ブラウン運動における初到達時間の分布

Hirotsugu Seike

2026 年 1 月 10 日

1　導入

ブラウン運動 (Brownian motion) とは, 初期値を 0 とし, 時間とともに予測不能な揺らぎを示しながら連続的に推移する確率過程である. 確率過程では, 各時刻 t に確率変数 X_t が対応付けられる. ここで, ブラウン運動のそれを B_t と定義する. つまり, $B_0 = 0$ である. また, 予測不能な揺らぎを表現するため, 任意の時刻 $0 \leq s < t$ について, $B_{t-s} \sim \mathrm{N}(0, t-s)$ とする. ここで, B_t は, 時間に対して連続ではあるが, 至る所微分不可能なことに注意する.

ブラウン運動は, 累積的な熱雑音の挙動, 株価の変動, 保険会社の財務状況 (剰余金) の変化等のモデル化に用いられる. ここで, B_t がある一定の閾値 $a > 0$ を超える時間 r_a を, 初到達時間 $r_a = \inf\{t > 0 | B_t = a\}$ として定義する. r_a が従う分布を考えることで, ブラウン運動に従う挙動のリスクを評価できる. 初到達時間 r_a の定義から, 次式が成立する.

$$\mathrm{P}(r_a \leq t) = \mathrm{P}(\sup_{0 \leq s \leq t} B_s \geq a). \tag{1}$$

右辺は, 最小上界 (greatest upper bound) である上限が a 以上となることを示している (上限・下限存在定理より, その存在が保証されることに注意する). また, 次式が成立する.

$$\mathrm{P}(\sup_{0 \leq s \leq t} B_s \geq a) = 2\mathrm{P}(B_t \geq a). \tag{2}$$

上式は, ブラウン運動において, その軌道が対称な経路が必ず存在する反射原理 (reflection principle) から導かれる. 図 1 にその様子を示す.

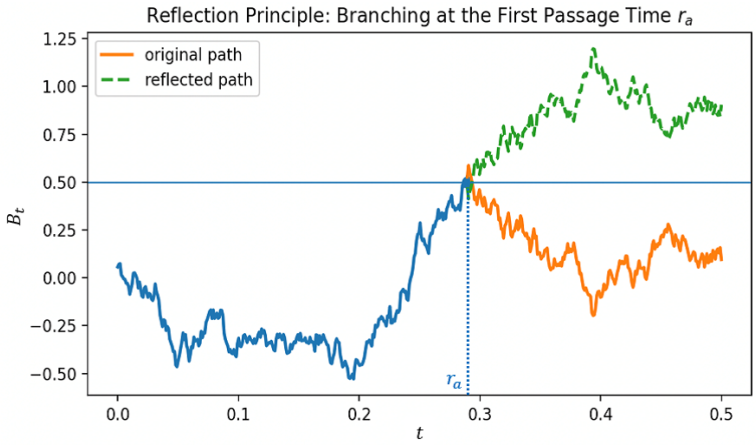

図 1: 反射原理の概念図

つまり, 初到達時間 r_a で $B_{r_a} = a$ となった場合, その後の時刻 $t > r_a$ で, $B_t > a$ となる経路と $B_t < a$ となる経路が同数だけ存在することを意味し, 2 式が成立する根拠となる. ブラウン運動の定義より, $\mathrm{P}(B_t \geq a) = 1 - \Phi(a/\sqrt{t})$ となる (ただし, $\Phi(\cdot)$ は, 標準正規分布の累積分布関数である). 1 式を 2 式の左辺に代入し, 両辺を t に関して微分すると, r_a の確率密度関数 $f_{r_a}(t)$ が得られ, r_a はレヴィ分布 (Lévy distribution) に従うことが分かる.

$$f_{r_a}(t) = \frac{\partial \mathrm{P}(r_a \leq t)}{\partial t}, \tag{3}$$

$$= \frac{\partial\ 2\left(1 - \Phi(a/\sqrt{t})\right)}{\partial t}, \tag{4}$$

$$= -2\phi(a/\sqrt{t}) \cdot \left(-\frac{a}{2} \cdot t^{-3/2}\right), \tag{5}$$

$$= \frac{a}{\sqrt{2\pi}} \cdot t^{-3/2} \cdot \exp(-\frac{a^2}{2t}). \tag{6}$$

ここで, $\phi(\cdot)$ は, 標準積分布の確率密度関数である.

2 プログラム

以下は, 図 1 を作成するプログラムである.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
a = 0.5
T = 0.5
dt = 0.001
n = int(T / dt)

np.random.seed(3)
# Brownian motion
t = np.linspace(0, T, n)
dB = np.sqrt(dt) * np.random.randn(n)
B = np.cumsum(dB)

# First passage time r_a = inf{t : B_t > a}
hit_idx = np.argmax(B > a)
r_a = t[hit_idx]

# Path up to r_a
t_pre = t[:hit_idx+1]
B_pre = B[:hit_idx+1]

# After r_a: original and reflected paths
t_post = t[hit_idx:]
B_up = B[hit_idx:]
B_down = 2*a - B_up   # reflection w.r.t. y = a

# Plot
plt.figure(figsize=(7, 4))
plt.plot(t_pre, B_pre, linewidth=2)
plt.plot(t_post, B_up, linewidth=2, label="original␣path")
plt.plot(t_post, B_down, "--", linewidth=2, label="reflected␣path")
```

```
plt.axhline(a, linewidth=1)
plt.scatter([r_a], [a], s=40)

plt.xlabel("t")
plt.ylabel("X(t)")
plt.title("Reflection␣Principle:␣Branching␣at␣the␣First␣Passage␣Time␣
    $r_a$")
plt.legend()
plt.tight_layout()
plt.show()
```

図 2: 図 1 を作成するプログラム

参考文献

付録 A　特になし