Processus stochastique : simulation d'un mouvement brownien

Benjamin Dosse

1 Idée générale

La simulation d'un mouvement brownien peut suivre au moins trois voies différentes :

— Exploitation de la définition du mouvement brownien ;

— Théorème de Donsker ;

— Théorème de construction par la base de Schauder.

Dans chacun de ces cas, nous utilisons des variables aléatoires *iid* selon une loi normale centrée réduite, sauf dans la construction utilisant la définition du mouvement brownien.

2 Construction

On utilise la transformation de Box-Müller pour produire des nombres distribués selon une loi normale centrée réduite à partir de nombres distribués selon une loi uniforme.

Algorithm 1 Transformation de Box-Müller

function RAND_NORMAL(#)
Require: $N \in \mathbb{N}$
Require: $U_1, U_2 \overset{\text{iid}}{\sim} \mathcal{U}[0, N]$
Ensure: $X, Y \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$
$X \leftarrow 0$
$Y \leftarrow 0$
while $(X^2 + Y^2 > 1) \vee (X = 0) \vee (Y = 0)$ **do**
$X \leftarrow 2\frac{U_1}{N} - 1$
$Y \leftarrow 2\frac{U_2}{N} - 1$
end while
$X \leftarrow X\sqrt{\frac{-2\log(X^2+Y^2)}{X^2+Y^2}}$
$Y \leftarrow Y\sqrt{\frac{-2\log(X^2+Y^2)}{X^2+Y^2}}$
return X, Y
end function

En langage C, le nombre N indiqué dans l'algorithme est fixé par une constante `RAND_MAX` définie dans le fichier `stdlib.h`. Il est assurée que cette constante vaut au moins 32767.

2.1 Base de Schauder

La construction par la base de Schauder utilise explicitement des variables aléatoires centrées réduites. En considérant les notations du théorème, on devrait générer au plus

$$n = j2^j + 1$$

nombres distribués indépendamment selon une loi gaussienne standard pour calculer l'image au temps t de la trajectoire du mouvement brownien $B(t)$.

Ainsi, si on désigne par `rand_normal(#)` une fonction qui ne prend aucun argument et qui rend un nombre généré selon l'algorithme de Box-Müller, et si on désigne par `schauder_basis(j, k, t)` une fonction qui prend en entrée trois arguments et qui rend l'élément d'indice j, k de la base de Schauder évalué au temps t, alors l'algorithme suivant retourne la position au temps t sur la trajectoire du mouvement $B(t)$.

Algorithm 2 Trajectoire brownienne (Schauder)

function SCHAUDER_BROWNIAN_MOTION(p, t)
Require: $p \in \mathbb{N}$
Require: $t \in [0, 1]$
Ensure: $X = B(t, \omega)$
  $X \leftarrow 0$
  for $j \in \{0, \ldots, p\}$ **do**
    for $k \in \{0, \ldots, 2^j - 1\}$ **do**
      $X \leftarrow X +$ `rand_normal(#)` + `schauder_basis(`j, k, t`)`
    end for
  end for
  $X \leftarrow X + t$ `rand_normal(#)`
  return X
end function

Dans ce qui précède, l'entrée p contrôle la *« précision »* du calcul (la somme portant sur les j va ainsi de 0 à p), et $B(t, \omega)$ désigne la trajectoire calculée par l'algorithme.

2.2 Théorème de Donsker

Les variables aléatoires du théorème de Donsker peuvent être distribuées identiquement et indépendamment selon une loi gaussienne standard. Une telle construction peut alors suivre l'algorithme suivant :

Algorithm 3 Tajectoire brownienne (Donsker)

function DONSKER_BROWNIAN_MOTION(n, t)
Require: $n \in \mathbb{N}$
Require: $t \in [0, 1]$
Ensure: $X = B(t, \omega)$
  $X \leftarrow 0$
  Limit $\leftarrow nt$
  FloorLimit $\leftarrow$ `Floor(`Limit`)`
  for $j \in \{0, \ldots,$ FloorLimit$\}$ **do**
    $X \leftarrow X +$ `rand_normal(#)`
  end for
  $X \leftarrow X +$ (Limit $-$ FloorLimit)`rand_normal(#)`
  $X \leftarrow \dfrac{X}{\sqrt{n}}$
  return X
end function

On s'assure alors, par le théorème de Donsker, que le comportement limite de l'algorithme ci-dessus rend bien une trajectoire brownienne.

On peut se passer de l'utilisation de variables aléatoires gaussiennes : une suite de variables aléatoires *iid* selon une loi uniforme $\mathcal{U}[0, N]$ convient parfaitement après standardisation ; cette piste n'a pas encore été exploité.

2.2.1 Trajectoire brownienne

L'obtention d'une trajectoire est obtenue en considérant un nombre d'étape $N \in \mathbb{N}_0$ et une des méthodes citée ci-avant : on évalue la fonction choisie en chaque $t_j = \frac{j}{N}$ ($j \in \{0, \dots, N\}$), on obtient alors la suite $(B(t_0) = 0, B(t_1), \dots, B(t_N) = B(1))$, et il suffit d'interpoller linéairement entre $B(t_j)$ et $B(t_{j+1})$ quel que soit $j \in \{0, \dots, N-1\}$ pour obtenir la trajectoire désirée.

Algorithm 4 Trajectoire sur le segment $[0, 1]$.

Require: $N \in \mathbb{N}$
Require: $n \in \mathbb{N}$
  $X \leftarrow 0$
  for $j \in \{0, \dots, N\}$ **do**
    $X \leftarrow$ `brownian_motion`$(n, j/N)$
    Print « TEMPS : » j/N
    Print « POSITION : » X
  end for

La fonction `brownian_motion(n,t)` désigne l'une des fonctions définies dans les Algorithme 2 et Algorithme 3.

2.3 Définition du mouvement brownien

On sait que si $B(t)$ désigne un mouvement brownien standard, alors

$$B(t) - B(s) \sim \mathcal{N}(0, t - s),$$

i.e. si on considère $\Delta t > 0$, on a $B(t + \Delta t) - B(t) \sim \mathcal{N}(0, \Delta t)$. Ainsi, on peut noter

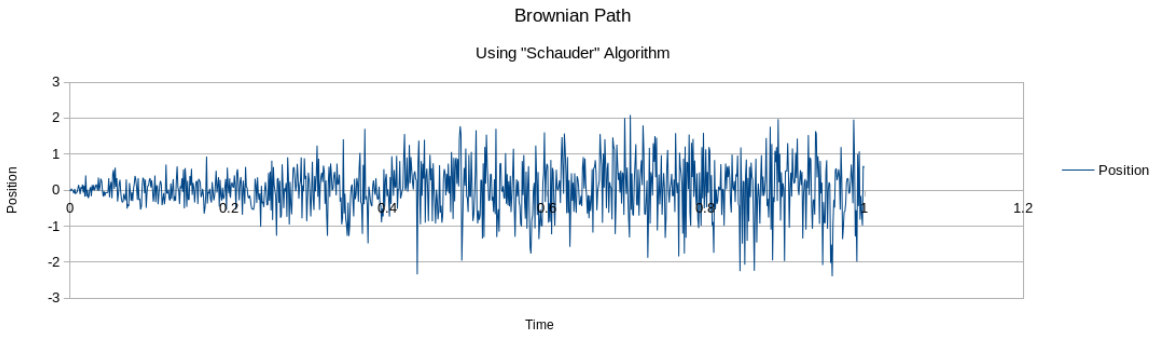
$$B(t + \Delta t) = B(t) + \delta,$$

où $\delta \sim \mathcal{N}(0, \Delta t)$. L'algorithme suivant illustre une construction plutôt simple à implémenter :

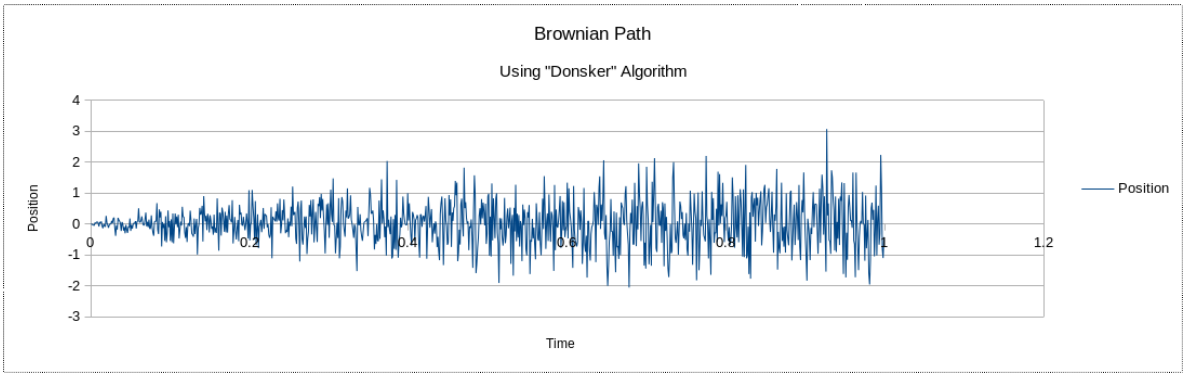
Algorithm 5 Trajectoire brownienne (définition)

Require: $\Delta t \in]0, \infty[$
  $X \leftarrow 0$
  $t \leftarrow 0$
  while $t < 1$ **do**
    $X \leftarrow X + \sqrt{\Delta t}$ `rand_normal(#)`
    Print « TIME : » t
    Print « POSITION : » X
    $t \leftarrow t + \Delta t$
  end while

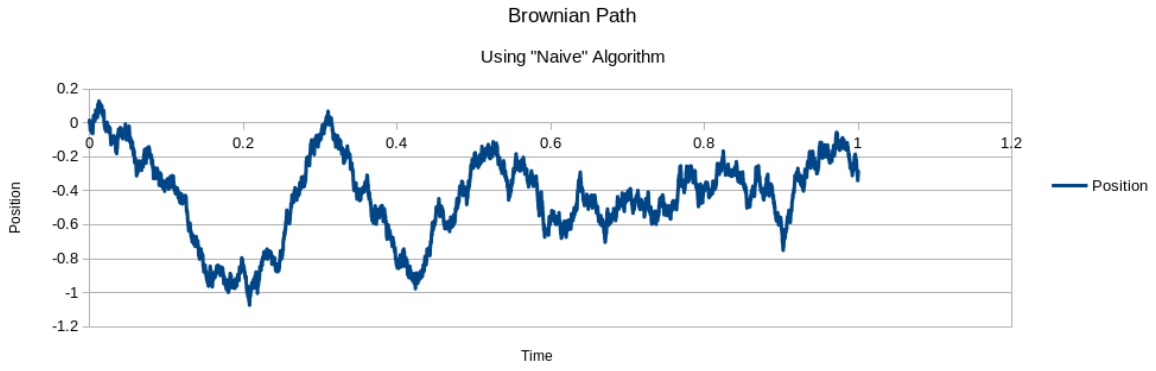
On sait que `rand_normal(#)` suit une loi gaussienne standard, donc $\sqrt{\Delta t}$ `rand_normal(#)` suit une loi gaussienne centrée de variance Δt.



(a) Algorithme 2, $n = 16$.



(b) Algorithme 3, $n = 10000$.



(c) Algorithme 5, $n = 10000$.

FIGURE 1 – Trajectoires browniennes.

3 Difficultés

Comme il a été discuté en fin de cours, les trajectoires générées ne ressemblent pas tout à fait à des trajectoires browniennes : elles restent bien trop centrées en zéro, et ont une variance très (trop ?) importante entre deux temps t_j et t_{j+1}.