mathbb{1}_{[0,+\infty[}(x)\, \frac{\lambda^k}{(k-1)\,!}\, x^{k-1}\, \mathrm{e}^{-\lambda x}, \qquad x \in \mathbb{R}.$$

Si X est une variable aléatoire de loi $\Gamma(\lambda, k)$, alors

$$\mathbb{E}[X] = \frac{k}{\lambda}, \qquad \mathrm{Var}(X) = \frac{k}{\lambda^2}, \qquad \varphi_X(\theta) = \Big(\frac{\lambda}{\lambda - \mathrm{i}\theta}\Big)^k.$$

L'extension du calcul de la fonction caractéristique pour k non entier est un peu délicat. Néanmoins on constate avec cette expression l'aspect multiplicatif en k de ces fonctions caractéristiques et ainsi la nature de semi-groupe de convolution.

- etc.

2. Simulation de processus à accroissements indépendants homogènes

Saisir le programme suivant pour effectuer les simulations et ce, même si vos programmes antérieurs sont tout aussi satisfaisants.

```
// code scilab

clear();

// Simulation de processus \‘a accroissements ind\’ependants homog\‘enes

// <nom> est une cha\^\i ne de caract\‘eres servant \‘a donner un titre
// \‘a la figure obtenue (et \‘a structurer la programmation).
// <dX> est le vecteur des accroissements.
// <T> est le temps terminal : les trajectoires sont simul\’ees sur [0, T].
// <n> est le nombre de pas (le nombre de lignes de <dX>) :
// en comptant la position initiale (t = 0, x = 0),
// il y aura n+1 points pour repr\’esenter la trajectoire.
// <N> est le nombre de trajectoires (le nombre de colonnes de <dX>).

function processus(nom, dX, T)
    // local n N time X Xzero Xmin Xmax i j;
    [n, N] = size(dX);
    time = [0:n]'*T/n; // n+1 instants
    Xzero = 0; // valeur initiale
    X = cumsum([Xzero*ones(1,N); dX], "r"); // n+1 valeurs
    Xmin = min(X); Xmax = max(X);
    //
    // param\’etrage de la fen\^etre graphique (visuel)
    // et des axes (coordonn\’ees r\’eelles ou math\’ematiques)
    //
    clf();
    xset("pixmap", 0);
    f = gcf(); // get current figure
    f.color_map = hotcolormap(N); // gradient de couleurs
    // f.auto_resize = "off";
    // f.figure_size = [400,300]; // taille de la fen\^etre visible
    f.axes_size = [400,300]; // taille de la fen\^etre virtuelle
    a = gca(); // param\‘etres des axes
    a.data_bounds = [0, Xmin; T, Xmax];
    a.axes_visible = "on";
    //
    // trac\’e progressif de la trajectoire (animation)
```

```
    //
    xtitle(["simulation d''un", nom]);
    f.pixmap = "on"; // mode double tampon pour animation
    for j = 1:min(5, N)// limitation volontaire et raisonnable
        for i = 1:n
            plot([time(i:i+1)], [X(i:i+1,j)], style=j);
            show_pixmap();
        end
    end
    f.pixmap = "off";
    halt("*** taper [entree] pour continuer ***\n");
    //
    // histogramme des valeurs terminales (derni\`ere ligne)
    // on le remplacera par un cgoftest ou dgoftest plus tard.
    //
    // clf(); histplot(int(log(N)/log(2)+1), X(n+1,[1:N]));
    // halt("*** taper [entree] pour continuer ***\n");
endfunction
```

Simulation de trajectoires browniennes. Nous ne poussons pas les paramètres trop loin pour commencer. (Décommenter les lignes intéressantes.)

```
T = 1; n = 100; N = 10;

// Accroissements browniens standard
// dB = grand(n, N, "nor", 0, sqrt(T/n));// m = 0, \sigma^2=1
// processus("mouvement brownien", dB, T);
```

Simulation de trajectoires poissonniennes. Tout est à disposition pour ce faire : la fonction `processus()` plus haut, le générateur aléatoire. (Décommenter les lignes intéressantes.)

```
// Accroissements poissonniens standard
// dN = grand(n, N, "poi", T/n);// lambda = 1
// processus("processus de Poisson", dN, T);
```

Simulation de trajectoires d'un processus de Cauchy. Il est demandé de vérifier que ce qui est programmé ici correspond à des lois de Cauchy dont on vérifiera les paramètres. (Décommenter les lignes intéressantes.)

```
// Accroissements de Cauchy standard
// dC = tan(%pi*(grand(n, N, "def")-0.5))*T/n;
// processus("processus de Cauchy", dC, T);
```

Et un dernier pour la route (fort heureusement prévu dans l'implémentation de `grand()`) : lois Gamma. (Décommenter les lignes intéressantes.)

```
// Accroissements Gamma standard
// dG = grand(n, N, "gam", T/n, 1);
// processus("processus Gamma", dG, T);
```

2.1. Test de Kolmogorov–Smirnov

Dans ce qui suit, la suite `x` (ou plus exactement `u`) est censée représenter un *échantillon* (*sample* en anglais) d'une loi de probabilité μ sur $\mathbb{R}$ de fonction de réparttion F. Une procédure classique pour évaluer la qualité de cet échantillonnage consiste à comparer la fonction de répartition observée avec la fonction de répartition cible.

La fonction de répartition empirique F_n associée à l'échantillon $(x_1, \dots, x_n)$ est définie par

$$F_n(x) = \frac{1}{n} \operatorname{Card}\{1 \leqslant i \leqslant n : x_i \leqslant x\}.$$

pour tout $x \in \mathbb{R}$. Elle prend ses valeurs dans $\{0, 1/n, \dots, (n-1)/n, 1\}$. La *statistique* de Kolmogorov–Smirnov associée à ce problème est

$$k = \|F - F_n\|_\infty = \sup_{x \in \mathbb{R}} |F(x) - F_n(x)|,$$

et on montre que si F est continue on a

$$k = \max_{i=1}^n \big(F(x_{(i)}) - (i-1)/n\big) \vee \big(i/n - F(x_{(i)})\big),$$

où $x_{(1)} \leqslant \cdots \leqslant x_{(n)}$ est l'échantillon ordonné dans le sens croissant et $a \vee b = \max(a, b)$, et qui est une expression facilement calculable sur ordinateur (le tri s'obtient avec des commandes `sort(`$\langle x \rangle$`)` ou `gsort(`$\langle x \rangle$`)` [attention à l'ordre de tri], le maximum avec `max(`$\langle x \rangle$`)`).

Pour une méthode donnée, cette statistique k va varier en fonction de l'échantillon obtenu. Lorsque celle-ci est généralement petite — ce qui est quantifié par la loi de Kolmogorov de paramètre n —, l'hypothèse d'uniformité est acceptable, sinon il faut remettre en cause la méthode.

Si le logiciel dispose de fonctions de calcul sur les distributions de Kolmogorov, on peut calculer la p-valeur du test d'adéquation de Kolmogorov–Smirnov à la loi cible : celle-ci est simplement $1 - F_{K_n}(k)$ où F_{K_n} est la fonction de répartition de la loi de Kolmogorov de paramètre n la taille de l'échantillon et k la statistique du test. Typiquement, si la p-valeur est inférieure à 5 %, on rejette l'hypothèse d'adéquation.

Avec Maple et Scilab, les fonctions de répartitions de lois de probabilités sont des commandes comportant `cdf` (*Cumulative Distribution Function*) dans leurs noms. Les commandes disponibles pour ces logiciels se limitent à des lois de probabilité relativement communes (et pas encore les lois de Kolmogorov).

Remarque. — Dudley (1964) a montré que pour tout $u > 0$,

$$\lim_{n \to \infty} \mathbb{P}\big\{K_n \leqslant u/\sqrt{n}\big\} = 1 + 2 \sum_{k=1}^{\infty} (-1)^k \exp\big(-2k^2u^2\big).$$

ou encore pour $x > 0$, lorsque n tend vers l'infini,

$$1 - \mathbb{P}\big\{K_n \leqslant x\big\} \sim 2 \sum_{k=1}^{\infty} (-1)^{k+1} \exp\big(-2k^2nx^2\big) = 2\exp\big(-2nx^2\big) - 2\exp\big(-8nx^2\big) + \cdots$$

En utilisant cette approximation grossière, on peut définir une fonction

`KSasymptoticpvalue(`$\langle u \rangle$`, `$\langle F \rangle$`)`

calculant la p-valeur approchée du test de Kolmogorov–Smirnov d'adéquation. Nous ne le ferons pas.

Correction. — Il est tout de même assez facile d'obtenir les fonctions de répartition approchées des distributions de Kolmogorov. Voici un code complet dont une partie a été intégrée à `kolmogorov.sci`.

```
// code scilab

global accuracy; accuracy = 1e-12;

// kolmogorov limiting cdf

function p = klmcdf(x, n)
    if x <= 0 then p = 0;
    elseif x >= 1 then p = 1;
    else
        // local z p term k pm;
        global accuracy;
        z = -2*(sqrt(n)+0.12+0.11/sqrt(n)^2*x*x;
        p = 1.0; k = 1; pm = -1; term = 1;
        while abs(term) > accuracy && k < 100
            term = 2*pm*exp(z*k*k);
            p = p+term; pm = -pm; k = k+1;
        end
        p = max(min(p, 1), 0);
    end
endfunction

function p = KSasymptoticpvalue(u, F)
    // local ks v n i;
    n = size(u,1); v = -gsort(-u); ks = 0;
    for i = 1:n; ks = max(ks, F(v(i))-(i-1)/n, i/n-F(v(i))); end
    p = 1-klm(ks, n);
endfunction
```

Exercice 1 (*pratique*). — Des implémentations des fonctions de répartitions des Lois de Kolmogorov se trouvent dans le répertoire

`http://www-math.univ-poitiers.fr/~phan/masterMMAS/documents/1m02/`

sous la forme de fichiers de macros `kolmogorov.sci` pour Scilab et `kolmogorov.txt` pour Maple. Récupérer le fichier `.sci` (*Scilab Input*), examiner rapidement son contenu. En profiter pour récupérer `goodness-of-fit.sci`, et, là encore, examiner le contenu (en particulier la fonction `cgoftest`).

Exemple. — Voici un exemple d'utilisation de `cgoftest` où on s'intéresse à la loi uniforme sur $[0, 1]$:

```
getf("kolmogorov.sci");
getf("goodness-of-fit.sci");

N = 100; u = zeros(N, 1);
for i = 1:N; u(i) = rand(); end // u doit \^etre un vecteur colonne
deff("y = F(x)", "y = min(max(x, 0), 1)");
cgoftest(u, F);
```

2.2. Test du χ^2 d'adéquation

Considérons une loi discrète. Si c'est une mesure de probabilité sur $\mathbb{R}$, alors sa fonction de répartition F est une fonction en escalier. La comparaison de cette fonction de répartition avec celle qui a été observée ne peut se faire par l'approche de Kolmogorov–Smirnov pour laquelle F est supposée continue. Dans le cas discret, nous nous retrouvons donc à devoir trouver une approche complémentaire de celle de de Kolmogorov–Smirnov. Celle-ci existe, c'est le *test du χ^2 d'adéquation à une loi discrète*.

Considérons une loi discrète π portée par un ensemble fini $E = \{e_1, \ldots, e_k\}$ et notons $\pi_i = \pi\{e_i\} > 0$. Si $x = (x_j)_{j=1}^n$ est une suite de points de E, un échantillon, on note $p_i = n_i/n$ la proportion de termes égaux à e_i. La distribution observée est proche de π si tous les p_i sont proches des π_i correspondants. La statistique associée à ce problème est la *statistique du test du χ^2 d'adéquation* (χ^2 se dit « khi-deux » ou *"chi-square"*)

$$cs = \sum_{i=1}^{k} \frac{(n_i - n \times \pi_i)^2}{n \times \pi_i} = n \times \sum_{i=1}^{k} \frac{(p_i - \pi_i)^2}{\pi_i},$$

et devrait prendre généralement de « faibles » valeurs quand la distribution observée est proche de la distribution cible. Ceci est quantifié de manière asymptotique en n par la loi du χ^2, ou de Pearson, à $\nu = k - 1$ degrés de liberté. Si F_{k-1} désigne la fonction de répartition de la loi du χ^2 à $k - 1$ degrés de liberté, la p-valeur asymptotique du test d'adéquation est $1 - F_{k-1}(cs)$. Lorsque des conditions d'application du test asymptotique sont satisfaites ($n \geqslant 30$, et $n_i \geqslant 5$, pour tout i, par exemple), une p-valeur très petite conduit à rejeter l'hypothèse selon laquelle l'échantillon aurait pu être tiré suivant la loi π. Notons que si l'échantillon comporte des valeurs x_j pour lesquelles $\pi\{x_j\} = 0$, on rejette l'hypothèse sans même se poser de question : l'échantillon ne peut en aucun cas avoir été tiré suivant la loi π.

Exercice 2. — Dans le fichier `goodness-of-fit.sci` est définie une fonction `dgoftest`. Examiner sa définition.

Table de quantiles de la statistique de Kolmogorov–Smirnov

Si K_n est la statistique de Kolmogorov–Smirnov correspondant à une taille d'échantillon égale à n, la table donne $k_{n,1-\alpha} \in [0,1]$ tel que $\mathbb{P}\{K_n \leqslant k_{n,1-\alpha}\} = 1-\alpha$.

α \ n	0	1	2	3	4	5	6	7	8	9
0.01	1.0000	0.9950	0.9293	0.8290	0.7342	0.6685	0.6166	0.5758	0.5418	0.5133
0.05	1.0000	0.9750	0.8419	0.7076	0.6239	0.5633	0.5193	0.4834	0.4543	0.4300
0.10	1.0000	0.9500	0.7764	0.6360	0.5652	0.5094	0.4680	0.4361	0.4096	0.3875
0.15	1.0000	0.9250	0.7261	0.5958	0.5248	0.4744	0.4353	0.4050	0.3806	0.3601
0.20	1.0000	0.9000	0.6838	0.5648	0.4927	0.4470	0.4104	0.3815	0.3583	0.3391

α \ n	10	11	12	13	14	15	16	17	18	19
0.01	0.4889	0.4677	0.4490	0.4325	0.4176	0.4042	0.3920	0.3809	0.3706	0.3612
0.05	0.4092	0.3912	0.3754	0.3614	0.3489	0.3376	0.3273	0.3180	0.3094	0.3014
0.10	0.3687	0.3524	0.3381	0.3255	0.3142	0.3040	0.2947	0.2863	0.2785	0.2714
0.15	0.3425	0.3273	0.3141	0.3023	0.2918	0.2823	0.2737	0.2659	0.2587	0.2520
0.20	0.3226	0.3083	0.2957	0.2847	0.2748	0.2658	0.2577	0.2503	0.2436	0.2373

α \ n	20	21	22	23	24	25	26	27	28	29
0.01	0.3524	0.3443	0.3367	0.3295	0.3229	0.3166	0.3106	0.3050	0.2997	0.2947
0.05	0.2941	0.2872	0.2809	0.2749	0.2693	0.2640	0.2591	0.2544	0.2499	0.2457
0.10	0.2647	0.2586	0.2528	0.2475	0.2424	0.2377	0.2332	0.2290	0.2250	0.2212
0.15	0.2459	0.2402	0.2348	0.2298	0.2251	0.2207	0.2166	0.2127	0.2089	0.2054
0.20	0.2315	0.2261	0.2211	0.2164	0.2120	0.2079	0.2040	0.2003	0.1968	0.1934

α \ n	30	31	32	33	34	35	36	37	38	39
0.01	0.2899	0.2853	0.2809	0.2768	0.2728	0.2690	0.2653	0.2618	0.2584	0.2552
0.05	0.2417	0.2379	0.2342	0.2308	0.2274	0.2242	0.2212	0.2183	0.2154	0.2127
0.10	0.2176	0.2141	0.2108	0.2077	0.2047	0.2018	0.1991	0.1965	0.1939	0.1915
0.15	0.2021	0.1989	0.1958	0.1929	0.1901	0.1875	0.1849	0.1825	0.1801	0.1779
0.20	0.1903	0.1873	0.1844	0.1817	0.1791	0.1766	0.1742	0.1718	0.1696	0.1675

α \ n	40	42	44	46	48	50	52	54	56	58
0.01	0.2521	0.2461	0.2406	0.2354	0.2306	0.2260	0.2217	0.2177	0.2138	0.2102
0.05	0.2101	0.2052	0.2006	0.1963	0.1922	0.1884	0.1848	0.1814	0.1782	0.1752
0.10	0.1891	0.1847	0.1805	0.1766	0.1730	0.1696	0.1664	0.1633	0.1604	0.1577
0.15	0.1757	0.1715	0.1677	0.1641	0.1607	0.1575	0.1545	0.1517	0.1490	0.1465
0.20	0.1654	0.1616	0.1579	0.1545	0.1514	0.1484	0.1456	0.1429	0.1404	0.1380

α \ n	60	65	70	75	80	85	90	95	100	105
0.01	0.2067	0.1988	0.1917	0.1853	0.1795	0.1742	0.1694	0.1649	0.1608	0.1570
0.05	0.1723	0.1657	0.1597	0.1544	0.1496	0.1452	0.1412	0.1375	0.1340	0.1308
0.10	0.1551	0.1491	0.1438	0.1390	0.1347	0.1307	0.1271	0.1238	0.1207	0.1178
0.15	0.1441	0.1385	0.1336	0.1291	0.1251	0.1214	0.1181	0.1150	0.1121	0.1094
0.20	0.1357	0.1305	0.1258	0.1216	0.1178	0.1144	0.1112	0.1083	0.1056	0.1031

α \ n	110	120	130	140	150	160	170	180	190	200
0.01	0.1534	0.1470	0.1413	0.1362	0.1316	0.1275	0.1237	0.1203	0.1171	0.1142
0.05	0.1279	0.1225	0.1178	0.1135	0.1097	0.1063	0.1031	0.1003	0.0976	0.0952
0.10	0.1151	0.1103	0.1060	0.1022	0.0988	0.0957	0.0929	0.0903	0.0879	0.0857
0.15	0.1070	0.1025	0.0985	0.0950	0.0918	0.0889	0.0863	0.0839	0.0817	0.0796
0.20	0.1008	0.0965	0.0928	0.0895	0.0865	0.0838	0.0813	0.0790	0.0769	0.0750

Lois de Pearson

Si X est une variable aléatoire suivant la loi du χ^2, de Pearson, à ν degrés de liberté, la table donne, pour α fixé, la valeur $k_{1-\alpha}$ telle que

$$\mathbb{P}\{X \geqslant k_{1-\alpha}\} = \alpha.$$

Ainsi, $k_{1-\alpha}$ est le quantile d'ordre $1-\alpha$ de la loi du χ^2 à ν degrés de liberté.

ν \ α	0.990	0.975	0.950	0.900	0.100	0.050	0.025	0.010	0.001
1	0.0002	0.0010	0.0039	0.0158	2.7055	3.8415	5.0239	6.6349	10.8276
2	0.0201	0.0506	0.1026	0.2107	4.6052	5.9915	7.3778	9.2103	13.8155
3	0.1148	0.2158	0.3518	0.5844	6.2514	7.8147	9.3484	11.3449	16.2662
4	0.2971	0.4844	0.7107	1.0636	7.7794	9.4877	11.1433	13.2767	18.4668
5	0.5543	0.8312	1.1455	1.6103	9.2364	11.0705	12.8325	15.0863	20.5150
6	0.8721	1.2373	1.6354	2.2041	10.6446	12.5916	14.4494	16.8119	22.4577
7	1.2390	1.6899	2.1673	2.8331	12.0170	14.0671	16.0128	18.4753	24.3219
8	1.6465	2.1797	2.7326	3.4895	13.3616	15.5073	17.5345	20.0902	26.1245
9	2.0879	2.7004	3.3251	4.1682	14.6837	16.9190	19.0228	21.6660	27.8772
10	2.5582	3.2470	3.9403	4.8652	15.9872	18.3070	20.4832	23.2093	29.5883
11	3.0535	3.8157	4.5748	5.5778	17.2750	19.6751	21.9200	24.7250	31.2641
12	3.5706	4.4038	5.2260	6.3038	18.5493	21.0261	23.3367	26.2170	32.9095
13	4.1069	5.0088	5.8919	7.0415	19.8119	22.3620	24.7356	27.6882	34.5282
14	4.6604	5.6287	6.5706	7.7895	21.0641	23.6848	26.1189	29.1412	36.1233
15	5.2293	6.2621	7.2609	8.5468	22.3071	24.9958	27.4884	30.5779	37.6973
16	5.8122	6.9077	7.9616	9.3122	23.5418	26.2962	28.8454	31.9999	39.2524
17	6.4078	7.5642	8.6718	10.0852	24.7690	27.5871	30.1910	33.4087	40.7902
18	7.0149	8.2307	9.3905	10.8649	25.9894	28.8693	31.5264	34.8053	42.3124
19	7.6327	8.9065	10.1170	11.6509	27.2036	30.1435	32.8523	36.1909	43.8202
20	8.2604	9.5908	10.8508	12.4426	28.4120	31.4104	34.1696	37.5662	45.3147
21	8.8972	10.2829	11.5913	13.2396	29.6151	32.6706	35.4789	38.9322	46.7970
22	9.5425	10.9823	12.3380	14.0415	30.8133	33.9244	36.7807	40.2894	48.2679
23	10.1957	11.6886	13.0905	14.8480	32.0069	35.1725	38.0756	41.6384	49.7282
24	10.8564	12.4012	13.8484	15.6587	33.1962	36.4150	39.3641	42.9798	51.1786
25	11.5240	13.1197	14.6114	16.4734	34.3816	37.6525	40.6465	44.3141	52.6197
26	12.1981	13.8439	15.3792	17.2919	35.5632	38.8851	41.9232	45.6417	54.0520
27	12.8785	14.5734	16.1514	18.1139	36.7412	40.1133	43.1945	46.9629	55.4760
28	13.5647	15.3079	16.9279	18.9392	37.9159	41.3371	44.4608	48.2782	56.8923
29	14.2565	16.0471	17.7084	19.7677	39.0875	42.5570	45.7223	49.5879	58.3012
30	14.9535	16.7908	18.4927	20.5992	40.2560	43.7730	46.9792	50.8922	59.7031

Lorsque le degré de liberté ν est tel que $\nu > 30$, la variable aléatoire

$$Z = \sqrt{2X} - \sqrt{2\nu - 1}$$

suit approximativement la loi normale centrée réduite.

TRAVAUX PRATIQUES N° 1. — PROCESSUS DE LÉVY RÉELS, QUELQUES CORRECTIONS (SUITE)

3. **Retour sur le cas poissonnien**

Pour la simulation d'une trajectoire d'un processus de Poisson par la loi des événements rares, on a

```
// code Scilab

// 3. Retour sur le cas poissonnien

T = 1; n = 1000; lambda = 1;
time = (T/n)*[0:n]';
Trials = grand(n, 1, "bin", 1, lambda*T/n);
X = cumsum([0; Trials], "r");
clf(); plot(time, X);
xtitle("Processus de Poisson et loi des evenements rares");
halt("*** taper [entree] pour continuer ***\n");
```

Allons un peu plus loin en simulant N trajectoires pour ne conserver que leurs valeurs terminales (il y a du gaspillage de mémoire vive).

```
N = 1000;
Trials = grand(n, N, "bin", 1, lambda*T/n);
// X = cumsum([zeros(1, N); Trials], "r");// inutile d'avoir les N trajectoires
// XT = X(n+1, :)';// puisque seules les N valeurs terminales servent
XT = sum(Trials, "r")';
```

Nous auvons à comparer la distribution observée de la variable terminale avec la loi de Poisson $\mathcal{P}(\lambda T)$ avec le test du χ^2 d'adéquation. Nous limiterons de manière artificielle et maladroite aux valeurs entières comprises entre 0 et la valeur maximale observée $x_{\max}$. La distribution observée sera alors comparée avec la distribution de Poisson tronquée en $x_{\max}$:

$$\pi\{k\} = \mathrm{e}^{-\lambda T}\frac{(\lambda T)^k}{k\,!} \quad \text{pour } 0 \leqslant k < x_{\max}, \quad \text{et} \quad \pi\{x_{\max}\} = 1 - \big(\pi\{0\} + \cdots + \pi\{x_{\max}-1\}\big).$$

Soit

```
// getf("kolmogorov.sci"); // pour plus tard uniquement
getf("goodness-of-fit.sci");

xmax = max(XT);
xtheo = [0:xmax]'; // une colonne
ptheo = zeros(xmax+1, 1); // initialisation
ptheo(1) = exp(-lambda*T); // premier terme
s = 0;
for k = 1:xmax;
    ptheo(k+1) = ptheo(k)*lambda*T/k; // attention aux indices
    s = s+ptheo(k);
```

```
end
ptheo(xmax+1) = 1-s; // reste de la loi de Poisson
// on \'ecrase au passage le ptheo(xmax+1) d\'efini au dernier tour de boucle

dgoftest(XT, xtheo, ptheo); // graphiques et test du chi^2
halt("*** taper [entree] pour continuer ***\n");
```

Le calcul des probabilités successives aura été fait en remarquant que

$$\pi\{0\} = \mathrm{e}^{-\lambda T} \qquad \text{et} \qquad \pi\{k\} = \pi\{k-1\} \times \frac{(\lambda T)^k}{k} \qquad \text{pour } k \geqslant 1,$$

et en prenant garde à l'indexation de Scilab : `ptheo(`$\langle k \rangle$`+1)` $= \pi\{k\}$.

⚄ Nous rappelons que si $U : (\Omega, \mathcal{A}, \mathbb{P}) \to [0,1]$ est une variable aléatoire de loi uniforme sur $[0,1]$ (on supposera la valeur 1 strictement impossible), alors pour $\lambda > 0$, $T = -\ln(1-U)/\lambda$ est une variable aléatoire réelle positive de loi exponentielle de paramètre λ.

⚄ Pour simuler une variable de loi de Poisson de paramètre λ, on peut simuler une suite une file d'attente à temps de record de paramètre λ et compter le nombre de passages au temps $T = 1$ ou encore une file d'attente de temps de record de paramètre 1 observée au temps λT.

```
function n = Poissonrand(lambda)
    n = -1;
    while lambda >= 0;
        n = n+1;
        lambda = lambda+log(1-grand(1, 1, "def"));
    end
endfunction
```

Si on veut conserver la trace des temps de record ou des dates de saut dans $[0,1]$, on pourra taper

```
function [n, scores] = Poissonrand(lambda)
    s = 0; scores = [];
    while s <= 1;
        s = s-log(1-grand(1, 1, "def"))/lambda;
        scores = [scores; s];
    end
    n = size(scores, 1)-1;
    scores(n+1) = 1; // ignorer ce qu'il se passe apr\`es $t = 1$
endfunction
```

⚄ Nous ne tracerons pas les trajectoires implicitement ou non définies par la chaque appel de la fonction `Poissonrand`. Regardons simplement ce qu'il se passe au niveau de la variable terminale.

```
clear n lambda T X XT ptheo xtheo;

N = 1000; lambda = 1; T = 1;
for k = 1:N; X(k) = Poissonrand(lambda*T); end // X est un vecteur colonne
xmax = max(X); xtheo = [0:xmax]';
ptheo = zeros(xmax+1, 1); ptheo(1) = exp(-lambda*T); s = 0;
for k = 1:xmax;
    ptheo(k+1) = ptheo(k)*lambda*T/k;
    s = s+ptheo(k);
end
ptheo(xmax+1) = 1-s; // reste de la loi de Poisson
```

```
dgoftest(X, xtheo, ptheo); // graphiques et test du chi^2
halt("*** taper [entree] pour continuer ***\n");
```

Remarque. — Le choix de $x_{\max}$ aurait dû être fait plus judicieusement dans tout ce qui précède. Nous aurions pu le choisir de sorte qu'il soit le plus grand entier k tel que $N \times \pi\{k\} \geqslant 5$ (la valeur 5 provenant des condition d'applicabilité du test [asymptotique] du χ^2). Par exemple le code précédent aurait été écrit

```
N = 1000; lambda = 1; T = 1;

clear ptheo; ptheo(1) = exp(-lambda*T); s = 0; xmax = 0;
while N*ptheo(xmax+1) >= 5;
    xmax = xmax+1;
    ptheo(xmax+1) = ptheo(xmax)*lambda*T/xmax;
    s = s+ptheo(xmax);
end
ptheo(xmax+1) = 1-s; xtheo = [0:xmax]

for k = 1:N; X(k) = min(Poissonrand(lambda*T), xmax); end

dgoftest(X, xtheo, ptheo);
halt("*** taper [entree] pour continuer ***\n");
```

4. **Retour sur le cas brownien**

Considérons la marche aléatoire simple sur $\mathbb{Z}$. Ses accroissements sont indépendants de loi $(\delta_{\{-1\}} + \delta_{\{-1\}})/2$ de moyenne 0 et de variance 1.

```
// 4. Retour sur le cas brownien

clear();
getf("kolmogorov.sci");
getf("goodness-of-fit.sci");

T = 1; n = 1000; time = (T/n)*[0:n]';
dX = sqrt(T/n)*(2*grand(n, 1, "bin", 1, 0.5));
X = cumsum([0; dX], "r");

clf(); plot(time, X);
halt("*** taper [entree] pour continuer ***\n");
```

Les accroissements générés par la machine sont supposés avoir une propriété statistitque similaire à l'indépendance. Notamment les théorèmes asymptotiques classiques doivent pouvoir être approximativement vérifiés (cela fait partie des tests pratiqués par les spécialistes des générateurs de nombres aléatoires). Nous devons donc avoir un théorème central limite. Vérifions-le avec le test de Kolmogorov–Smirnov sur $N = 100$ valeurs terminales (procédure `cgoftest` définie dans `goodness-of-fit.sci`).

```
N = 100; // 1000 serait peut-\^etre trop pour kolmogorovcdf(.)
dX = sqrt(T/n)*(2*grand(n, N, "bin", 1, 0.5));
// X = cumsum([zeros(1, N); dX], "r"); // inutile d'avoir les N trajectoires
// XT = X(n+1,:)'; // puisque seules les N valeurs terminales servent
XT = sum(dX, "r")';

deff("y = F(x)", "y = cdfnor(""PQ"", x, 0, sqrt(T))"); cgoftest(XT, F);
halt("*** taper [entree] pour continuer ***\n");
```

TRAVAUX PRATIQUES N° 2. — INTÉGRALES STOCHASTIQUES

1. La méthode standard

Il n'y a pas beaucoup de choix pour calculer de manière approchée une intégrale stochastique $\int_0^T H_t \, \mathrm{d}X_t$. On dispose d'une subdivision du temps $0 = t_1 < \cdots < t_{n+1} = T$ (indexation cohérente avec SCILAB) fixée pour une raison ou une autre (nature ou simulation du processus X, dates de relevés réels, ...). Nous savons de plus que la non nullité éventuelle de la variation quadratique $[X, X]$ implique que les schémas numériques classiques (rectangles, trapèzes) donnent lieu à des intégrales stochastiques différentes.

Nous avons donc pour l'intégrale de Itô

$$(H \cdot X)_T = \int_0^T H_t \, \mathrm{d}X_t \approx \sum_{i=1}^{n} H(t_i) \times \big(X(t_{i+1}) - X(t_i)\big)$$

et pour l'intégrale de Stratonovich

$$(H \circ X)_T = \int_0^T H_t \circ \mathrm{d}X_t \approx \sum_{i=1}^{n} \frac{1}{2}\big(H(t_i) + H(t_{i+1})\big) \times \big(X(t_{i+1}) - X(t_i)\big)$$

Ce qu'on peut mettre en œuvre de manières diverses, par exemple

```
// code Scilab

T = 1; n = 1000;
time = (T/n)*[0:n]';
dX = grand(n, 1, "nor", 0, sqrt(T/n)); // accroissements browniens standard
X = cumsum([0; dX], "r"); // brownien issu de 0. Noter que dX(i) = X(i+1)-X(i)

H = 2*X; // H_t=2X_{t-} (voir formules d'int\'egration par parties)

HdotX = zeros(n+1, 1);
HcircX = zeros(n+1, 1);
HdotX(1) = 0;
HcircX(1) = 0;
for i = 1:n;
    HdotX(i+1) = HdotX(i)+H(i)*(X(i+1)-X(i));
    HcircX(i+1) = HcircX(i)+(H(i)+H(i+1))/2*(X(i+1)-X(i));
end
```

Les boucles, bien que très déconseillées en SCILAB, assurent un bon contrôle des indices et la cohérence avec les définitions.

EXERCICE 1. — ⚄ Saisir et exécuter ce morceau de programme.

⚄ Vérifier que les contraintes suivantes sont satisfaites :

- Les « intégrales stochastiques » ont une valeur initiale nulle.
- Elles ont le même ensemble d'indices que le processus X.
- L'intégrale de Itô définit une martingale lorsque X est une martingale.

⚄ Représenter sur un même graphique les trois processus X, $H \cdot X$ et $H \circ X$.

```
clf(); plot(time, [X, HdotX, HcircX]);
legend(["processus X"; "processus H.X"; "processus HoX"]);
halt("*** taper [entree] pour continuer ***\n");
```

Exercice 2. — ⚄ Définir une fonction Scilab `Ito(`$\langle H\rangle$`, `$\langle X\rangle$`)` retournant le processus $H \cdot X$. On s'imposera pour contrainte de ne pas employer de boucle et de faire dépendre la taille du vecteur (colonne) de sortie de celles des vecteurs (colonne) d'entrée en avertissant d'une incohérence éventuelle.

⚄ Définir de même une fonction Scilab `Strato(`$\langle H\rangle$`, `$\langle X\rangle$`)` respectant des contraintes similaires.

Correction. — ⚄ Pour l'intégrale de Itô :

```
function Y = Ito(H, X)
    // local m n
    n = size(H, 1); m = size(X,1);
    if n+1 == m then
        mprintf("H et X de tailles compatibles ");
        mprintf("pour l''integrale de Ito.\n");
    elseif n >= m then
        mprintf("des valeurs de H seront ignorees ");
        mprintf("pour l''integrale de Ito.\n");
        n = m-1;
    else
        mprintf("des valeurs de X seront ignorees ");
        mprintf("pour l''integrale de Ito.\n");
    end
    Y = [0; cumsum(H(1:n).*(X(2:n+1)-X(1:n)), "r")];
endfunction
```

⚄ Pour l'intégrale de Stratonovich :

```
function Y = Strato(H, X)
    // local m n
    n = size(H, 1); m = size(X,1);
    if n == m then
    mprintf("H et X de tailles compatibles ");
    mprintf("pour l''integrale de Stratonovich.\n");
    elseif n > m then
        mprintf("des valeurs de H seront ignorees ");
        mprintf("pour l''integrale de Stratonovich.\n");
        n = m;
    else
        mprintf("des valeurs de X seront ignorees ");
        mprintf("pour l''integrale de Stratonovich.\n");
    end
    Y = [0; cumsum(0.5*(H(1:n-1)+H(2:n)).*(X(2:n)-X(1:n-1)), "r")];
endfunction
```

On vérifie

```
clf(); plot(time, [X, Ito(H, X), Strato(H, X)]);
legend(["processus X"; "processus Ito(H, X)"; "processus Strato(H, X)"]);
halt("*** taper [entree] pour continuer ***\n");
```

2. **Formules de changement de variables**

Nous rappelons que si $X = (X_t)_{t\geqslant 0}$ est une semi-martingale réelle et $f : \mathbb{R} \to \mathbb{R}$ est une fonction de classe C^2, alors la formule du changement de variable, ou formule de Itô, s'écrit

$$f(X_T) - f(X_0) = \int_0^T f'(X_{t-})\,\mathrm{d}X_t + \frac{1}{2}\int_0^T f''(X_{t-})\,\mathrm{d}[X,X]_t, \qquad T \geqslant 0.$$

où $[X,X]$ est la variation quadratique de X qui est égale à sa projection duale prévisible $\langle X,X\rangle$ lorsque X est à trajectoires continues, et simplement $[X,X]_t = t$ lorsque X est un mouvement brownien linéaire. Aux difficultés techniques près, cette formule repose simplement sur la définition de l'intégrale stochastique comme limite de sommes de Riemann et sur la formule de Taylor. Elle doit pouvoir se retrouver sans trop de difficultés avec ces approximations. Et il devrait en être de même de la formule de changement de variables, qui est la formule ordinaire, dans le cadre de l'intégrale de Stratonovich.

Exercice 3. — ⚄ Reconnaître dans les exercices précédents les formules de changement de variables correspondantes. Comparer graphiquement les processus $f(X) - f(X_0)$, l'intégrale de Itô $f'(X)\cdot X$, le second membre de la formule de Itô et l'intégrale de Stratonovich $f'(X)\circ X$ pour la fonction f considérée.

⚄ Pour $\langle G\rangle$ un vecteur colonne représentant un processus quelconque, à quoi correspondent les résultats de `Ito(`$\langle G\rangle$`, time)` et `Strato(`$\langle G\rangle$`, time)` ?

⚄ Mettre à l'épreuve les approximations des intégrales stochastiques pour la fonction trigonométrique $x \mapsto \sin(2\pi x/T)$.

⚄ Mettre à l'épreuve les approximations des intégrales stochastiques pour les fonctions exponentielles $x \mapsto \mathrm{e}^x$ et $x \mapsto \mathrm{e}^{-x}$.

⚄ Qu'obtient-on pour $x \mapsto |x|$, $x \mapsto x^+ = \max(x,0)$, et $x \mapsto -x^- = \min(x,0)$? *Le temps local en 0 du mouvement brownien vient de faire son apparition...*

Remarque. — Le temps local du mouvement brownien en 0 est défini comme le processus croissant L^0 tel que $X_t^+ - X_0^+ = \int_0^t \mathbb{1}_{\{X_s>0\}}\,\mathrm{d}X_s + \frac{1}{2}L_t^0$. Il apparaît dans les formules de Itô–Tanaka liées à X^-, $|X|$, ... et définit une mesure positive non nulle portée par les zéros du browniens qui sont de mesure de Lebesgue nulle. L'homogénéité spatiale du mouvement brownien amène à considérer le temps local L^x en $x \in \mathbb{R}$ qui est un processus similaire à L^0 (le même quitte à changer la valeur initiale du processus).

⚄ Déduire des formules d'approximation des intégrales stochastiques une approximation du temps local L^0. Lorsque X est approché à l'aide de la marche aléatoire simple sur $\varepsilon\mathbb{Z}$, exprimer cette approximation en terme de *nombre de montées* ou de *nombre de descentes* du processus autour de $x = 0$.

Correction. — ⚄ Ici on a bien sûr $f(x) = x^2$.

```
// le processus

T = 1; n = 1000;
time = (T/n)*[0:n]';
dX = grand(n, 1, "nor", 0, sqrt(T/n));
X = cumsum([0; dX], "r");

// f(x) = x^2 (car f(x) = x est trop simple\dots)
```

```
fX = X.^2;
H = 2*X;
G = 2*ones(size(X));
I = Ito(H, X);
Z = time; // 0.5*Ito(G, time);
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = X^2");
legend(["f(X)-f(Xo)"; "Int 2X dX"; "Int 2X dX + t"; "Strato 2X dX"]);
halt("*** taper [entree] pour continuer ***\n");
```

⚄ `Ito(`$\langle G \rangle$`, time)` est une simple somme de Riemann (méthode des rectangles) approchant $\int_0^T G_t \, \mathrm{d}t$, tandis que `Strato(`$\langle G \rangle$`, time)` correspond à l'approximation par la méthode des trapèzes.

⚄

```
// f(x) = sin(2pi*x/T)

fX = sin(2*%pi*X/T);
H = 2*%pi/T*cos(2*%pi*X/T);
G = -(2*%pi/T)^2*sin(2*%pi*X/T);
I = Ito(H, X);
Z = 0.5*Strato(G, time);// mieux que Ito ?
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = sin(2pi*X/T)");
legend(["f(X)-f(Xo)"; "Int f''(X) dX"; ..
"Int f''(X) dX + 1/2Int f''''(X) dt"; "Strato f''(X) dX"]);
halt("*** taper [entree] pour continuer ***\n");
```

⚄

```
// f(x) = exp(x)

fX = exp(X);
H = fX;
G = fX;
I = Ito(H, X);
Z = 0.5*Ito(G, time);
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = exp(X)");
legend(["f(X)-f(Xo)"; "Int f''(X) dX"; ..
"Int f''(X) dX + 1/2Int f''''(X) dt"; "Strato f''(X) dX"]);
halt("*** taper [entree] pour continuer ***\n");
```

⚄

```
// f(x) = exp(-X)
```

```
fX = exp(-X);
H = -fX;
G = fX;
I = Ito(H, X);
Z = 0.5*Ito(G, time);
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = exp(-X)");
legend(["f(X)-f(Xo)"; "Int f''(X) dX"; ..
"Int f''(X) dX + 1/2Int f''''(X) dt"; "Strato f''(X) dX"]);
halt("*** taper [entree] pour continuer ***\n");
```

```
// f(x) = exp(-X)

fX = abs(X);
for i = 1:n+1;
    if X(i) < 0 then H(i) = -1;
    elseif X(i) > 0 then H(i) = 1;
    else H(i) = 0; end
end
G = zeros(n+1, 1);
I = Ito(H, X);
Z = 0.5*Ito(G, time);
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = |X|");
legend(["f(X)-f(Xo)"; "Int f''(X) dX"; ..
"Int f''(X) dX + 1/2Int f''''(X) dt"; "Strato f''(X) dX"]);
halt("*** taper [entree] pour continuer ***\n");
```

Attention ! — L'intégrale de Itô est égale au second membre de la formule correspondante, donc les courbes (rouge et verte) correspondantes coïncident. Curieusement, l'intégrale de Stratonovich continue à donner des résultats convenables, elle semble ignorer le temps local.

```
// f(x) = x+
for i = 1:n+1;
    if X(i) < 0 then fX(i) = 0; H(i) = 0;
    else fX(i) = X(i); H(i) = 1; end
end
G = zeros(n+1, 1);
I = Ito(H, X);
Z = 0.5*Ito(G, time);
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = X+");
legend(["f(X)-f(Xo)"; "Int f''(X) dX"; ..
"Int f''(X) dX + 1/2Int f''''(X) dt"; "Strato f''(X) dX"]);
halt("*** taper [entree] pour continuer ***\n");
```

Y a-t'il des différences lorsque la condition pour définir $\langle H \rangle$ est modifiée ? Non, nous sommes dans un cadre discret.

```
// f(x) = -x-

for i = 1:n+1;
    if X(i) > 0 then fX(i) = 0; H(i) = 0;
    else fX(i) = X(i); H(i) = -1; end
end
G = zeros(n+1, 1);
I = Ito(H, X);
Z = 0.5*Ito(G, time);
S = Strato(H, X);
clf(); plot(time, [fX-fX(1), I, I+Z, S]);
xtitle("X mouvement brownien, f(X) = -X-");
legend(["f(X)-f(Xo)"; "Int f''(X) dX"; ..
"Int f''(X) dX + 1/2Int f''''(X) dt"; "Strato f''(X) dX"]);
halt("*** taper [entree] pour continuer ***\n");
```