exitbm: a library for simulating Brownian motion's exit times and positions from simple domains

Antoine Lejay

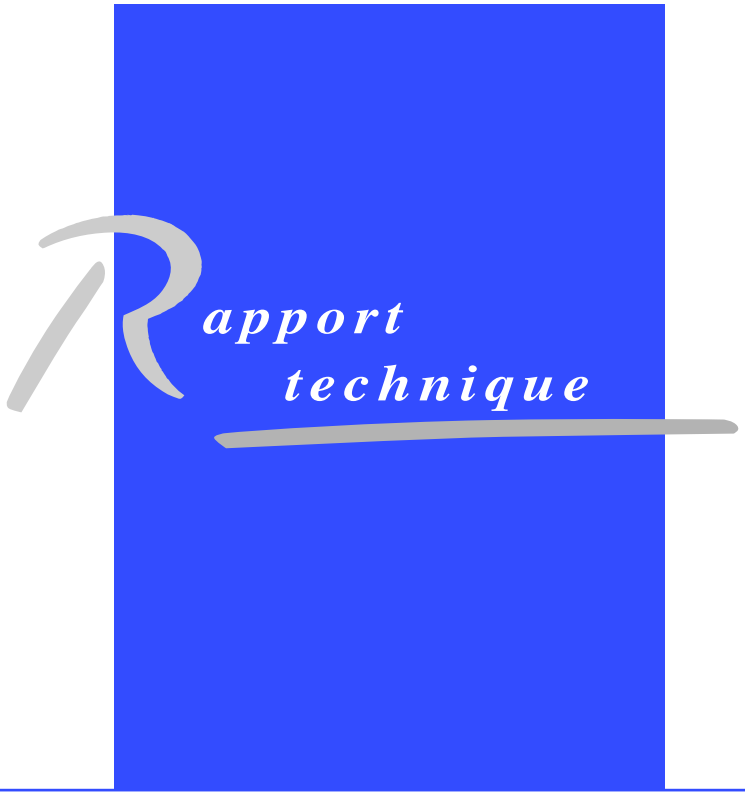
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

exitbm:
a library for simulating Brownian motion's exit times and positions from simple domains

Antoine Lejay

N° 0402

January 2011



ISSN 0249-0803 ISRN INRIA/RT--0402--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

exitbm: a library for simulating Brownian motion's exit times and positions from simple domains

Antoine Lejay

Theme :

Équipes-Projets Tosca

Rapport technique n° 0402 — January 2011 — 24 pages

Abstract: This C library aims at computing and simulating various quantities and random variables related to where and when a the Brownian motion hit the boundary of an interval, a square or a rectangle. We present here the algorithms used in this library.

Key-words: Brownian motion, Skew Brownian motion, exit time, exit position, heat equation, Laplace operator

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

`exitbm`: une bibliothèque pour simuler les temps et positions de sortie de domaines simples du mouvement brownien

Résumé : Cette bibliothèque en langage `C` calcule et simule diverses quantités et variables aléatoire reliées au temps et positions de sortie pour un mouvement brownien d'un intervalle, d'un carré ou d'un rectangle. Nous présentons ici les algorithmes utilisés dans cette bibliothèque.

Mots-clés : Mouvement brownien, mouvement brownien biaisé, temps de sortie, position de sortie, équation de la chaleur, opérateur de Laplace

Contents

1. Introduction

This numerical library provides a way to compute several quantities related to the Brownian motion on an interval (first exit time, ...), a square or a rectangle and to generated the corresponding random variates.

The motivation beyond this library is to provide all the necessary basic bricks to implement the methods of *random walk on squares* and *random walk on rectangle*, as a substitute to the method of *random walk on spheres*. These Monte Carlo methods aim at solving for example Dirichlet problems, to compute the density of the first exit time and position from a domain as well as other quantities (some effective coefficients [8, 28], the first eigenvalue and eigenfunction of the Laplace operator [17, 18],...).

The random walk on spheres was introduced by E.M. Muller in 1956 [25] and is a very efficient method to solve the Dirichlet problem (see for example [14, 21, 22, 27, 29] for analysis, applications and extensions). However, it is more difficult to deal with the exit time. With the random walk on squares and rectangles, the joint distribution of the first exit time and exit position is exactly simulated. By an appropriate choice of squares and rectangles, one may also get the exact exit time and position from a domain with polygonal boundary. This could be important when the exit

time is required with a good accuracy as for the Monte Carlo computation of the first eigenvalue of the Laplace operator [17, 18]. The random walk on squares was introduced by O. Faure in his Ph.D. thesis [13] and by G.N. Milstein and M.V. Tretyakov [23, 24] (see [20] for variants and extensions). The random walk on rectangles was introduced in [9, 10].

The key of these approaches is to use conditioning to simulate random variables in dimension one. The library may then also be used for simulating diffusion processes with discontinuous coefficients [19] or other simulation problems related to the one-dimensional Brownian motion or for diffusion simulation on a graph [15].

This library has been split in several parts:

- `exitbm` One-dimensional distributions for exit time, positions, ...
- `exitbm_fz` One-dimensional distributions for exit time, positions, ... when the starting point is in the middle of the interval (relies on tabulated values).
- `exitbm_fhor` Exit time and position from an interval.
- `exitbm_square` Exit time and position from a square.
- `exitbm_rectangle` Exit time and position from a rectangle.
- `exitbm_test` For testing the functions.
- `exitbm_benchmark` For testing the speed of the implementation.

The dependencies between the headers (except for `exitbm_test` and `exitbm_benchmark` which depend on all the headers files) in shown in Figure 1.

Dependencies: This library is written in standard `C` and requires the Gnu Scientific Library (`GSL`, [2]). However, it may be easily interfaced with other scientific library providing both random number generators and some common functions such as the error function `erf`, ...

This library also comes with files used to generate the benchmarks and the figures in the documentations. They rely on `Ruby` [4] and related tools and libraries, namely `rake` [6].

The figures of this documentation are plotted with `R` [5]. This document presents mainly the mathematical part of the library. The code is documented using `Doxygen` [3], which contains a description of the arguments of the functions. Again, none of these softwares are mandatory to get the library working.

Licence: This library is distributed under the Free Software license *CeCILL* [1].

Web site: This library is hosted at ⟨`http://exitbm.gforge.inria.fr`⟩.

Version: This document describes the version 2.0 of the library released in January 2011.

Acknowledgement. This work has been motivated from collaborations with Fabien Campillo, Madalina Deaconu, Miguel Martinez, Sylvain Maire and Samih Zein and the author wishes to thank them for interesting discussions about simulation issues. Some of the motivations for the development of this library also come from my participation to the Groupement MOMAS (funded by Andra, BRGM, CEA, CNRS and IRSN) which granted some financial support.

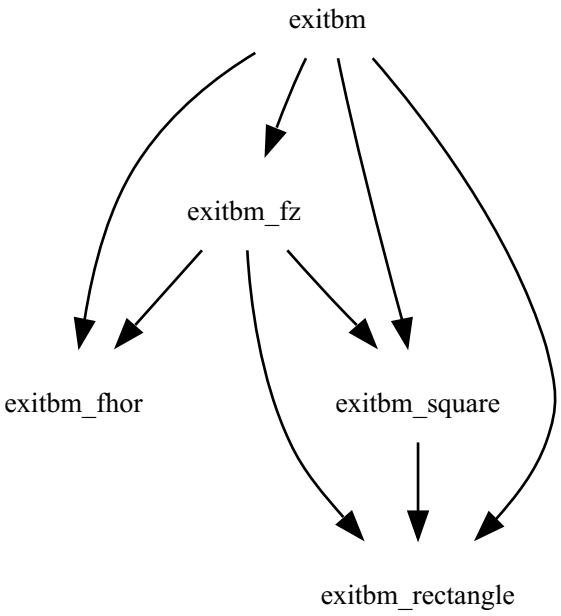


Figure 1: Dependencies between the header files.

2. Exit time and position for the one-dimensional Brownian motion

Let B be a one-dimensional Brownian motion. Its first exit time from $[-1, 1]$ is defined by

$$\tau = \inf\{t \geq 0 \mid B_t \notin [-1, 1]\}.$$

The computations presented in this Section may be applied to any intervals: See Appendix A.1.

Almost all of the expressions presented here implies infinite series. Yet a few term are sufficient to get an accurate approximation in almost all the cases.

Remark 1. The code uses formula that may differ from the one presented here (transformation of the error function erf to the complementary error function, trigonometric transforms, ...).

2.1. Probability that the Brownian motion reaches the right endpoint before the left endpoint

Let us denote by $\mathbb{P}_x$ the distribution of the Brownian motion with $B_0 = x$. The probability that B reaches 1 before -1 is given by

$$\mathbb{P}_x[B_\tau = 1] = S(x) = \frac{x+1}{2}.$$

The function $S(x)$ is called the *scale function*.

Numerical aspects: Due to its simplicity, there is no function for computing the scale function.

2.2. Density of the killed Brownian motion

The density of the killed Brownian motion is the fundamental solution of the parabolic PDE

$$\begin{cases} \dfrac{\partial p(t,x,y)}{\partial t} + \frac{1}{2}\triangle p(t,x,y) = 0 \text{ on } \mathbb{R}_+ \times (-1,1), \\ p(t,x,y) \xrightarrow[t\to 0]{} \delta_y(x), \\ p(t,x,-1) = p(t,x,1) = 0, \ t > 0, \ x \in (-1,1). \end{cases}$$

Analytical expressions hold for this density [7, 23] (see also for example [30] for some explanations). The first one comes from the *method of images*:

$$p(t,x,y) = \frac{1}{\sqrt{2\pi t}} \sum_{n=-\infty}^{+\infty} \left(\exp\left(-\frac{(x-y-4n)^2}{2t} \right) - \exp\left(-\frac{(x+y-2-4n)^2}{2t} \right) \right). \tag{1}$$

The second comes from the spectral decomposition of the Laplace operator and

$$p(t,x,y) = \sum_{n=1}^{+\infty} \exp\left(-\frac{n^2\pi^2}{8} t \right) \sin\left(\frac{n\pi}{2}(x+1) \right) \sin\left(\frac{n\pi}{2}(y+1) \right). \tag{2}$$

The density of the killed Brownian motion has the following probabilistic representation

$$\int_{-1}^{1} p(t,x,y) g(y) \, \mathrm{d}y = \mathbb{E}_x[f(B_t); t < \tau] \tag{3}$$

for any bounded, measurable function g on $[-1,1]$.

The integral

$$P(t,x,y) = \mathbb{P}_x[B_t < y | t < \tau] = \int_{-1}^{y} p(t,x,z) \, \mathrm{d}z$$

is then given by

$$P(t,x,y) = \frac{1}{2\sqrt{t}} \sum_{n=-\infty}^{+\infty} \left(\operatorname{erf}\left(\frac{x+1-4n}{\sqrt{2t}} \right) - \operatorname{erf}\left(\frac{x-y-4n}{\sqrt{2t}} \right) + \operatorname{erf}\left(\frac{x-3-4n}{\sqrt{2t}} \right) - \operatorname{erf}\left(\frac{x+y-2-4n}{\sqrt{2t}} \right) \right)$$

where erf stands for the following function

$$\operatorname{erf}(y) = \frac{2}{\sqrt{\pi}} \int_0^y \exp\left(-z^2\right) \, \mathrm{d}z.$$

With the spectral decomposition,

$$P(t,x,y) = \sum_{n=1}^{+\infty} \frac{2}{n\pi} \exp\left(-\frac{\pi n^2}{8t}\right) \sin\left(n\frac{(x+1)\pi}{2}\right)\left(1-\cos\left(n\frac{(y+1)\pi}{2}\right)\right). \tag{4}$$

Since $\mathbb{P}_x[B_\tau < y | t < \tau] = (1-F(t,x))P(t,x,y)$, a realization of B_τ given $\{t<\tau\}$ may then be obtained by solving $(1-F(t,x))P(t,x,\overline{Z}) = \overline{U}$ for a realization $\overline{U}$ of a uniform random variable on $[0,1]$.

It is also possible to simulate B_τ given $\{t < T < \tau\}$, since

$$\mathbb{E}_x[g(B_t), t<T<\tau] = \mathbb{E}_x[g(B_t)\mathbb{P}_x[\tau > T|\mathcal{F}_t], t<\tau] = \mathbb{E}_x[g(B_t)\mathbb{P}_{B_T}[\tau > T-t], t<\tau],$$

where $(\mathcal{F}_t)_{t\geq 0}$ is the filtration generated by the Brownian motion. Then the density of B_t given $\{t<T<\tau\}$ is

$$p^\star(t,x,y,T) = p(t,x,y)\frac{1-F(T-t,y)}{1-F(T,x)} = \frac{p(t,x,y)}{1-F(t,y)}\frac{1-F(t,y)}{1-F(T,x)}(1-F(T-t,y)). \tag{5}$$

Numerical aspects: Formula (2) is suitable for "large" values of t while (1) is suitable for "small" values of t. The density is computed by `d_pos` which uses one of the two formula according to the value of t. The function `id_pos` computes $P(t,x,y)$. The function `idd_pos` computes both $p(t,x,y)$ and $P(t,x,y)$.

The method `q_pos_gne` finds y solution to $(1-F(t,x))P(t,x,y) = u$, $u \in (0,1)$.

The method `r_pos_gne` gives a realization of B_t given $\{\tau < t\}$. The method `r_pos_getg` gives a realization of B_t given $\{\tau > T\}$ with $T > t$. This method relies on Formula (5) and the rejection algorithm given in [11, § 3.3, p. 47].

In `exitbm_fz`, the method `r_pos_gne_fz` returns a realization of B_t given $\{\tau < t\}$ under $\mathbb{P}_0$. For efficiency, simulations rely on the rejection method proposed by O. Faure in [13] for $t \in [0.25, 2]$ and on the inversion technique proposed by N.G. Milstein and M.V. Tretyakov in [23] otherwise.

2.3. Distribution function and density of the first exit time

From (3), one may deduce the distribution function $F(t,x) = \mathbb{P}_x[\tau < t]$ of τ, by taking $g \equiv 1$. Then

$$F(t,x) = 1 - \int_{-1}^{1} p(t,x,y)\,\mathrm{d}y$$

With (1), one gets

$$F(t,x) = 1 - \sum_{n=-\infty}^{+\infty}\left(\mathrm{erf}\left(\frac{x-3-4n}{\sqrt{2t}}\right) - \mathrm{erf}\left(\frac{x+1-4n}{\sqrt{2t}}\right)\right). \tag{6}$$

With (2), one gets

$$F(t,x) = 1 - \frac{4}{\pi}\sum_{n=0}^{+\infty}\frac{(-1)^n}{2n+1}\exp\left(-\frac{\pi^2(2n+1)^2 t}{8}\right)\cos\left(x\pi\left(n+\frac{1}{2}\right)\right). \tag{7}$$

Derivating (6) and (7), one gets that $f(t,x) = \mathbb{P}_x[\tau \in \mathrm{d}t]$ is equal to

$$f(t,x) = \frac{1}{\sqrt{2\pi t^3}}\sum_{n=-\infty}^{\infty}\left((x+4n+1)\exp\left(-\frac{(x+4n+1)^2}{2t}\right)\right.$$
$$\left. -(x+4n-1)\exp\left(-\frac{(x+4n-1)^2}{2t}\right)\right) \tag{8}$$

and

$$f(t,x) = \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{1}{2n+1}\exp\left(-\frac{(2n+1)^2\pi^2}{8}t\right)\sin\left(n\frac{(x+1)\pi}{2}\right). \tag{9}$$

One may simulate a realization of τ by inverting $t \mapsto F(t,x)$ and computing the value $\overline{\tau}$ such that $F(\overline{\tau},x) = \overline{U}$ for a realization $\overline{U}$ of a uniform random variable on $[0,1]$.

In addition, for $t < T$,

$$\mathbb{P}_x[\tau < t | \tau < T] = \frac{\mathbb{P}_x[\tau < t]}{\mathbb{P}_x[\tau < T]}.$$

This way, it is easy to simulate a realization of τ given $\{t < T\}$ by solving $F(\overline{\tau},x) = F(T,x)\overline{U}$ for a realization $\overline{U}$ of a uniform random variable on $[0,1]$.

Numerical aspects: The functions $F(t,x)$ and $f(t,x)$ are computed by the methods

- `d_exit_time` (density).
- `p_exit_time` (distribution function).
- `pd_exit_time` (simultaneous computation of $F(t,x)$ and $f(t,x)$).

In addition, `p_exit_time_fz` (in `exitbm_fz`) returns the values of $F(t,0)$ using tabulated values (faster).

In addition, the function `q_exit_time` returns the value t such that $F(t,x) = v$ for $v \in (-1,1)$. The computation relies on the Newton method, and by dichotomy if the Newton method fails to converge.

In addition, `q_exit_time_fz` (in `exitbm_fz`) returns the solution to $F^{-1}(t,0) = v$ using tabulated values (faster).

A realization of τ under $\mathbb{P}_x$ is returned by `r_exit_time`. This method relies on calling `q_exit_time`.

In addition, `r_exit_time_fz` in (`exitbm_fz`) returns a realization of τ under $\mathbb{P}_0$.

A realization of τ given $\{\tau < T\}$ under $\mathbb{P}_x$ is returned by `r_exit_time_gets`.

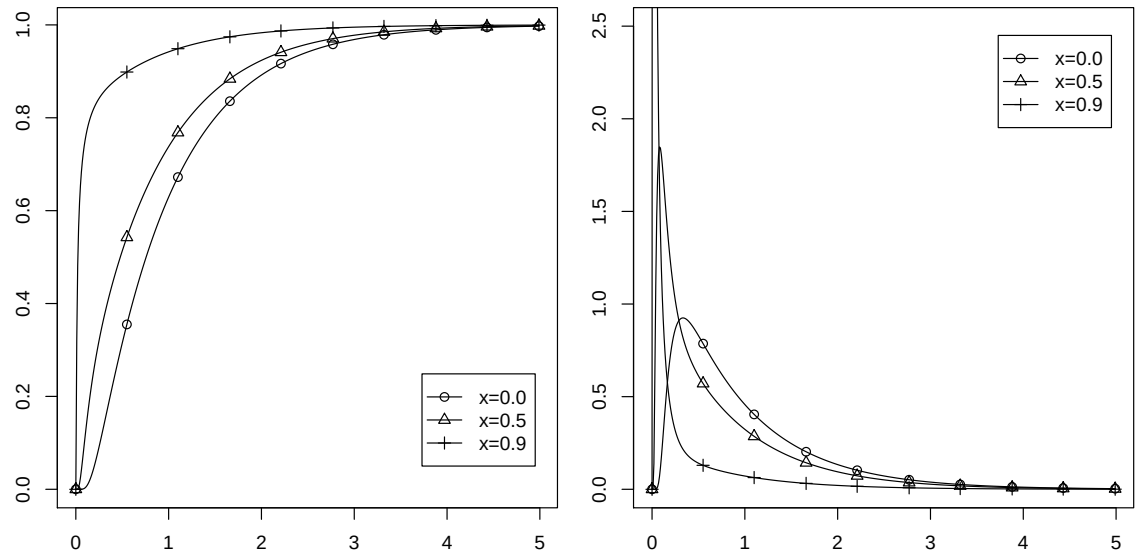


Figure 2: Distribution function $F(t,x)$ (left) and density $f(t,x)$ (right) of the first exit time for three values of the starting point x. Note that $F(t,-x) = F(t,x)$ and $f(t,-x) = f(t,x)$ for $x \in (-1,1)$.

2.4. Position of the Brownian motion before exit

The distribution function $Q(t,x)$ of B_t given $\{t < \tau\}$ is

$$\begin{aligned} Q(t,x,y) = \mathbb{P}_x[B_t < y | t < \tau] &= \mathbb{P}_x[B_t < y, t < \tau]\mathbb{P}_x[t < \tau] \\ &= \frac{\int_{-1}^{y} p(t,x,z)\,\mathrm{d}z}{1 - F(t,x)} = \frac{P(t,x,y)}{1 - F(t,x)}. \end{aligned}$$

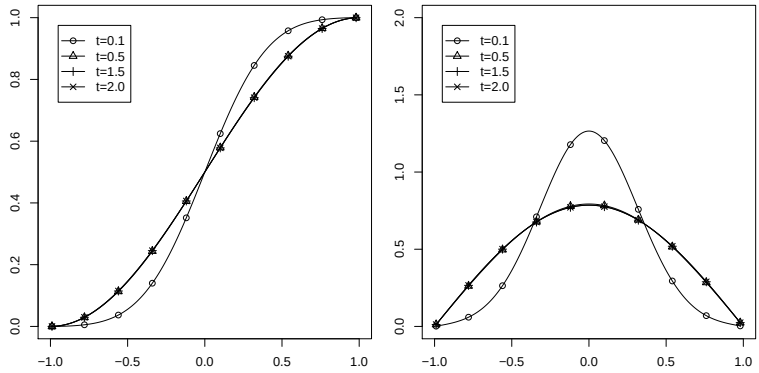
The density $q(t,x,y)$ of B_t given $\{t < \tau\}$ is

$$q(t,x,y) = \frac{P(t,x,y)}{1 - F(t,x)}.$$

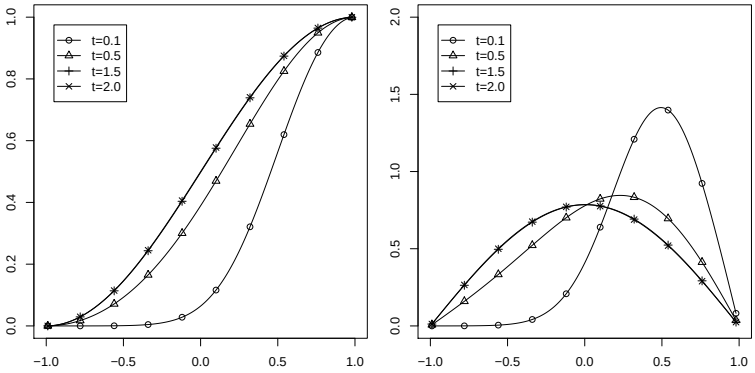
2.5. Position of the Brownian motion given its exit time

The distribution function $q(t,x)$ of B_t given $\{\tau = T\}$ may be computed using a Doob's transform (also called a h-transform, see Appendix A.2). From the strong Markov property applied to time τ,

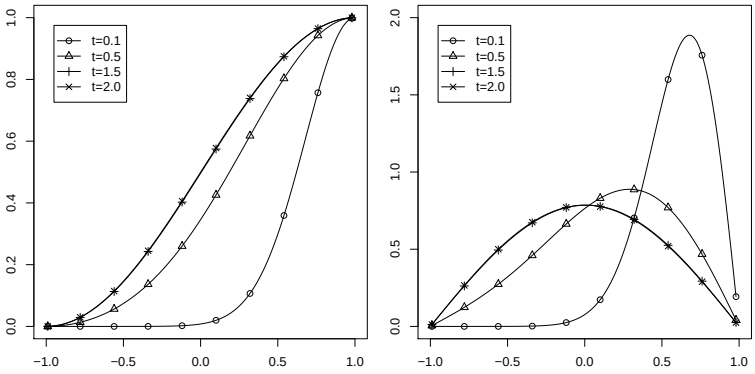
$$\begin{aligned} \mathbb{E}_x[g(B_t)|\tau = T] &= \lim_{\epsilon \to 0} \frac{\mathbb{P}_x[g(B_t), \tau \in [T-\epsilon, T+\epsilon]]}{\mathbb{P}_x[\tau \in [T\epsilon, T+\epsilon]]} \\ &= \frac{\int_{-1}^{1} p(t,x,y) g(y) \int_{T-\epsilon}^{T+\epsilon} f(s-t,y)\,\mathrm{d}s\,\mathrm{d}y}{\int_{T-\epsilon}^{T+\epsilon} f(s-t,y)\,\mathrm{d}y} \xrightarrow[\epsilon \to 0]{} \int_{-1}^{1} r(t,x,y) g(y)\,\mathrm{d}y \end{aligned}$$



(c) $x = 0.0$



(f) $x = 0.5$



(i) $x = 0.9$

Figure 3: Distribution function $Q(t, x)$ (left) and density $q(t, x)$ (right) of the position of the Brownian motion B_t given $\{t < \tau\}$ for different values of the time and different values of the starting point x.

with

$$r(t,x,y) = p(t,x,y)\frac{f(T-t,y)}{f(T,x)}.$$

This function $r(t,x,y)$ gives the density of B_t given $\{\tau = T\}$ around y under $\mathbb{P}_x$. Similarly, we may define the density of B_t given $\{\tau = T, B_\tau = 1\}$ around y under $\mathbb{P}_x$ by

$$r^\dagger(t,x,y) = p^\dagger(t,x,y)\frac{f^\dagger(T-t,y)}{f^\dagger(T,x)}.$$

Numerical aspects: The random variable Y with density $r^\dagger(t,x,\cdot)$, we use the rejection method described in [11, § 3.3, p. 47] by writing

$$r^\dagger(t,x,y) = c\frac{p^\dagger(t,x,y)}{1-F^\dagger(t,x)}\varphi(t,T,x,y)$$

with $\varphi(t,T,x,y) = (1-r^\dagger(t,x))f^\dagger(T-t,y)/(cf^\dagger(T,x))$ and c a real number such that $c \geq \max\{1, \sup_{y\in(-1,1)}\varphi(t,T,x,y)\}$. The idea of the algorithm is to simulate a realization $\overline{X}$ of a random variable with density $p^\dagger(t,x,\cdot)/(1-F^\dagger(t,x))$ until $\varphi(t,T,x,\overline{X})/c$ becomes greater than the realization of a uniform random variable on $[0,1]$. As soon as this condition is satisfied, $\overline{X}$ is a realization of a random variable with density $r^\dagger(t,x,\cdot)$.

The method `r_pos_right_gete` gives a realization of B_t given $\{\tau = T, B_\tau = 1\}$. However, when possible, one should avoid using this method which is slow.

2.6. Exit position of the Brownian motion given it exits from the right endpoint

Again using a h-transform,

$$\mathbb{E}_x[g(B_t), t<\tau | B_\tau = 1] = \frac{\mathbb{E}_x[g(B_t), t<\tau, B_\tau = 1]}{\mathbb{P}_x[B_\tau = 1]}.$$

As $S(B)$ is a martingale with $S(B_\tau) = 1$, τ is a stopping time, then

$$\mathbb{E}_x[g(B_t), t<\tau, B_\tau = 1] = \mathbb{E}_x[g(B_t)S(B_t), t<\tau]$$

while $\mathbb{P}_x[B_\tau = 1] = S(x)$. Thus

$$\mathbb{E}_x[g(B_t), t<\tau | B_\tau = 1] = \int_{-1}^{1} p^\dagger(t,x,y)g(y)\,\mathrm{d}y$$

with

$$p^\dagger(t,x,y) = p(t,x,y)\frac{S(y)}{S(x)} = p(t,x,y)\frac{1+y}{1+x}.$$

2.7. Distribution function and density of the first exit time when the Brownian motion exits first from the right

The distribution function $F^\dagger(t,x) = \mathbb{P}_x[\tau < t | B_\tau = 1]$ of τ is given by

$$F^\dagger(t,x) = 1 - \int_{-1}^{1} p^\dagger(t,x,y)\,\mathrm{d}y.$$

With (1),

$$F^\dagger(t,x) = \frac{2}{1+x}\left(2 - \operatorname{erfc}\left(\frac{x-1}{\sqrt{2t}}\right) - \sum_{n=1}^{+\infty}\left(\operatorname{erfc}\left(\frac{x-1-4n}{\sqrt{2t}}\right) + \operatorname{erfc}\left(\frac{x-1+4n}{\sqrt{2t}}\right) - 2\right)\right). \quad (10)$$

With (2),

$$F^\dagger(t,x) = 1 - \frac{4}{\pi(1+x)}\sum_{n=1}^{+\infty}\frac{(-1)^{n+1}}{n}\exp\left(-\frac{n^2\pi^2 t}{8}\right)\sin\left(n\frac{\pi(x+1)}{2}\right). \quad (11)$$

Derivating (10) and (11), one gets that $f^\dagger(t,x) = \mathbb{P}_x[\tau \in \mathrm{d}t | B_\tau = 1]$ is equal to

$$f^\dagger(t,x) = \frac{-2}{(1+x)\sqrt{2\pi}t^{3/2}}\sum_{n=-\infty}^{+\infty}(x-1-4n)\exp\left(-\frac{(x-1-4n)^2}{2t}\right) \quad (12)$$

and

$$f^\dagger(t,x) = \frac{\pi}{2(1+x)}\sum_{n=1}^{+\infty}(-1)^{n+1}n\exp\left(-\frac{n^2\pi^2 t}{8}\right)\sin\left(n\frac{\pi(x+1)}{2}\right). \quad (13)$$

One may simulate a realization of τ by inverting $t \mapsto F^\dagger(t,x)$ and computing the value $\overline{\tau}$ such that $F^\dagger(\overline{\tau},x) = \overline{U}$ for a realization $\overline{U}$ of a uniform random variable on $[0,1]$.

In addition, for $t < T$,

$$\mathbb{P}_x[\tau < t | \tau < T, B_\tau = 1] = \frac{\mathbb{P}_x[\tau < t]}{\mathbb{P}_x[\tau < T, B_\tau = 1]}.$$

This way, it is easy to simulate a realization of τ given $\{t < T\}$ by solving $F^\dagger(\overline{\tau},x) = \overline{U}F^\dagger(T,x)$ for a realization of a uniform random variable $\overline{U}$ on $[0,1]$.

Numerical aspects: The functions $F^\dagger(t,x)$ and $f^\dagger(t,x)$ are computed by the methods

- `d_exit_time_right` (density).

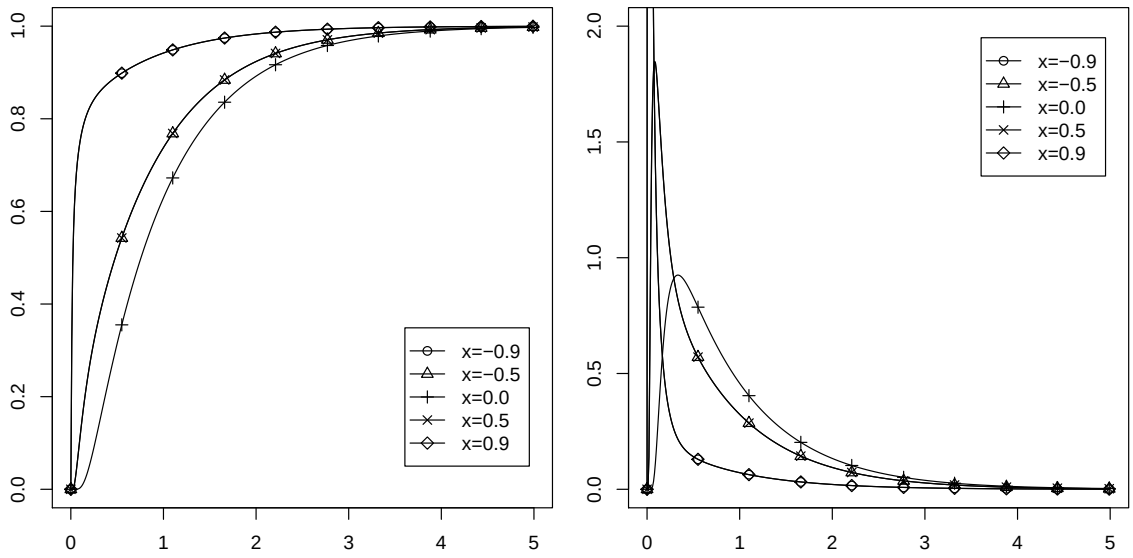


Figure 4: Distribution function $F^\dagger(t,x)$ (left) and density $f^\dagger(t,x)$ (right) of the first exit time for six values of the starting point x.

- `p_exit_time_right` (distribution function).
- `pd_exit_time_right` (simultaneous computation of $F^\dagger(t,x)$ and $f^\dagger(t,x)$).

In addition, the function `q_exit_time_right` returns the value t such that $F^\dagger(t,x) = v$ for $v \in (-1,1)$. The computation relies on the Newton algorithm and the dichotomy when the Newton algorithm fails to converge.

A realization of τ under $\mathbb{P}_x$ is returned by `r_exit_time_right`. This method relies on calling `q_exit_time_right`.

A realization of τ given $\{\tau < \theta\}$ under $\mathbb{P}_x$ is returned by `r_exit_time_right_gets`.

3. Exit time and position from an interval with finite time horizon

We consider a one-dimensional Brownian motion W as well as a fixed time $T > 0$. We are interested in generating random variates $(\tau \wedge T, W_{\tau\wedge T})$ where τ is the first exit time from $[-1,1]$ with $W_0 = x$. Again, the case of an arbitrary interval may be considered using the scaling and spatial homogeneity properties of the Brownian motion (See § A.1). The code relies on Algorithm 1 presented in [15].

Numerical aspects: The value of $\mathbb{P}_x[B_\tau = 1 | \tau < T]$ is given by the Bayes formula

$$\mathbb{P}_x[B_\tau = 1 | \tau < T] = \frac{\mathbb{P}_x[\tau < T | B_\tau = 1]\mathbb{P}_x[B_\tau = 1]}{\mathbb{P}_x[\tau < T]}.$$

Data: A starting point W_0 and a time $T > 0$.
Result: A realization of $(\tau \wedge T, W_{\tau\wedge T})$.
Use a Bernoulli random variate of parameter $\mathbb{P}_x[\tau < T]$ to decide whether $\tau < T$ or $\tau > T$;
if $\tau < T$ **then**
  Use a Bernoulli random variate of parameter $\mathbb{P}_x[W_\tau = 1 | \tau < T]$ to decide whether $W_\tau = 1$ or $W_\tau = -1$;
  if $W_\tau = 1$ **then**
    Generate τ given $\{W_\tau = 1,\ \tau < T\}$;
  else
    Generate τ given $\{W_\tau = -1,\ \tau < T\}$;
  end
else
  Generate W_T given $\{T < \tau\}$
end

Algorithm 1: Exit time and position from $[-1, 1]$ with a finite horizon.

To generate τ given $\{B_\tau = 1,\ \tau < T\}$, use `r_exit_time_right_gets` with y as a starting point. To generate τ given $\{B_\tau = -1,\ \tau < T\}$, use `r_exit_time_right_gets` with $-y$ as a starting point.

The following functions are defined in the sub-library `exitbm_fhor`.

- `r_exit_time_position_before_horizon` returns a realization of $(\tau \wedge T, W_{\tau\wedge T})$ from $[-1, 1]$ for an arbitrary starting point.
- `r_exit_time_position_before_horizon_fz` returns a realization of $(\tau\wedge T, W_{\tau\wedge T})$ from $[-1, 1]$ when $W_0 = 0$ (faster).
- `r_exit_time_position_before_horizon_from_interval` returns a realization of $(\tau\wedge T, W_{\tau\wedge T})$ from an arbitrary interval.

4. Exit time and position from an interval with finite time horizon for the Skew Brownian motion

The Skew Brownian motion (SBM) is a generalization of the Brownian motion and may be constructed by several ways (See the survey article [16] for an introduction to this object and its properties). Among them, one may consider the excursions of the reflected Brownian motion. The excursions are part of the path between two successive, distant zeroes. A Skew Brownian motion is then construct by choosing randomly and independently using a Bernoulli random variable of parameter α. If $\alpha = 1$ (resp. $\alpha = -1$), then this process is a positively (resp. negatively) reflected Brownian motion. If $\alpha = 1/2$, this is a Brownian motion.

It can be shown that the SBM is the unique strong solution of the Stochastic Differential Equation with Local Time

$$X_t = x + B_t + (2\alpha - 1)L_t^0(X), \tag{14}$$

where $(L_t^0(X))_{t\geq 0}$ is the symmetric local time at 0 of the process X:

$$L_t^0(X) = \lim_{\epsilon \to 0} (2\epsilon)^{-1} \text{meas}\{0 \leq s \leq t | X_s \in [-\epsilon, \epsilon]\}.$$

Remark 2. Here, we choose the parameter $\alpha \in (0, 1)$. It is sometimes convenient to use a parameter $\theta = 2\alpha - 1$ in $(-1, 1)$, mainly when one works with (14).

As shown in [12, 19, 26] for example, the SBM is the fundamental process to understand the behavior of a diffusion when it crosses some interface.

Using the symmetry properties of the Brownian motion and the above construction, if $\tau = \inf\{t \geq 0 | X_t \notin [-1, 1]\}$, then $(\tau \wedge T, X_{\tau \wedge T})$ is easily simulated when $X_0 = 0$. The scaling property still applied to the SBM.

Algorithm 2 may then be used as one of the basic brick for implementing the algorithm proposed in [19].

Data: The skewness parameter $\alpha \in (0, 1)$ and a finite horizon $T > 0$.
Result: A realization of $(\tau \wedge T, X_{\tau \wedge T})$ for the SBM of parameter α.
Use a Bernoulli random variate of parameter $\mathbb{P}_0[\tau < T]$ to decide whether $\{\tau < T\}$ of $\{T > \tau\}$;
if $\tau < T$ **then**
  Generate a realization $\overline{\tau}$ of τ given $\tau < T$;
  Use a Bernoulli random variate of parameter α to decide whether $z = X_\tau = 1$ or $z = X_\tau = -1$;
  Return (τ, z);
else
  Generate the position y of a Brownian motion at time T given $\{\tau > T\}$;
  Generate a Bernoulli random variate ϵ in $\{1, -1\}$ with $\mathbb{P}[\epsilon = 1] = \alpha$;
  Return $(T, |y|\epsilon)$;
end

Algorithm 2: First exit time and position from $[-1, 1]$ for the Skew Brownian motion starting from 0 with a finite horizon.

Numerical aspects: The realizations of τ are obtained using the one for the Brownian motion.

The sub-library `exitbm_fhor` provides the following functions:

- `r_exit_time_position_before_horizon_sbm` for the exit time and position from $[-1, 1]$ with a finite horizon.
- `r_exit_time_position_before_horizon_from_interval_sbm` for the exit time and position from an arbitrary interval with a finite horizon.

5. Exit time and position from the square

Now, let us consider a d-dimensional Brownian motion W and set

$$\theta = \inf\{t > 0 \mid W_t \notin [-1,1]^d\}.$$

By using the one-dimensional distribution, as shown in [13, 23, 24], it is possible to simulate the joint distribution (θ, W_θ) under $\mathbb{P}_0$, that is when the Brownian motion start from the center. For another starting point, one may use the variant of the random walk on squares called the random walk of rectangles: see Section 6 below.

Let $(W^1, \dots, W^d)$ be the coordinates of W, which are independent Brownian motion. Note first that

$$\theta = \min_{i=1,\dots,d} \tau^i \text{ for } \tau^i = \inf\{t > 0 \mid W_t^i \notin [-1,1]\}.$$

IF $F_d(t)$ is the distribution function of θ, then a simple computation shows that

$$F_d(t) = 1 - (1 - F(t,0))^d$$

where $F(x,t) = \mathbb{P}_x[\tau^i < t]$.

For a realization $\overline{U}$ of a uniform random variable on $[0,1]$, the solution $\overline{\theta}$ to $F(\overline{\theta}, 0) = 1 - \overline{U}^{1/d}$ is then a realization of θ.

Then a realization of (θ, W_θ) is simulated using Algorithm 3.

Result: A realization of $(\overline{\theta}, \overline{W}_{\overline{\theta}})$.
Draw a realization $\overline{\theta}$ of θ;
Draw with a probability $1/d$ an index $\overline{k}$ and with probability $1/2$ a sign $\overline{\epsilon} \in \{-1, 1\}$;
Set $\overline{W}_{\overline{\theta}}^{\overline{k}} \leftarrow \overline{\epsilon}$;
for $i = 1, \dots, d$, $i \neq \overline{k}$ **do**
  Simulate a realization $\overline{W}_{\overline{\theta}}^{i}$ of $W_{\overline{\theta}}^i$ given $\{\tau^i > \overline{\theta}\}$ using the distribution function $P(\overline{\theta}, 0, \cdot)$;
end

Algorithm 3: Simulation of the first exit time and position from a hyper-cube for a Brownian motion starting from 0.

It is also possible to simulate $(\theta \wedge T, W_{\theta \wedge T})$ by the Algorithm 4.

Numerical aspects: In `exitbm_square`, the functions `p_exit_time_square` and `p_exit_time_hypercube` returns the distribution function of θ for $d = 2$ and any $d \geq 2$.

The functions `r_exit_time_square` and `r_exit_time_hypercube` returns a realization of θ. The function `r_exit_time_square_gets` returns a realization of θ given $\{\theta < T\}$ for $d = 2$.

The function `r_exit_time_position_space_time_square` returns a realization of $(\theta \wedge T, W_{\theta \wedge T})$, while `r_exit_time_position_square` returns a realization of (θ, W_θ).

Data: A finite horizon $T > 0$.
Result: A realization $(\overline{\theta}, \overline{W})$ of $(\theta \wedge T, W_{\theta \wedge T})$.
Use a Bernoulli random variate of parameter $\mathbb{P}_0[\theta < T]$ to decide whether $\theta < T$ or $T \geq \theta$;
if $\theta < T$ **then**
  Generate a random variate $\overline{\theta}$ of θ given $\{\theta < T\}$ by solving $F(\overline{\theta}, 0) = \alpha(1 - \overline{U}^{1/d}, x))$ for a random variate $\overline{U}$ uniform on $[0, 1)$;
  Draw with a probability $1/d$ an indice $\overline{k}$ and with probability $1/2$ a sign $\overline{\epsilon} \in \{-1, 1\}$;
  Set $\overline{W}_{\overline{\theta}}^{\overline{k}} \leftarrow \overline{\epsilon}$;
  for $i = 1, \ldots, d$, $i \neq \overline{k}$ **do**
    Generate a random variate $\overline{W}_{\overline{\theta}}^{i}$ of $W_{\overline{\theta}}^{i}$ given $\{\tau^i > \overline{\theta}\}$ using the distribution function $P(\overline{\theta}, 0, \cdot)$;
  end
else
  for $i = 1, \ldots, d$, $i \neq \overline{k}$ **do**
    Simulate a realization $\overline{W}_{T}^{i}$ of W_{T}^{i} given $\{\tau^i > T\}$ using the distribution function $P(T, 0, \cdot)$;
  end
end

Algorithm 4: Simulation of the first exit time and position from a hyper-cube for a Brownian motion starting from 0.

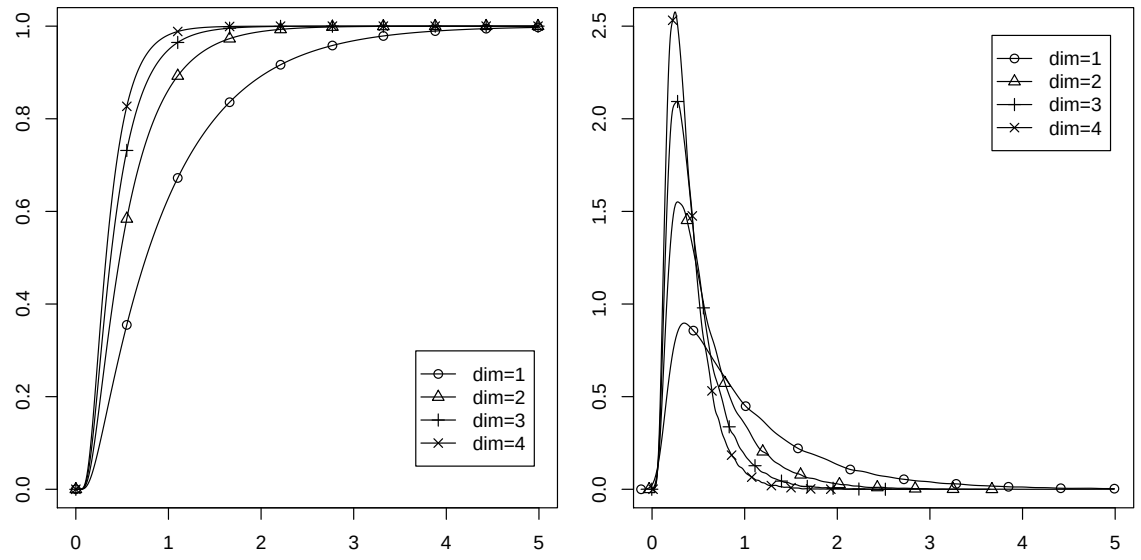


Figure 5: Distribution function $F_d(t)$ (left) and density $f_d(t)$ (right) of the first exit time of an hyper-cube for dimensions from 1 to 4. The density $f_d(t)$ is estimated from random samples.

6. Exit time and position from a rectangle

In [9], we have introduced the *random walk on rectangles* as a way to simulate (θ, W_θ) where

$$\theta = \inf\{t > 0 \mid W_t \notin R = [-L_1, L_1] \times \cdots \times [-L_d, L_d]\}$$

under $\mathbb{P}_x$ for any starting in the (hyper-)rectangle R.

The algorithm in [9] is presented in dimension $d = 2$ yet it may be generalized to any dimension d. Although the current version of `exitbm` provides only interfaces to exit time from rectangles, we present the general algorithm in Algorithm 5.

Remark 3. The permutation σ is used only to reduce the number of potential calls to the evaluation of B_t given $\{\tau = t\}$ (function `r_pos_gete`).

Remark 4. The probability that $B_\tau = 1$ given that $\{\tau < t\}$ may be evaluated through the Bayes' rule

$$\mathbb{P}_x[B_\tau = 1 | \tau < t] = \frac{\mathbb{P}_x[\tau < t | B_\tau = 1]}{\mathbb{P}_x[\tau < t]} \mathbb{P}_x[B_\tau = 1] = \frac{F^\dagger(t,x)}{F(t,x)} \frac{1+x}{2}.$$

Numerical aspects: The library `exitbm_rectangle` provides

- `r_exit_time_position_rectangle`: returns a realization of the first exit time and position (τ, W_τ) from a rectangle.

Data: A rectangle $R = [-L_1, L_1] \times \cdots \times [-L_d, L_d]$ and an initial position $x = (x_1, \ldots, x_d)$.
Compute the distance $d_i = L_i - |x_i|$, $i = 1, \ldots, d$;
Find a permutation σ such that $d_{\sigma(1)} \leq \cdots \leq d_{\sigma(d)}$;
`/* To simplify the notations, we assume that` $\sigma(i) = i$`, See Remark 3. */`
Simulate a realization $(\overline{\tau}_1, \overline{z}_1)$ of (τ_1, W^1_τ) where τ_1 is the first exit time from $[-L_1, L_1]$ for W^1;
Set $\theta \leftarrow \overline{\tau}$, $i_{min} \leftarrow 1$, $J \leftarrow \emptyset$;
for $i = 2, \ldots, d$ **do**
  Evaluate $\alpha_i \leftarrow \mathbb{P}_{x_i}[\tau_i < \theta]$, where τ_i is the first exit time from $[-L_i, L_i]$ for W^i, and use a Bernoulli random variate of parameter α_i to decide whether or not $\{\tau_i < \overline{\tau}\}$;
  if $\{\tau_i < \theta\}$ **then**
    Draw a realization $\overline{z}_i$ of $W^i_{\overline{\tau}_i}$ given $\{\tau_i < \theta\}$;
    `/* See Remark 4. */`
    Draw a realization $\overline{\tau}_i$ of τ_i given $\{\tau_i < \theta, B^i_{\tau_i} = \overline{z}_i\}$ and set $\theta \leftarrow \tau_i$, $i_{min} \leftarrow i$;
  else
    $J \leftarrow J \cup \{i\}$;
    $C_i \leftarrow i_{min}$;
  end
end
for $i = 1, \ldots, d$ **do**
  if $i \notin J$ *and* $i \neq i_{min}$ **then**
    Draw a realization $\overline{z}_i$ of B^i_θ given $\{\tau_i = \overline{\tau}_i\}$;
  else
    Draw a realization of B^i_θ given $\{\tau_i > C_i\}$;
  end
end
Return the exit time θ and the exit position $(z_1, \ldots, z_d)$;

Algorithm 5: First exit time and position from a rectangle for an arbitrary starting point.

- `r_exit_time_position_space_time_rectangle`: returns a realization of $(T \wedge \tau, W_{T\wedge\tau})$ from a rectangle for a fixed T (the algorithm is an easy modification of the previous one).

Numerical tests show that this algorithm is slow with respect to the random walk on squares (See Table 1). This is due to the fact that for the random walk on squares, tabulated values may be used for the exit time. Here, any random variables depends on 1 or 2 parameters, so that tabulating values requires a lot of memory.

This is why we provide the alternative functions

- `r_exit_time_position_rectangle_rws`
  $\rightsquigarrow$ `r_exit_time_position_rectangles`
- `r_exit_time_position_space_time_rectangle_rws`
  $\rightsquigarrow$ `r_exit_time_position_space_time_rectangles`

which relies in the walk on square algorithm in each rectangles, and which are respectively 4 and 6.5 time faster in average (see Table 1 below).

Note that this does not completely supersede the interest of of the random walk on rectangles. First because thie method may be used in combination with Neumann boundary conditions, which is not the case for the random walk on squares. Second because the rectangles may be chosen prior to any simulation, so that one may find a covering (here, the rectangles should overlap) of the domain with rectangles and perform a random walk on squares in each rectangles. On the other hand, with the random walk on squares/spheres, the size of the square/sphere shall be chosen at each step in function of the distance to the boundary. Because of the simple geometry of the squares and rectangles, the condition to know wether or not the particle has reached the boundary is pretty simple. This avoid the cumbersome algorithm proposed in [8] to construct a square in the domain with a boundary which is possibly on one of the boundary.

7. Tests and benchmarks

The library `exitbm_test` contains a series of tests to evaluate and compare the code (simulation of random variable, numerical integration of the density, ...).

Using the `exitbm_benchmark` library and the Rake task `compile_benchmark`, one may define a `C` file `benchmark.c` that may be used to test the time consumption of a simulation (with the Unix command `time` for example).

The execution times for the random variables generators are summarized in Table 1 for $1{,}000{,}000$ samples (on a MacBook 12", 2.4GHz Intel Core 2 duo, the code being compiled with `gcc`), using the Mersennes Twister random number generator of the GSL library. Regarding the parameters, the starting point is chosen uniformly in $(-1, 1)$. For `r_pos_gne` the time t is chosen using an exponential random variable of parameter 1. For `r_pos_getg` (resp. `r_pos_gete`), the minimal time time (resp. the exit time) θ is chosen using an exponential of parameter 1 and the time t is chosen uniformly on $[0, \theta]$.

r_exit_time_fz	0 min 00 sec 05″
r_exit_time_gets_fz	0 min 00 sec 14″
r_pos_gne_fz	0 min 19 sec
r_pos_gne	0 min 57 sec
r_exit_time_right	0 min 58 sec
r_exit_time	1 min 57 sec
r_exit_time_right_gets	2 min 13 sec
r_pos_getg	4 min 14 sec
r_exit_time_gets	4 min 57 sec
r_pos_gete	24 min 09 sec
r_exit_time_position_square	00 min 10 sec
r_exit_time_position_space_time_square	00 min 36 sec
r_exit_time_position_space_time_rectangle_rws	0 min 21 sec
r_exit_time_position_rectangle_rws	0 min 38 sec
r_exit_time_position_rectangle	1 min 21 sec
r_exit_time_position_space_time_rectangle	4 min 20 sec
r_exit_time_position_before_horizon_fz	0 min 16 sec
r_exit_time_position_before_horizon	1 min 03 sec
r_exit_time_position_before_horizon_sbm	0 min 15 sec

Table 1: Execution times for the random variables generators with 1,000,000 samples.

For `r_exit_time_position_space_time_square`, the final time is drawn using an exponential random variable of parameter 1.

For `r_exit_time_position_rectangle` and `r_exit_time_position_rectangle_rws`, the half-width and the half-height of the rectangle are chosen using exponential random variables of parameter 1, while the initial position is chosen uniformly in the rectangle.

For `r_exit_time_position_space_time_rectangle`, the parameters are chosen as for `r_exit_time_position_rectangle`, and the final time is also drawn using an exponential random variable of parameter 1.

For `r_exit_time_position_before_horizon`, the starting point is chosen using uniformly on $(-1, 1)$ and the horizon is draw from an exponential distribution of parameter 1. We also use an exponential distribution of parameter 1 for `r_exit_time_position_before_horizon_fz` as for `r_exit_time_position_before_horizon_sbm` with a skewness parameter uniformly drawn in $(0, 1)$.

A. Appendix

A.1. Scaling and shifting

All the functions relies on the Brownian motion living in $[-1, 1]$. Of course, using scaling and shifting, there is no problem to consider the problem of the first exit time and position for any interval.

The first exit time τ from $[a, b]$ under $\mathbb{P}_x$ is obtained by setting $L = (b-a)/2$ (half-length of the interval) and $y = (x-a)/2(b-a) - 1$ (starting point). Any time parameter shall be divided by L^2. Then the distribution $\mathbb{P}_y$ should be used and any position shall be multiplied by L before being shifted by $(a+b)/2$, and random times shall be multiplied by L^2.

If we are interested in quantities such as the exit time before a given time T, then this time has to be divided by L^2. The position $z \in [-1, 1]$ at a given time T is then transformed to $(a+b)/2 + zL$.

Using symmetry, one may also transform any condition $\{B_\tau = 1\}$ to $\{B_\tau = -1\}$ by changing first x to $-x$, and then by changing the sign of any random position.

A.2. Doob's transform

Let $h(t, x)$ be a function in $\mathcal{C}^{1,2}([0, T) \times \mathbb{R}; \mathbb{R}) \cap \mathcal{C}([0, T] \times \mathbb{R}; \mathbb{R})$ which is solution to

$$\begin{cases} \dfrac{\partial h(t,x)}{\partial t} + \frac{1}{2}\triangle h(t,x) = 0, \text{ on } [0, T) \times \mathbb{R}, \\ h(T, x) = g(x) \text{ on } \mathbb{R}. \end{cases}$$

Then $(h(t, B_t))_{t\in[0,T]}$ is a martingale provided for example that $\mathbb{E}\left[\int_0^T |\nabla_x h(t, B_t)|^2 \, \mathrm{d}t\right]$ is finite.

In this case, if Φ is a function defined on $\mathcal{C}([0, t]; \mathbb{R})$ for some $t \leq T$,

$$\mathbb{E}[\Phi((B_s)_{s\in[0,t]})g(B_T)] = \mathbb{E}[\Phi((B_s)_{s\in[0,t]})h(t, B_t)].$$

In particular, if $g(x) = \mathbf{1}_{[a,b]}(x)$, then

$$\mathbb{E}_x\big[\Phi((B_s)_{s\in[0,t]}) \,|\, B_T \in [a, b]\big] = \frac{\mathbb{E}_x\big[\Phi((B_s)_{s\in[0,t]})h(t, B_t)\big]}{h(0, x)}.$$

This is a *Doob's transform*, or a *h-transform*.

Future works

Future works shall include

- Treatement of other boundary conditions (Neumann/Neumann, and Dirichlet/Neumann).
- Presence of a drift term.
- Random walk on spheres.

Bibliography

Softwares and WEB sites

[1] CEA, CNRS, and INRIA, *CeCILL, Free Software License*. <www.cecill.info>.

[2] M. Galessi *et al.*, *GNU Scientific Library Reference Manual*, 3rd ed., Network Theory Ltd., January 2009.

[3] D. van Heesch *et al.*, *Doxygen*, 1997. <www.doxygen.org>.

[4] Y. Mastumoto *et al.*, *Ruby*, 1995. <ruby-lang.org>.

[5] R. Ihaka and R. Gentleman, *R: A Language for Data Analysis and Graphics*, Journal of Computational and Graphical Statistics **5** (1996), no. 3, 299–314.

[6] J. Weirich, *Rake*. <rubyforge.org/projects/rake>.

Books and articles

[7] J.V. Beck, K.D. Cole, A. Haji-Sheikh, and B. Litkouhi, *Heat conduction using Green's functions*, Series in Computational and Physical Processes in Mechanics and Thermal Sciences, Hemisphere Publishing Corp., 1992.

[8] F. Campillo and A. Lejay, *A Monte Carlo method without grid for a fractured porous domain model*, Monte Carlo Methods Appl. **8** (2002), no. 2, 129–148.

[9] M. Deaconu and A. Lejay, *A random walk on rectangles algorithms*, Methodol. Comput. Appl. Probab. **8** (2006), no. 1, 135–151.

[10] ———, *Simulation of a diffusion process using the importance sampling paradigm* (2007). Preprint.

[11] L. Devroye, *Nonuniform random variate generation*, Springer-Verlag, New York, 1986.

[12] P. Étoré, *On random walk simulation of one-dimensional diffusion processes with discontinuous coefficients*, Electron. J. Probab. **11** (2006), no. 9, 249–275.

[13] O. Faure, *Simulation du mouvement brownien et des diffusions*, Ph.D. thesis, École Nationale des Ponts et Chaussées, 1992.

[14] N. Golyandina, *Convergence rate for spherical processes with shifted centres*, Monte Carlo Methods Appl. **10** (2004), no. 3–4, 287–296.

[15] A. Lejay, *Simulating a diffusion on a graph. Application to reservoir engineering*, Monte Carlo Methods Appl. **9** (2003), no. 3, 241–256.

[16] ———, *On the constructions of the Skew Brownian motion*, Probab. Surv. **3** (2006), 413–466.

[17] A. Lejay and S. Maire, *Computing the principal eigenvalue of the Laplace operator by a stochastic method*, Math. Comput. Simulation **73** (2007), no. 3, 351–363.

[18] ———, *Computing the first eigenelements of some linear operators using a branching Monte Carlo method*, J. Comput. Phys. (2008).

[19] A. Lejay and M. Martinez, *A scheme for simulating one-dimensional diffusion processes with discontinuous coefficients*, Ann. Appl. Probab. **16** (2006), no. 1, 107–139.

[20] S. Maire and E. Tanré, *Some new simulations schemes for the evaluation of Feynman-Kac representations*, Monte Carlo Methods Appl. **14** (2008), no. 1, 29–51.

[21] R. N. Makarov and E. V. Shkarupa, *Stochastic algorithms with Hermite cubic spline interpolation for global estimation of solutions of boundary value problems*, SIAM J. Sci. Comput. **30** (2007/08), no. 1, 169–188.

[22] M. Mascagni and N. A. Simonov, *Monte Carlo methods for calculating some physical properties of large molecules*, SIAM J. Sci. Comput. **26** (2004), no. 1, 339–357 (electronic).

[23] G.N. Milstein and M.V. Tretyakov, *Simulation of a space-time bounded diffusion*, Ann. Appl. Probab. **9** (1999), no. 3, 732–779.

[24] G. N. and Tretyakov Milstein M. V., *Stochastic numerics for mathematical physics*, Scientific Computation, Springer-Verlag, Berlin, 2004.

[25] M. E. Muller, *Some continuous Monte Carlo methods for the Dirichlet problem*, Ann. Math. Statist. **27** (1956), 569–589.

[26] J. Ramirez, E. Thomann, E. Waymire, J. Chastenet, and B. Wood, *A Note on the Theoretical Foundations of Particle Tracking Methods in Heterogeneous Porous Media*, Water Resour. Res. **44** (2007), W01501, DOI 10.1029/2007WR005914.

[27] K. K. Sabelfeld, *Monte Carlo methods in boundary value problems*, Springer Series in Computational Physics, Springer-Verlag, Berlin, 1991.

[28] N.A. Simonov and M. Mascagni, *Random Walk Algorithm for Estimating Effective Properties of Digitized Porous Media*, Monte Carlo Methods Appl. **10** (2004), no. 3–4, 599–608. Conference proceeding of *IV IMACS Seminar on Monte Carlo Methods*.

[29] K. K. Sabelfeld and D. Talay, *Integral formulation of the boundary value problems and the method of random walk on spheres*, Monte Carlo Methods Appl. **1** (1995), no. 1, 1–34.

[30] E. Zauderer, *Partial differential equations of applied mathematics*, 3rd ed., Pure and Applied Mathematics (New York), Wiley-Interscience [John Wiley & Sons], 2006.

History

v 1.0 First release, March 4, 2009

v 2.0 Add a new library `exitbm_fhor`, January 28, 2011

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803