Exact simulation of final, minimal and maximal values of Brownian Motion and jump-diffusions with applications to option pricing

Martin Becker*
Saarland University, Saarbrücken, Germany

05.10.2007

Abstract

We introduce a method for generating $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$, where $W_{x,T}^{(\mu,\sigma)}$ denotes the final value of a Brownian motion starting in x with drift μ and volatility σ at some final time T, $m_{x,T}^{(\mu,\sigma)} = \inf_{0\le t\le T} W_{x,t}^{(\mu,\sigma)}$ and $M_{x,T}^{(\mu,\sigma)} = \sup_{0\le t\le T} W_{x,t}^{(\mu,\sigma)}$.
By using the trivariate distribution of $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$, we obtain a fast method which is unaffected by the well-known random walk approximation errors. The method is extended to jump-diffusion models.
As sample applications we include Monte Carlo pricing methods for European double barrier knock-out calls with continuous reset conditions under both models. The proposed methods feature simple importance sampling techniques for variance reduction.

Keywords: Brownian Motion, Monte Carlo simulation, jump-diffusions, double barrier options, importance sampling

1 Introduction

"Unfortunately, Monte Carlo simulations, which usually provide a flexible and easy approach, do not perform well in the context of barrier options."

As stated in [BaCaIo99], the bad performance of Monte Carlo simulations in this setting is due to the lack of reliable methods for generating triplets $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$, where $W_{x,T}^{(\mu,\sigma)}$ denotes the value of a Brownian motion starting in x with drift μ and volatility σ at some final time T, $m_{x,T}^{(\mu,\sigma)} = \inf_{0\le t\le T} W_{x,t}^{(\mu,\sigma)}$ and $M_{x,T}^{(\mu,\sigma)} = \sup_{0\le t\le T} W_{x,t}^{(\mu,\sigma)}$. The standard methods rely on a discrete N-step random walk approximation of the Brownian motion. Using normally distributed increments, the path of the Brownian motion is sampled at discrete times $t = 0, \frac{1}{N}T, \ldots, \frac{N-1}{N}T, T$ without bias. The minimal and maximal value are then calculated as running minimum resp. maximum of all $N+1$ values. Figure 1 illustrates the bias induced by this procedure for a typical random walk approximation run. Minimal and maximal value are over– resp. underestimated with probability 1, and — as it can be seen in figure 1 — the bias for $N = 365$ can be remarkably high.
[BaCaIo99] give a survey of attempts to improve the performance of Monte Carlo methods for pricing barrier options in geometric Brownian motion models with the help of Brownian bridge concepts[1]. The key idea of these attempts is to use a Brownian bridge for evaluating the probability that the Brownian motion breaches the barrier during a simulation step. They conclude:

*Tel.: +49 681 302 3571 Fax.: +49 681 302 3551 Address: Saarland University, Im Stadtwald, Building C3.1, Room 206, 66123 Saarbrücken, email: `martin.becker@mx.uni-saarland.de`

[1]Brownian bridge concepts have also been used in [MeAt02] for Merton jump-diffusion models.

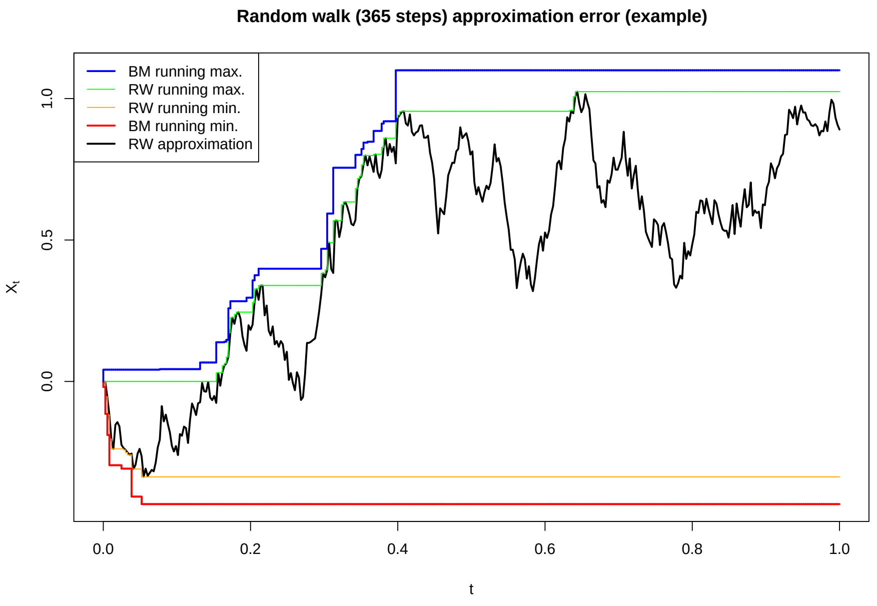

Figure 1: Random walk approximation error for minimum and maximum of Brownian motion. Approximate running minimum/maxium (thin lines) based on 365 step random walk interpolation, running minimum/maximum of corresponding Brownian motion path (thick lines) simulated with algorithm 1.

> "However the above-mentioned authors are limited to those situations in which this probability is known exactly (i.e., one single constant barrier). In particular, cases of interest such as double barriers (possibly time-dependent) cannot be treated."

To overcome some of these limitations, we introduce a fast and unbiased method for generating $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$, relying on the joint distribution of $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$, which is given e.g. in [BoSa02]. This method is extended to jump-diffusion models.
Possible applications for triplets generated with our method include option pricing in geometric Brownian motion and Merton jump-diffusion models, but are not restricted to them. Other applications which use the information contained in final, minimal and maximal values of price processes are volatility/covariance estimation (e.g. [Par80], [GaKl80], [Bec83], [BaTo84], [RoSa91], [YaZh00], [BrDi06], [RoZh07]) and specification tests (e.g. [BeFrKlSK07], [Klö06], [Klö07]). In these contexts, simulation can be used for performance measurements of estimators and power studies of statistical tests.
The remainder is organized as follows: in section 2, we derive the proposed simulation method for $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$. Section 3 contains the extension of the new method to jump-diffusion models. In section 4, the proposed methods are adapted to Monte Carlo option valuation, in particular with respect to European continously monitored constant double barrier knock-out calls in geometric Brownian motion and Merton jump-diffusion frameworks. Section 5 gives a short summary of the results.

2 Simulation method for Brownian motion

The proposed simulation method relies on a sequential generation procedure of final, minimal and maximal value based on univariate (conditional) distributions. The first two components can be generated very fast with the unconditional distribution of final value and the conditional

distribution of minimal value given final value. The conditional distribution of the third component (maximal value) given both other components is more complicated, but tractable with numerical methods.
Before we turn to the univariate distributions, we simplify the problem by eliminating some parameters. Considering scaling properties of the Brownian motion, it is sufficient to generate triplets of $(W_{0,1}^{(\widetilde{\mu},1)}, m_{0,1}^{(\widetilde{\mu},1)}, M_{0,1}^{(\widetilde{\mu},1)})$ for arbitrary $\widetilde{\mu}$, because

$$W_{x,t}^{(\mu,\sigma)} \overset{d}{=} x + \sigma\sqrt{T} W_{0,\frac{t}{T}}^{(\sqrt{T}\frac{\mu}{\sigma},1)}$$

and thus

$$(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)}) \overset{d}{=} (x + \sigma\sqrt{T} W_{0,1}^{(\sqrt{T}\frac{\mu}{\sigma},1)}, x + \sigma\sqrt{T} m_{0,1}^{(\sqrt{T}\frac{\mu}{\sigma},1)}, x + \sigma\sqrt{T} M_{0,1}^{(\sqrt{T}\frac{\mu}{\sigma},1)}) \ ,$$

cf. [BoSa02, p. 68].
To simplify notation, we write for a given $\widetilde{\mu}$

$$(Z, A, B) := (W^{(\widetilde{\mu})}, m^{(\widetilde{\mu})}, M^{(\widetilde{\mu})}) := (W_{0,1}^{(\widetilde{\mu},1)}, m_{0,1}^{(\widetilde{\mu},1)}, M_{0,1}^{(\widetilde{\mu},1)}) \ .$$

The (unconditional) univariate distribution of the final value Z is given e.g. in [BoSa02, 1.0.6, p. 250] as

$$P(Z \in dz) = \frac{1}{\sqrt{2\pi}} e^{-(z-\widetilde{\mu})^2/2} dz \ ,$$

i.e. $Z \sim N(\widetilde{\mu}, 1)$, $F_Z(z) = \Phi(z - \widetilde{\mu})$ and $F_Z^{-1}(u) = \widetilde{\mu} + \Phi^{-1}(u)$, where $\Phi(\,\cdot\,)$ denotes the cumulative distribution function (cdf) of the standard normal distribution.
The conditional univariate cdf of the minimal value A given the final value Z can easily be deduced from [BoSa02, 1.2.8., p. 252]. For $a \leq 0$,

$$\begin{aligned} P(A \leq a; Z \in dz) &= \frac{1}{\sqrt{2\pi}} e^{\widetilde{\mu}z - \widetilde{\mu}^2/2 - (|z-a|-a)^2/2} dz \\ &= \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-(z-\widetilde{\mu})^2/2} dz \cdot e^{-2a(a-z)} & : \ a \leq \min(0,z) \\ \frac{1}{\sqrt{2\pi}} e^{-(z-\widetilde{\mu})^2/2} dz & : \ a > \min(0,z) \end{cases} , \end{aligned}$$

and thus

$$F_{A|Z=z}(a) = \begin{cases} e^{-2a(a-z)} & : \ a \leq \min(0,z) \\ 1 & : \ a > \min(0,z) \end{cases} \quad \text{and} \quad F_{A|Z=z}^{-1}(u) = \frac{z}{2} - \sqrt{\frac{z^2}{4} - \frac{\ln u}{2}} \ .$$

The trivariate distribution of final, minimal and maximal value is given e.g. in [BoSa02, 1.15.8, p. 271]. For $a < \min(0,z) \leq \max(0,z) < b$:

$$\begin{aligned} &P(a < A, B < b, Z \in dz) = \\ &\quad \frac{1}{\sqrt{2\pi}} e^{\widetilde{\mu}z - \widetilde{\mu}^2/2} \sum_{k=-\infty}^{\infty} \left(e^{-(z+2k(b-a))^2/2} - e^{-(z-2a+2k(b-a))^2/2} \right) dz \ . \end{aligned}$$

Differentiation w.r.t. a leads to

$$\begin{aligned} &P(A \in da, B < b, Z \in dz) = \\ &\quad \frac{1}{\sqrt{2\pi}} e^{\widetilde{\mu}z - \widetilde{\mu}^2/2} \sum_{k=-\infty}^{\infty} \Big(2(k+1)(z - 2a + 2k(b-a)) e^{-(z-2a+2k(b-a))^2/2} \\ &\qquad\qquad - 2k(z + 2k(b-a)) e^{-(z+2k(b-a))^2/2} \Big) dz da \end{aligned}$$

for $a < \min(0,z) \leq \max(0,z) < b$.

The cdf of maximal value B given final value Z and minimal value A can now be deduced by division by the bivariate probability density function (pdf) of final and minimal value, which is the result of differencing $P(A \leq a; Z \in dz)$ w.r.t. a:

$$f_{Z,A}(z,a) = \frac{1}{\sqrt{2\pi}} e^{-(z-\widetilde{\mu})^2/2} \cdot 2(z-2a) \cdot e^{-2a(a-z)}$$

The required univariate cdf of maximal value B given final and minimal value (Z, A) for $a < \min(0,z) \leq \max(0,z) < b$ results in:

$$\begin{aligned} F_{B|Z=z,A=a}(b) &= \frac{e^{z^2/2+2a(a-z)}}{z-2a} \sum_{k=-\infty}^{\infty} \Big((k+1)(z-2a+2k(b-a)) e^{-(z-2a+2k(b-a))^2/2} \\ &\qquad -k(z+2k(b-a)) e^{-(z+2k(b-a))^2/2} \Big) \ . \end{aligned}$$

Differentiation w.r.t. b yields the pdf:

$$\begin{aligned} f_{B|Z=z,A=a}(b) &= \frac{e^{z^2/2+2a(a-z)}}{z-2a} \sum_{k=-\infty}^{\infty} \Big(-2k^2(1-(z+2k(b-a))^2) e^{-(z+2k(b-a))^2/2} \\ &\qquad +2k(k+1)(1-(z-2a+2k(b-a))^2) e^{-(z-2a+2k(b-a))^2/2} \Big) \end{aligned}$$

for $a < \min(0,z) \leq \max(0,z) < b$.
It is remarkable that neither $F_{A|Z=z}(a)$ nor $F_{B|Z=z,A=a}(b)$ resp. $f_{B|Z=z,A=a}(b)$ depend directly on $\widetilde{\mu}$ (of course, there is an indirect impact of $\widetilde{\mu}$, because the realisation z of Z depends on $\widetilde{\mu}$). Triplets of (Z, A, B) can now be simulated with a multivariate method of inversion: At first, a realisation z of Z is generated either by using a well-known method for normally distributed (pseudo-)random numbers (prn) or by computing the inverse cdf F_Z^{-1} for $U[0,1]$-distributed prn. In the second step, the realisation a of A can easily be generated by inversion method (from $U[0,1]$-distributed prn applied to $F_{A|Z=z}$), as $F_{A|Z=z}^{-1}$ has a simple closed-form expression. The last step, applying the inversion method again to $F_{B|Z=z,A=a}$, is computationally most burdensome, since the analytical form of $F_{B|Z=z,A=a}$ contains a series. Evaluation and inversion of $F_{B|Z=z,A=a}$ therefore has to be done numerically. Before we focus on this topic, we summarize the simulation procedure in algorithm 1.

Algorithm 1: Simulation of $(W_{x,T}^{(\mu,\sigma)}, \underline{m}_{x,T}^{(\mu,\sigma)}, \overline{M}_{x,T}^{(\mu,\sigma)})$

```
    Input: $x \in \mathbb{R}$, $\mu \in \mathbb{R}$, $\sigma > 0$, $T > 0$, $\varepsilon > 0$
    Output: realisation $(z,a,b)$ of $(W_{x,T}^{(\mu,\sigma)}, \underline{m}_{x,T}^{(\mu,\sigma)}, \overline{M}_{x,T}^{(\mu,\sigma)})$
1 $\widetilde{\mu} \leftarrow \sqrt{T}\frac{\mu}{\sigma}$;
2 (independently) draw $U(0,1)$-distributed prn $u_1$, $u_2$, $u_3$;
3 $\widetilde{z} \leftarrow \Phi^{-1}(u_1) + \widetilde{\mu}$;
4 $\widetilde{a} \leftarrow F_{A|Z=\widetilde{z}}^{-1}(u_2) = \frac{\widetilde{z}}{2} - \sqrt{\frac{\widetilde{z}^2}{4} - \frac{\ln u_2}{2}}$;
5 $\widetilde{b} \leftarrow F_{B|Z=\widetilde{z},A=\widetilde{a}}^{-1}(u_3)$;
6 $(z,a,b) \leftarrow (x + \sqrt{T}\sigma\widetilde{z}, x + \sqrt{T}\sigma\widetilde{a}, x + \sqrt{T}\sigma\widetilde{b})$;
```

The evaluation of $F_{B|Z=z,A=a}$ has to be done by approximating the series. Since the terms in the series decrease with increasing $|k|$, the evaluation of the sum is stopped when the summands fall below a given bound. Mainly depending on $b-a$, bounds of e.g. 10^{-12} are mostly reached for very small $|k|$ (see figure 2).

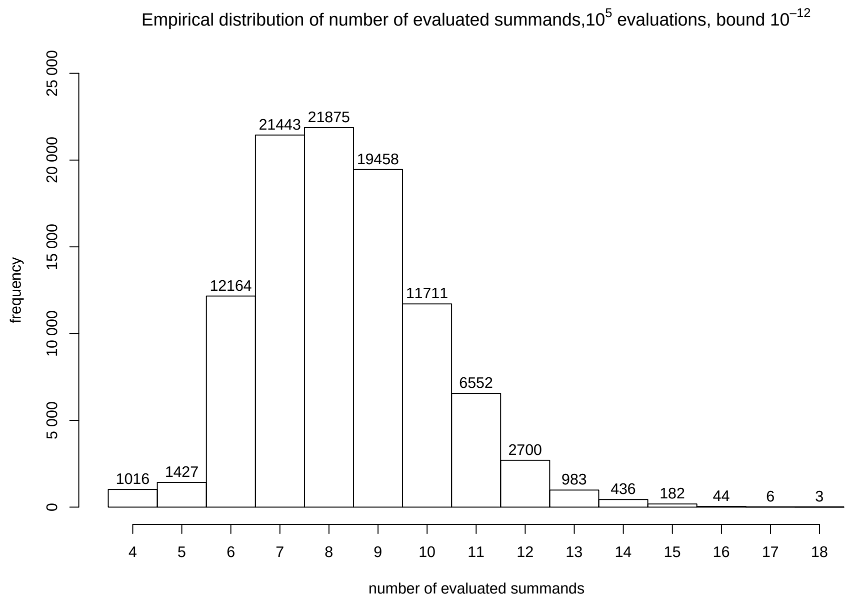

Figure 2: Empirical distribution of number of evaluated terms in series approximation of $F_{B|Z=z,A=a}$ based on 10^5 evaluations with bound 10^{-12}.

To avoid numerical instabilities concerning the evaluation of $e^{z^2/2}$, which emerge for large $|z|$, $F_{B|Z=z,A=a}$ and $f_{B|Z=z,A=a}$ have to be written as

$$\begin{aligned} F_{B|Z=z,A=a}(b) &= \frac{e^{2a(a-z)}}{z-2a} \sum_{k=-\infty}^{\infty} \Big((k+1)(z-2a+2k(b-a))e^{-(z-2a+2k(b-a))^2/2+z^2/2} \\ &\qquad -k(z+2k(b-a))e^{-(z+2k(b-a))^2/2+z^2/2} \Big) \end{aligned}$$

for $a < \min(0,z) \leq \max(0,z) < b$ and

$$\begin{aligned} f_{B|Z=z,A=a}(b) &= \frac{e^{2a(a-z)}}{z-2a} \sum_{k=-\infty}^{\infty} \Big(-2k^2(1-(z+2k(b-a))^2)e^{-(z+2k(b-a))^2/2+z^2/2} \\ &\qquad +2k(k+1)(1-(z-2a+2k(b-a))^2)e^{-(z-2a+2k(b-a))^2/2+z^2/2} \Big) \end{aligned}$$

for $a < \min(0,z) \leq \max(0,z) < b$, resp..
For the inversion of $F_{B|Z=z,A=a}$, a Newton-type algorithm is feasible, since the pdf $f_{B|Z=z,A=a}$ is available. To find a starting value for the Newton iteration, we use our knowledge of the distribution of B conditional on Z (*unconditional* w.r.t. A): From [BoSa02, 1.1.8, p. 251], we have for $b \geq 0$

$$P(B \geq b; Z \in dz) = \frac{1}{\sqrt{2\pi}} e^{\tilde{\mu}z - \tilde{\mu}^2/2 - (|z-b|+b)^2/2} dz \ ,$$

and get — repeating the calculation we did for $F_{A|Z=z}$ —

$$F_{B|Z=z}(b) = \begin{cases} 1 - e^{-2b(b-z)} & : \ b \geq \max(0,z) \\ 0 & : \ b < \max(0,z) \end{cases} \quad \text{and } F^{-1}_{B|Z=z}(u) = \frac{z}{2} + \sqrt{\frac{z^2}{4} - \frac{\ln(1-u)}{2}} \ .$$

As an ad-hoc improvement we propose the following slight modification, which takes the dependence on A into account again:

$$\bar{b}(z,a,u) = \frac{\ln(\frac{1}{z^+ - a})^+}{2} + \frac{z}{2} + \sqrt{\frac{z^2}{4} - \frac{(1 + \ln(1 + \frac{a^2}{|z|+0.01}))\ln(1-u)}{2}}$$

The resulting initial values for the Newton-iteration perform remarkably well (see figure 3). From scratch, we know $F^{-1}_{B|Z=z,A=a}$ is located in $[\max(z,0),\infty)$, and because of the monotony

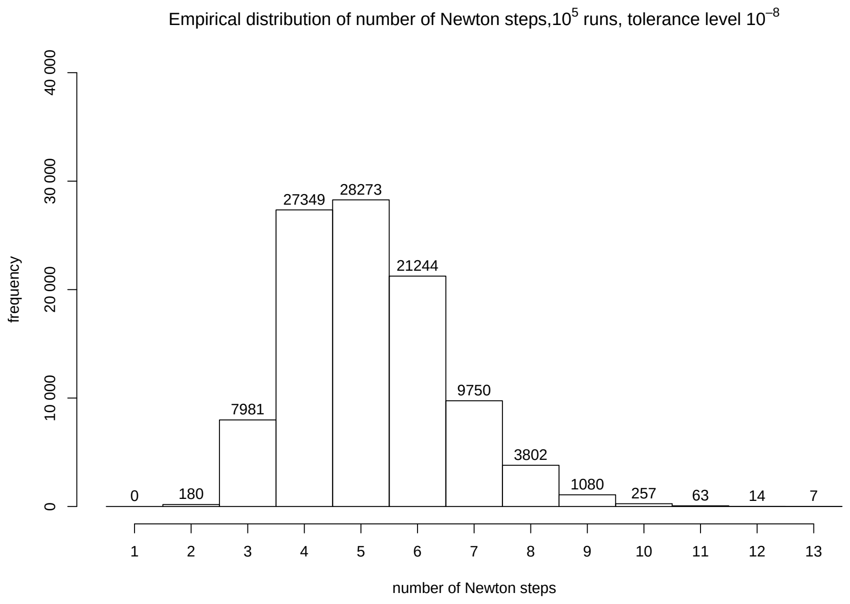

Figure 3: Empirical distribution of Newton-step numbers for 10^5 runs with tolerance level $\varepsilon = 10^{-8}$, initial values for Newton iterations by ad-hoc method.

of $F^{-1}_{B|Z=z,A=a}$, we can shorten this interval in following Newton iterations. If a Newton step would leave the interval, we may force it to stay within. The Newton-type inversion is summarized in algorithm 2.

Based on algorithm 2, a significant speed improvement can be achieved at the cost of some memory (and a fix amount of cpu time required for initialization), if many realisations of $(W^{(\mu,\sigma)}_{x,T}, m^{(\mu,\sigma)}_{x,T}, M^{(\mu,\sigma)}_{x,T})$ for similar values of $\widetilde{\mu} = \sqrt{T}\frac{\mu}{\sigma}$ are to be generated. The speed improvement is accomplished by calculating the initial values for the Newton iterations with the help of a multidimensional linear interpolation based on values of $F^{-1}_{B|Z=z,A=a}(u)$ evaluated at a suitable grid. As an example for applications with a small $\widetilde{\mu}$, the following choice for the grid performs well:

- $z \in \{-5, -4.9, -4.8, \ldots, 4.8, 4.9, 5.0\}$ (101 nodes)
- $a - \min\{0, z\} \in \{-1.976, , -1.951, -1.926, \ldots, -0.051, -0.026, -0.001\}$ (80 nodes)
- $u \in \{0.002, 0.014, 0.026, \ldots, 0.974, 0.986, 0.998\}$ (84 nodes)

In the initialisation procedure for this example, $101 \cdot 80 \cdot 84 = 678720$ inversions have to be done, the memory requirement is approx. 5.18 MByte. Figure 4 illustrates the speed increase on a Pentium D 3.2GHz Linux system (one cpu core used). The observable reduction in cpu

Algorithm 2: Evaluation of $F^{-1}_{B|Z=z,A=a}$

```
   Input: $\varepsilon > 0$, $0 < u < 1$, $z, a \in \mathbb{R}$ where $a \leq \min\{z, 0\}$
   Output: approximation $b$ of $F^{-1}_{B|Z=z,A=a}(u)$ having $|F^{-1}_{B|Z=z,A=a}(u) - b| < \varepsilon$
1  $l \leftarrow \max\{0, z\}$; $r \leftarrow +\infty$;
2  $x_n \leftarrow \frac{\ln(\frac{1}{z^+ - a})^+}{2} + \frac{z}{2} + \sqrt{\frac{z^2}{4} - \frac{(1+\ln(1+\frac{a^2}{|z|+0.01}))\ln(1-u)}{2}}$;
3  repeat
4      $x_o \leftarrow x_n$;
5      $f \leftarrow f_{B|Z=z,A=a}(x_o)$;
6      $F \leftarrow F_{B|Z=z,A=a}(x_o)$;
7      if $F < u$ then $l \leftarrow x_o$ else $r \leftarrow x_o$;
8      $x_n \leftarrow x_o - \frac{F-u}{f}$;
9      if $x_n \notin (l, r)$ then
10         if $r < \infty$ then $x_n \leftarrow \frac{l+r}{2}$ else $x_n \leftarrow 1.2 \cdot (l + 0.1)$;
11 until $|x_n - x_o| < \varepsilon$ ;
12 $b \leftarrow \frac{x_n + x_o}{2}$;
```

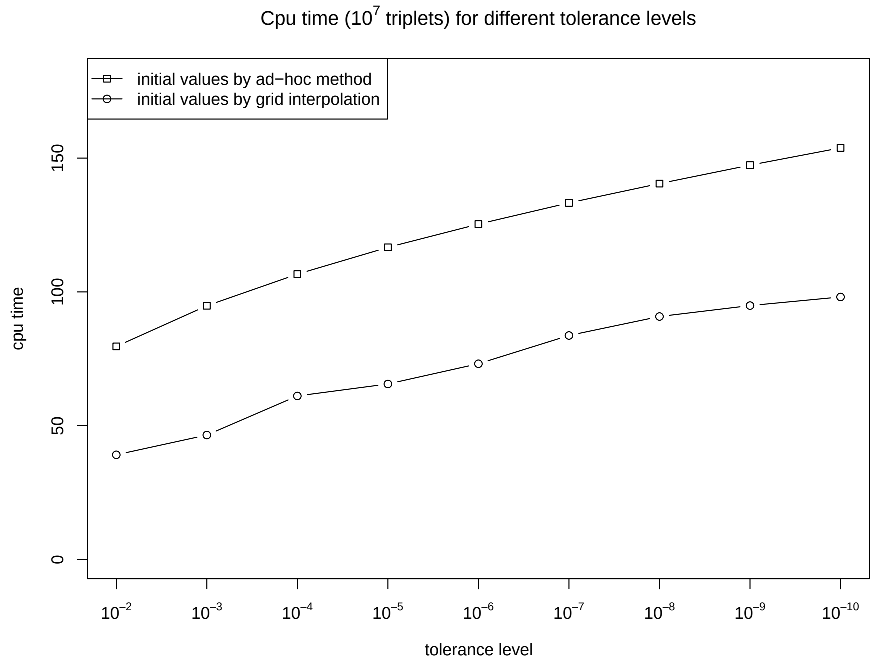

Figure 4: Required cpu time (per 10^7 triplets) for different tolerance levels ε. Comparison of ad-hoc generation and multidimensional linear interpolation for starting values of Newton iterations.

time consumption is essentially the result of a reduced number of Newton steps due to the improved initial values. The proposed simulation algorithm (as all other algorithms presented in this paper) has been implemented as a package for the statistical software R (see [R]). To increase performance, all time critical parts were written in C. The code has been tested on several Intel CPUs (OS: Debian Linux) with the speed measurements (initial values by grid method) given in table 1. On the Pentium D, the performance is measured for one core, using both cores essentially doubles simulation speed.

CPU	time for 10^7 triplets	triplets/sec.
Pentium IV 1.6 GHz	168.16 sec.	59 467
Pentium D 3.2 GHz	90.82 sec.	110 108
Xeon 2.7 GHz	87.48 sec.	114 312

Table 1: Speed of simulation for Brownian motion (algorithm 1) on different CPUs (tolerance level $\varepsilon = 10^{-8}$).

For comparison, we include the speed measurements for generating triplets with a discrete N-step random walk approximation in table 2. As for the new method, the main code is written in C, using the standard random number generator of R (Mersenne-Twister) for uniformly distributed prn.
Ball and Torous have chosen $N = 100\,000$ steps in [BaTo84] "to simulate a Wiener process adequately". For this number of steps, our unbiased method is — depending on CPU type — roughly 1 800 to 2 750 times faster. For $N = 10\,000$ and $N = 5\,000$ steps, the speed factors are reduced to approx. 180-275 and 90-140 resp..

CPU	steps (N)	time for 10^4 triplets	triplets/sec.
Pentium IV 1.6 GHz	100 000	401.04 sec.	25
Pentium IV 1.6 GHz	10 000	40.10 sec.	249
Pentium IV 1.6 GHz	5 000	20.04 sec.	499
Pentium D 3.2 GHz	100 000	251.94 sec.	40
Pentium D 3.2 GHz	10 000	25.18 sec.	397
Pentium D 3.2 GHz	5 000	12.60 sec.	794
Xeon 2.7 GHz	100 000	156.28 sec.	64
Xeon 2.7 GHz	10 000	15.68 sec.	638
Xeon 2.7 GHz	5 000	7.86 sec.	1 272

Table 2: Speed of classical simulation method (random walk approximation) on different CPUs.

3 Extension to jump-diffusion models

The proposed simulation method can easily be extended to more complex models which include Brownian motion components. A well-known class of these models are Lévy processes of jump-diffusion type, also denoted as jump-diffusion models, which have the form

$$X_t = \mu t + \sigma W_t + \sum_{i=1}^{N_t} Y_i,\ t \geq 0,$$

where $(W_t)_{t\geq 0}$ is a Brownian motion, $(N_t)_{t\geq 0}$ is a Poisson process (independent of $(W_t)_{t\geq 0}$) with intensity parameter λ counting the jumps of X, and Y_i are i.i.d. jump sizes (independent of $(W_t)_{t\geq 0}$ and $(N_t)_{t\geq 0}$), see e.g. [CoTa04, p. 111].
Jump diffusions can be understood as Brownian motions which are disrupted by random jump incidences. Since jump times and jump sizes are independent of the Brownian motion itself, algorithm 1 can be used for simulation of final and extremal values of the Brownian motion parts between jumps without modification. For the generation of the jump component, i.e. the compound poisson processes, see e.g. [CoTa04, p. 174].
Triplets $(X_T, \inf_{0\leq t\leq T} X_t, \sup_{0\leq t\leq T} X_t)$ of final and extremal values for jump-diffusions can then be generated with the procedure in algorithm 3.

Algorithm 3: Simulation of $(X_T, \inf_{0\le t\le T} X_t, \sup_{0\le t\le T} X_t)$ for jump diffusions X_t

```
    Input: $x, \mu \in \mathbb{R}$, $\lambda, \sigma, T > 0$, generator for jump sizes $Y_i$
    Output: realisation $(z, a, b)$ of $(X_T, \inf_{0\le t\le T} X_t, \sup_{0\le t\le T} X_t)$ for jump diffusion $X_t$
 1  Draw number of jumps $n$ from Poisson distribution with parameter $\lambda T$;
 2  Draw $n$ jump positions $t_1, \dots, t_n$ independently uniformly distributed on $[0, T]$;
 3  Sort jump positions $t_{(1)} < \dots < t_{(n)}$;
 4  $t_{(0)} \leftarrow 0$; $t_{(n+1)} \leftarrow T$; $x_0 \leftarrow x$;
 5  for $i \in \{1, \dots, n+1\}$ do $\tau_i \leftarrow t_{(i+1)} - t_{(i)}$;
    /* skip following loop if $n = 0$                                    */
 6  for $i \in \{1, \dots, n\}$ do
 7      $(w_i, m_i, M_i) \leftarrow$ realisation of $(W^{\mu,\sigma}_{x_{i-1},\tau_i}, m^{(\mu,\sigma)}_{x_{i-1},\tau_i}, M^{(\mu,\sigma)}_{x_{i-1},\tau_i})$;
 8      $y_i \leftarrow$ realisation of $Y_i$ (generator provided);
 9      $x_i \leftarrow w_i + y_i$;
10  $(w_{n+1}, m_{n+1}, M_{n+1}) \leftarrow$ realisation of $(W^{\mu,\sigma}_{x_n,\tau_{n+1}}, m^{(\mu,\sigma)}_{x_n,\tau_{n+1}}, M^{(\mu,\sigma)}_{x_n,\tau_{n+1}})$;
11  $(z, a, b) \leftarrow (w_{n+1}, \min_{i\in\{1,\dots,n+1\}} m_i, \max_{i\in\{1,\dots,n+1\}} M_i)$;
```

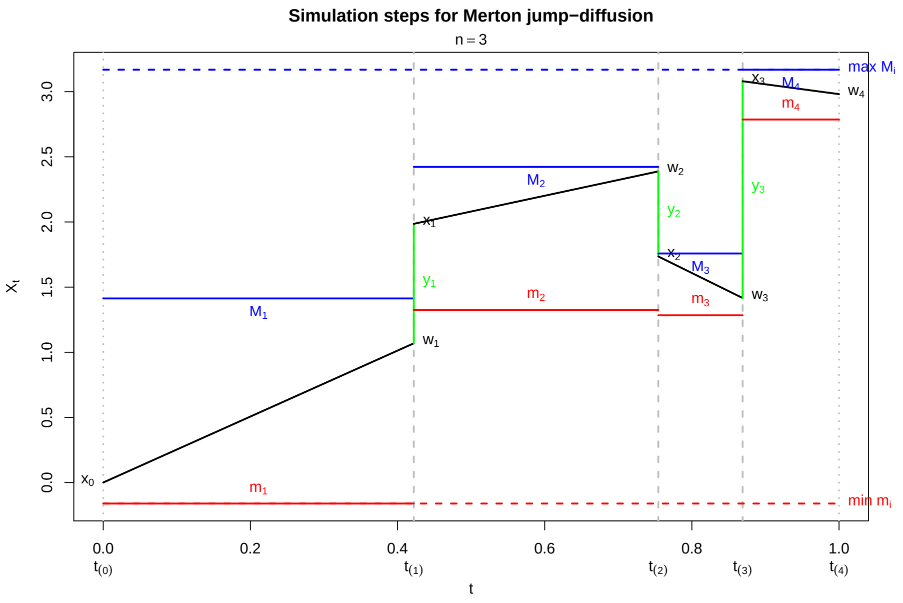

Figure 5: Typical simulation run for Merton jump-diffusion (algorithm 3).

In the Merton model (see [Mer76]), jump sizes are assumed to have a Gaussian distribution (i.e. $Y_i \overset{i.i.d.}{\sim} N(\mu_J, \sigma_J^2)$ for $\mu_j \in \mathbb{R}$, $\sigma_J > 0$). Figure 5 visualizes a typical simulation pass for $\mu = 0$, $\sigma = 1$, $T = 1$, $\mu_J = 0$, $\sigma_J = 1$, $\lambda = 2$. The Brownian motion is disrupted by 3 jumps, resulting in 4 applications of algorithm 1 for the simulation of the extremal values of the Brownian motion parts.

4 Application to option pricing

Since fair prices of options are discounted expected payoffs under equivalent martingale measures, they can be estimated with Monte Carlo methods by simulating under EMM the relevant characteristics of the underlyings and averaging the resulting discounted payoffs.
For some options, the relevant characteristics are (subsets of) final, minimal and maximal value. With our proposed simulation method for these characteristics, a tool for unbiased and fast Monte Carlo pricing is provided.
If the dependence of option payoffs on these characteristics has a special structure (i.e. only parts of the information in the characteristics is reflected in the option payoffs), the price estimation can be improved. In the following we demonstrate importance sampling improvements for European continuously monitored constant double barrier knock-out calls, where dependence of payoffs from maximal and minimal values is limited to the information whether lower or upper boundaries are crossed or not.
We start with a geometric Brownian motion as price process for the underlying, since we are able to compare our results with a benchmark given in [GeYo96] for this case. We extend one of the results for single barrier options in [MeAt02] by turning to Merton jump-diffusions later.

4.1 Option pricing in geometric Brownian motion models

The payoff X for the considered double barrier call option can be written as

$$X = (S_T - K)^+ \mathbb{1}_{L<\min_{0\le t\le T} S_t} \mathbb{1}_{\max_{0\le t\le T} S_t<U} \ ,$$

where L denotes the lower barrier, U the upper barrier and K the strike price at time T ($L < K < U$). The price process S_t of the underlying is assumed to be a geometric Brownian motion starting in $S_0 \in [L, U]$ at time $t = 0$ with drift μ and volatility σ, i.e. under the (in this case unique) risk-neutral measure (depending on the risk-free interest rate r), $S_t = e^{s_t}$ for a Brownian motion s_t with drift $r - \frac{\sigma^2}{2}$ and volatility σ starting in $s_0 = \ln S_0$.
To apply our simulation procedure for Brownian motion, we set

$$\widetilde{\mu} := \sqrt{T}\frac{r-\frac{\sigma^2}{2}}{\sigma},\ l := \frac{\ln(L)-s_0}{\sigma\sqrt{T}},\ u := \frac{\ln(U)-s_0}{\sigma\sqrt{T}}$$

and use $(Z, A, B) = (W^{(\widetilde{\mu})}, m^{(\widetilde{\mu})}, M^{(\widetilde{\mu})})$ to write the payoff as

$$X_{\text{sim}} = (e^{s_0+\sigma\sqrt{T}Z} - K)^+ \mathbb{1}_{l<A} \mathbb{1}_{B<u}.$$

The simple Monte Carlo approach could now be performed as follows:

- Generate N realisations $(z_i, a_i, b_i)_{i=1\ldots N}$ from (Z, A, B).
- For $i = 1, \ldots, N$ calculate $x_i = (e^{s_0+\sigma\sqrt{T}z_i} - K)^+ \mathbb{1}_{l<a_i} \mathbb{1}_{b_i<u}$.
- Estimate fair price as discounted estimated expectation under risk-neutral measure as $\widehat{C}_{K,L,U} = e^{-rT}\widehat{E(X)} = e^{-rT} \cdot \frac{1}{N}\sum_{i=1}^N x_i$.

The main disadvantage of this procedure is the (potentially large) number of zeros in the sequence of calculated payoffs x_i. Apart from having bad impact on the variance of the estimation procedure, every time a triplet (z_i, a_i, b_i), where $e^{s_0+\sigma\sqrt{T}z_i} < K$, $a_i < l$ or $b_i > u$, is generated, most of the CPU time is vasted, since most of the information in (z_i, a_i, b_i) is discarded.
In order to reduce variance and increase simulation speed, we restrict the simulation procedure to the "important" region, where the payoff x_i is not zero. With $k := \frac{\ln(K)-s_0}{\sigma\sqrt{T}}$ this region consists of all triplets (z_i, a_i, b_i) with

$$l \le \min(a_i, k) \le \max(a_i, k) \le z_i \le b_i \le u \ .$$

Of course, the generated (z_i, a_i, b_i) resp. the resulting payoffs x_i have to be weighted with a "likelihood" for fulfilling these conditions.
In the following, we write the expected payoff as an integral, to illustrate how our sequential simulation method can be applied to improve estimation by importance sampling:

$$
\begin{aligned}
&E(X)\\
&= \int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}} (e^{s_0+\sigma\sqrt{T}z} - K)^+ \mathbb{1}_{l<a}\mathbb{1}_{b<u} f_{(Z,A,B)}(z,a,b)\, db\, da\, dz\\
&= \int_k^u \int_l^z \int_z^u (e^{s_0+\sigma\sqrt{T}z} - K) f_{(Z,A,B)}(z,a,b)\, db\, da\, dz\\
&= \int_k^u (e^{s_0+\sigma\sqrt{T}z} - K) \int_l^z \int_z^u f_{B|Z=z,A=a}(b)\, db\, f_{A|Z=z}(a) f_Z(z)\, da\, dz\\
&= \int_k^u (e^{s_0+\sigma\sqrt{T}z} - K) \int_l^z F_{B|Z=z,A=a}(u) f_{A|Z=z}(a) f_Z(z)\, da\, dz\\
&= \int_k^u (e^{s_0+\sigma\sqrt{T}z} - K) \int_l^z F_{B|Z=z,A=a}(u) \frac{f_{A|Z=z}(a)}{p_2(l,z)} p_2(l,z) \frac{f_Z(z)}{p_1(k,u)} p_1(k,u)\, da\, dz\ ,
\end{aligned}
$$

where $p_1(k,u) = F_Z(u) - F_Z(k) = \Phi(u-\widetilde{\mu}) - \Phi(k-\widetilde{\mu})$ measures the likelihood (probability) for fulfilling the condition $k < z_i < u$, and — for a given $z_i \in [k,u]$ — the term $p_2(l,z) = F_{A|Z=z}(z) - F_{A|Z=z}(l) = 1 - e^{-2l(l-z)}$ measures the likelihood for the condition $l < a_i (< z_i)$ for a given z_i.
$\widetilde{f}_{k,u}(z) := \frac{f_Z(z)}{p_1(k,u)}\mathbb{1}_{[k,u]}(z)$ can be interpreted as a density of Z restricted to the interval $[k,u]$, whereas $\widetilde{f}_{l,z}(a) := \frac{f_{A|Z=z}(a)}{p_2(l,z)}\mathbb{1}_{[l,z]}(a)$ is a density of $A|Z=z$ restricted to $[l,z]$. In the resulting importance sampling method, we draw realisations of Z and $A|Z$ from these truncated distributions.
The computationally burdensome inversion of $F_{B|Z,A}$ is completely avoided, since the innermost part of the integral can be evaluated directly with the cdf $F_{B|Z,A}$, without the need for simulating the component b_i at all.
The estimation procedure for the option price is summarized in algorithm 4.

Algorithm 4: Double Barrier knock-out call valuation, geometric BM

```
   Input: $\mu, r \in \mathbb{R}$, $\sigma, T, S_0, \varepsilon > 0$, $0 < L < S_0$, $U > S_0$, $L < K < U$, $N \in \mathbb{N}$
   Output: Estimator $\widehat{C}_{K,L,U}$ of call price
 1 $s_0 \leftarrow \ln(S_0)$; $\widetilde{\mu} \leftarrow \sqrt{T}\frac{r-\frac{\sigma^2}{2}}{\sigma}$; $l \leftarrow \frac{\ln(L)-s_0}{\sigma\sqrt{T}}$; $u \leftarrow \frac{\ln(U)-s_0}{\sigma\sqrt{T}}$; $k \leftarrow \frac{\ln(K)-s_0}{\sigma\sqrt{T}}$;
 2 $p_{z_m} \leftarrow \Phi(k-\widetilde{\mu})$; $p_{z_M} \leftarrow \Phi(u-\widetilde{\mu})$; $p_1 \leftarrow p_{z_M} - p_{z_m}$;
 3 for $i \in \{1,\dots,N\}$ do
 4     (independently) draw $U(0,1)$-distributed prn $u_1$, $u_2$;
 5     $z_i \leftarrow \widetilde{\mu} + \Phi^{-1}(p_{z_m} + u_1(p_{z_M} - p_{z_m}))$;          /* $\rightsquigarrow z_i \in [k,u]$ */
 6     $p_2 \leftarrow 1 - e^{-2l(l-z_i)}$;
 7     $a_i \leftarrow \frac{z_i}{2} - \sqrt{\frac{z_i^2}{4} - \frac{\ln(u_2 \cdot p_2)}{2}}$;          /* $\rightsquigarrow a_i \in [l,z_i]$ */
 8     $p_3 \leftarrow F_{B|Z=z_i,A=a_i}(u)$ (with tolerance $\varepsilon$);
 9     $x_i \leftarrow (e^{s_0+\sigma\sqrt{T}z_i} - K) \cdot p_1 \cdot p_2 \cdot p_3$;
10 $\widehat{C}_{K,L,U} \leftarrow e^{-rT}\frac{1}{N}\sum_{i=1}^N x_i$;
```

In table 3, we compare the accuracy of our new Monte Carlo pricing method to the results in [GeYo96]. In [GeYo96], different pricing methods for European continuously monitored constant double barrier knock-out calls — including a standard Monte Carlo method based on random-walk approximation — were compared for three different parameter sets. For all considered call options, $T = 1$ and $S_0 = 2$.

Pricing method	$\sigma = 0.2$, $r = 0.02$, $K = 2$, $L = 1.5$, $U = 2.5$	$\sigma = 0.5$, $r = 0.05$, $K = 2$, $L = 1.5$, $U = 3$	$\sigma = 0.5$, $r = 0.05$, $K = 1.75$, $L = 1$, $U = 3$
Geman-Yor	0.0411	0.0178	0.07615
Kunitomo-Ikeda	0.041089	0.017856	0.076172
Old MC	0.0425	0.0191	0.0772
est. st.dev.	0.0030	0.0030	0.0030
New MC, 10^6 draws	0.04109369	0.01785189	0.07616213
est. st.dev.	0.00002466	0.00001482	0.00005738
New MC, 10^8 draws	0.04108519	0.01785838	0.07616824
est. st.dev.	0.00000247	0.00000148	0.00000574

Table 3: Estimation results for geometric Brownian motion, compared to [GeYo96]. The values for the first three methods are adopted from [GeYo96], p. 1236.

As expected, there is no significant bias in our results. The computation speed is approx. 350 000 draws per second on a Xeon 2.7 GHz CPU. The estimation based on 10^6 draws was done in 3 seconds.

4.2 Option pricing in Merton jump-diffusion models

The extension of the importance sampling techniques to options, where the log-price of the underlying follows a Merton jump-diffusion, is straightforward[2]: for the last part of the log-price-process (the Brownian motion component after the last jump), we can adopt the procedure without changes. For the other parts, which consist of a Brownian motion with a jump at the end, we have to change the procedure in the following manner:

- The Brownian motion must not leave the interval $[l, u]$, but the final value may fall below k. So, performing the importance sampling, we have to change k to l.
- In order to avoid a knock out, the jump size has to be adjusted in such a way, that the log-price-process does not leave the interval $[l, u]$. This is again done by importance sampling of the jump size.

This extension of the importance sampling techniques for Brownian motion to jump diffusions ensures that the barriers are neither crossed by the Brownian motion parts nor by the jumps. So, positive payoffs — weighted with the appropriate likelihood ratios — are attained in every iteration.
Note that Merton jump-diffusion models lead to incomplete markets, so there is no unique EMM any more. Since the discussion of option pricing in incomplete markets is far beyond the scope of this example, we simply adopt the choice from [Mer76] without further investigations. The method is summarized in algorithm 5. The likelihood ratio lr is initialized in line number 8. The inner for-loop (lines 9–19) covers all but the last Brownian motion parts (and the adjacent jumps), the last Brownian motion part is addressed separately in lines 20–28.

[2]Importance sampling techniques for single barrier options in Merton jump-diffusion models were also applied in [JoLe07].

Algorithm 5: Double Barrier knock-out call valuation, Merton jump-diffusion

```
Input: $\mu, \mu_J, r \in \mathbb{R}$, $\sigma, \sigma_J, \lambda, T, S_0, \varepsilon > 0$, $0 < L < S_0 < U$, $L < K < U$, $N \in \mathbb{N}$
Output: Estimator $\widehat{C}_{K,L,U}$ of call price
1  $s_0 \leftarrow \ln(S_0)$;
2  for $i \in \{1, \ldots, N\}$ do
3      Draw number of jumps $n$ from Poisson distribution with parameter $\lambda T$;
4      Draw $n$ jump positions $t_1, \ldots, t_n$ independently uniformly distributed on $[0, T]$;
5      Sort jump positions $t_{(1)} < \ldots < t_{(n)}$;
6      $t_{(0)} \leftarrow 0$; $t_{(n+1)} \leftarrow T$;
7      for $i \in \{1, \ldots, n+1\}$ do $\tau_i \leftarrow t_{(i+1)} - t_{(i)}$;
8      $z_i \leftarrow s_0$; $lr \leftarrow 1$;            /* start with likelihood ratio ($lr$) 1 */
       /* skip following loop if $n = 0$ */
9      for $j \in \{1, \ldots, n\}$ do
10         $\widetilde{\mu} \leftarrow \frac{\sqrt{\tau_j}}{\sigma}(r - \frac{\sigma^2}{2} - \lambda(e^{\mu_J + \frac{\sigma_J^2}{2}} - 1))$; $l \leftarrow \frac{\ln(L) - z_i}{\sigma\sqrt{\tau_j}}$; $u \leftarrow \frac{\ln(U) - z_i}{\sigma\sqrt{\tau_j}}$;
11         $p_{z_m} \leftarrow \Phi(l - \widetilde{\mu})$; $p_{z_M} \leftarrow \Phi(u - \widetilde{\mu})$; $p_1 \leftarrow p_{z_M} - p_{z_m}$;
12         (independently) draw $U(0,1)$-distributed prn $u_1$, $u_2$, $u_3$;
13         $z_i \leftarrow \widetilde{\mu} + \Phi^{-1}(p_{z_m} + u_1(p_{z_M} - p_{z_m}))$;        /* $\rightsquigarrow z_i \in [l, u]$ */
14         $p_2 \leftarrow 1 - e^{-2l(l - z_i)}$;
15         $a_i \leftarrow \frac{z_i}{2} - \sqrt{\frac{z_i^2}{4} - \frac{\ln(u_2 \cdot p_2)}{2}}$;        /* $\rightsquigarrow a_i \in [l, z_i]$ */
16         $p_3 \leftarrow F_{B|Z=z_i, A=a_i}(u)$ (with tolerance $\varepsilon$);
17         $p_{y_m} \leftarrow \Phi(\frac{l - z_i - \mu_J}{\sigma_J})$; $p_{y_M} \leftarrow \Phi(\frac{u - z_i - \mu_J}{\sigma_J})$; $p_4 \leftarrow p_{y_M} - p_{y_m}$;
18         $y \leftarrow \mu_J + \sigma_J \Phi^{-1}(p_{y_m} + u_3(p_{y_M} - p_{y_m}))$;        /* $\rightsquigarrow z_i + y \in [l, u]$ */
19         $z_i \leftarrow z_i + y$; $lr \leftarrow lr \cdot p_1 \cdot p_2 \cdot p_3 \cdot p_4$;
20     $\widetilde{\mu} \leftarrow \frac{\sqrt{\tau_{n+1}}}{\sigma}(r - \frac{\sigma^2}{2} - \lambda(e^{\mu_J + \frac{\sigma_J^2}{2}}))$; $l \leftarrow \frac{\ln(L) - z_i}{\sigma\sqrt{\tau_{n+1}}}$; $u \leftarrow \frac{\ln(U) - z_i}{\sigma\sqrt{\tau_{n+1}}}$;
21     $k \leftarrow \frac{\ln(K) - z_i}{\sigma\sqrt{\tau_{n+1}}}$;
22     $p_{z_m} \leftarrow \Phi(k - \widetilde{\mu})$; $p_{z_M} \leftarrow \Phi(u - \widetilde{\mu})$; $p_1 \leftarrow p_{z_M} - p_{z_m}$;
23     (independently) draw $U(0,1)$-distributed prn $u_1$, $u_2$;
24     $z_i \leftarrow \widetilde{\mu} + \Phi^{-1}(p_{z_m} + u_1(p_{z_M} - p_{z_m}))$;        /* $\rightsquigarrow z_i \in [k, u]$ */
25     $p_2 \leftarrow 1 - e^{-2l(l - z_i)}$;
26     $a_i \leftarrow \frac{z_i}{2} - \sqrt{\frac{z_i^2}{4} - \frac{\ln(u_2 \cdot p_2)}{2}}$;        /* $\rightsquigarrow a_i \in [l, z_i]$ */
27     $p_3 \leftarrow F_{B|Z=z_i, A=a_i}(u)$ (with tolerance $\varepsilon$);
28     $lr \leftarrow lr \cdot p_1 \cdot p_2 \cdot p_3$;
29     $x_i \leftarrow (e^{s_0 + \sigma\sqrt{T} z_i} - K) \cdot lr$;
30 $\widehat{C}_{K,L,U} \leftarrow e^{-rT} \frac{1}{N} \sum_{i=1}^{N} x_i$;
```

5 Summary

In this paper we proposed a new method for generating (pseudo-) random triplets of the trivariate distribution of $(W_{x,T}^{(\mu,\sigma)}, m_{x,T}^{(\mu,\sigma)}, M_{x,T}^{(\mu,\sigma)})$. In contrast to standard simulation methods for this purpose, which rely on random walk approximations, our method is unbiased. Narrowing the bias for standard methods must be payed for with slowdowns of simulation, which makes our method about 2 000 times faster than standard methods in common situations.
A typical application for simulating final, minimal and maximal values of a stochastic process is the simulation of final, high and low (log-)prices of securities. Although not restricted to Brownian motion itself, stochastic models which include Brownian motion components, such as jump-diffusion models, are often used to model log-prices in financial markets.
With the extension of our new simulation methods to jump-diffusion models, a reliable tool for Monte Carlo option pricing in geometric Brownian motion and (Merton) jump-diffusion

models is provided. In our sample application, we focused on European continously monitored constant double barrier knock-out calls. The investigation of other types of options, such as swing options or options with changing barriers, and the extension to other types of price processes which include diffusion components is left to further studies.

References

[BaCaIo99] Paolo Baldi, Lucia Caramellino & Maria Gabriella Iovino, *Pricing General Barrier Options: A Numerical Approach Using Sharp Large Deviations*, Mathematical Finance **9** (1999), Nr. 4, 293–322.

[BaTo84] Clifford A. Ball & Walter N. Torous, *The Maximum Likelihood Estimation of Security Price Volatility: Theory, Evidence, and Application to Option Pricing*, Journal of Business **57** (1984), Nr. 1, 97–112.

[Bec83] Stan Beckers, *Variances of Security Price Returns Based on High, Low, and Closing Prices*, Journal of Business **55** (1983), Nr. 1, 97–112.

[BeFrKlSK07] Martin Becker, Ralph Friedmann, Stefan Klößner & Walter Sanddorf-Köhle, *A Hausman test for Brownian motion*, AStA - Advances in Statistical Analysis **91** (2007), Nr. 1, 3–21.

[BoSa02] Andrei N. Borodin & Paavo Salminen, *Handbook of Brownian Motion - Facts and Formulae*, 2. Aufl., Probability and its Applications, Birkhäuser, Basel et al., 2002.

[BrDi06] Michael W. Brandt & Francis X. Diebold, *A No-Arbitrage Approach to Range-Based Estimation of Return Covariances and Correlations*, Journal of Business **79** (2006), Nr. 1, 61–74.

[CoTa04] Rama Cont & Peter Tankov, *Financial Modelling with Jump Processes*, CRC Financial Mathematics Series, Chapman & Hall, Boca Raton et al., 2004.

[GaKl80] Mark B. Garman & Michael J. Klass, *On the Estimation of Security Price Volatilities from Historical Data*, Journal of Business **53** (1980), Nr. 1, 67–78.

[GeYo96] Hélyette Geman & Marc Yor, *Pricing and Hedging Double-Barrier Options: A Probabilistic Approach*, Aktuarielle Ansätze für Finanz-Risiken: Beiträge zum 6. Internationalen AFIR-Colloquium, Nürnberg, 1.-3. Oktober (Peter Albrecht, Hrsg.), Actuarial Approach for Financial Risks (AFIR), International Actuarial Association, 1996, 1227–1246.

[JoLe07] Mark S. Joshi & Terence S. Leung, *Using Monte Carlo simulation and importance sampling to rapidly obtain jump-diffusion prices of continuous barrier options*, The Journal of Computational Finance **10** (2007), Nr. 4, 93–105.

[Klö06] Stefan Klößner, *Empirical Evidence: Intraday Returns are neither Symmetric nor Lévy Processes*, 2006, Paper presented at Statistische Woche, Dresden, 2006, September 18-21.

[Klö07] ———, *On Intraday Time-Reversibility of Return Processes*, 2007, Paper presented at Statistics under one umbrella, Bielefeld, 2007, March 27-30.

[MeAt02] Steve A. K. Metwally & Amir F. Atiya, *Using Brownian Bridge for Fast Simulation of Jump-Diffusion Processes and Barrier Options*, The Journal of Derivatives **10** (2002), Nr. 1, 43–54.

[Mer76] Robert C. Merton, *Option pricing when underlying stock returns are discontinuous*, Journal of Financial Economics **3** (1976), Nr. 1–2, 125–144.

[Par80] Michael Parkinson, *The Extreme Value Method for Estimating the Variance of the Rate of Return*, Journal of Business **53** (1980), Nr. 1, 61–65.

[R] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2007, ISBN 3-900051-07-0.

[RoSa91] L. C. G. Rogers & S. E. Satchell, *Estimating variance from high, low and closing prices*, The Annals of Applied Probability **1** (1991), Nr. 4, 504–512.

[RoZh07] L. C. G. Rogers & Fanyin Zhou, *Estimating correlation from high, low, opening and closing prices*, February 2007, to appear in: The Annals of Applied Probability.

[YaZh00] Dennis Yang & Qiang Zhang, *Drift-Independent Volatility Estimation Based on High, Low, Open, and Close Prices*, Journal of Business **73** (2000), Nr. 3, 477–491.