Monte Carlo simulation for Barndorff-Nielsen and Shephard model under change of measure

Takuji Arai* †
Yuto Imai‡

March 13, 2024

Abstract

The Barndorff-Nielsen and Shephard model is a representative jump-type stochastic volatility model. Still, no method exists to compute option prices numerically for the non-martingale case with infinite active jumps. We develop two simulation methods for such a case under change of measure and conduct some numerical experiments.

Keywords: Barndorff-Nielsen and Shephard model, Stochastic volatility model, Minimal martingale measure, Monte Carlo simulation

1 Introduction

The Barndorff-Nielsen and Shephard (BNS) model, undertaken by Barndorff-Nielsen and Shephard [2] and [3], is a non-Gaussian Ornstein-Uhlenbeck (OU) type stochastic volatility model and has been actively studied in recent years. In option pricing for the BNS model, the Carr-Madan method based on the Fast Fourier Transform is well-known, but it is available only when the discounted asset price process is a martingale. Thus, this paper aims to develop simulation methods for the non-martingale BNS model under a martingale measure.

The asset price process S in the BNS model is driven by a one-dimensional Brownian motion W and a driftless subordinator H_λ, where $\lambda > 0$ is the time-scale parameter. The square of the volatility process σ in the BNS model is given as an OU process driven by H_λ. Since we can easily simulate S for the case where H_λ is finite active, we treat only the infinite active case. In particular, we focus on the IG-OU type BNS model, a typical example of the BNS model

---

*Corresponding author

†Department of Economics, Keio University, 2-15-45 Mita, Minato-ku, Tokyo, 108-8345, Japan.
(arai@econ.keio.ac.jp)

‡Faculty of International Politics and Economics, Nishogakusha University, 6-16 Sanban-cho, Chiyoda-ku, Tokyo, 102-8336, Japan.
(y-imai@nishogakusha-u.ac.jp)

with infinite active jumps. Exact simulation methods for σ^2 in the IG-OU case have been developed by Qu et al. [6] and Sabino and Petroni [7]. The algorithm proposed in [7] is used here. Its details can be found in Appendix.

On the other hand, when the discounted asset price process is not a martingale, we need to change the underlying probability measure, denoted by $\mathbb{P}$, into an equivalent martingale measure in order to compute option prices. Note that the uniqueness of equivalent martingale measures does not hold for the BNS model because the market is incomplete. Therefore, we select the minimal martingale measure (MMM), one of the most representative equivalent martingale measures. The MMM, denoted by $\mathbb{P}^*$, is an equivalent martingale measure that appears in discussions of local risk-minimization, one of well-known optimal hedging strategies for incomplete markets. However, such background is not discussed here. Note that the MMM is the only equivalent martingale measure that can be written down concretely among equivalent martingale measures that appear in the context of hedging and risk management. More importantly, H_λ is no longer a Lévy process under $\mathbb{P}^*$; more precisely, H_λ has neither independent nor stationary increments. Thus, explicitly describing the characteristic function of S under $\mathbb{P}^*$ is impossible, implying the Carr-Madan method is unavailable.

In light of the above, we develop two simulation algorithms for S under $\mathbb{P}^*$, that is, the distribution of S_T under $\mathbb{P}^*$, where T is the maturity of our market. In the first algorithm, we define Z, the density process of $\mathbb{P}^*$, as a martingale generated by the conditional expectation under $\mathbb{P}$ of the Radon-Nikodym density $d\mathbb{P}^*/d\mathbb{P}$ and compute the distributions of Z_T and S_T under $\mathbb{P}$ separately. We make a seemingly rough approximation in the simulation of Z, but it performs unexpectedly accurately when α the drift coefficient of S is small. However, when α becomes large, this algorithm does not work. On the other hand, we simulate S under $\mathbb{P}^*$ directly in the second algorithm. This algorithm is robust with respect to the value of α. The second algorithm relies on the acceptance/rejection (A/R) scheme.

This paper is organized as follows: We provide some mathematical preparations in the next section. The two algorithms of our main objectives will be illustrated in Sections 3 and 4. Section 5 is devoted to numerical experiments, and this paper concludes in Section 6.

2 Mathematical preliminaries

Throughout this paper, we consider a financial market composed of one riskless asset and one risky asset. We denote by $T > 0$ the maturity of our market. Without loss of generality, we may assume that the interest rate is zero for simplicity. Now, the risky asset price process $S = \{S_t\}_{0 \le t \le T}$ is given as follows:

$$S_t := S_0 \exp \left\{ \int_0^t \left(\mu - \frac{1}{2}\sigma_s^2 \right) ds + \int_0^t \sigma_s dW_s + \rho H_{\lambda t} \right\}, \quad t \in [0, T], \quad (2.1)$$

where $S_0 > 0$, $\rho \leq 0$, $\mu \in \mathbb{R}$, $\lambda > 0$, $\sigma = \{\sigma_t\}_{0 \leq t \leq T}$ is the volatility process defined below, $W = \{W_t\}_{0 \leq t \leq T}$ is a one dimensional standard Brownian motion, and $H_\lambda = \{H_{\lambda t}\}_{0 \leq t \leq T}$ is a subordinator without drift, that is, a driftless non-decreasing Lévy process. Let N be the Poisson random measure of H_λ, and ν its Lévy measure. In addition, we define

$$\widetilde{N}(dt, dx) := N(dt, dx) - \nu(dx)dt,$$

which is the so-called compensated Poisson random measure of H_λ. Then, S is a solution to the following stochastic differential equation (SDE):

$$dS_t = S_{t-} \left(\alpha dt + \sigma_t dW_t + \int_0^\infty (e^{\rho x} - 1)\widetilde{N}(dt, dx) \right),$$

where

$$\alpha := \mu + \int_0^\infty (e^{\rho x} - 1)\nu(dx).$$

Let $\{\mathcal{F}_t\}_{0 \leq t \leq T}$ be the filtration generated by W and H_λ. For more details on the BNS model, see Arai et al. [1], Nicolato and Venardos [5], and Schoutens [8].

Now, we define the volatility process σ as the square root of a solution $\sigma^2 = \{\sigma_t^2\}_{0 \leq t \leq T}$ to the following SDE:

$$d\sigma_t^2 = -\lambda \sigma_t^2 dt + dH_{\lambda t}, \quad \sigma_0^2 > 0. \tag{2.2}$$

Here, the BNS model is said to be IG-OU type if the Lévy measure ν is described as

$$\nu(dx) = \frac{\lambda a}{2\sqrt{2\pi}} x^{-\frac{3}{2}} (1 + b^2 x) \exp\left\{ -\frac{1}{2}b^2 x \right\} dx, \quad x \in (0, \infty), \tag{2.3}$$

where $a, b > 0$. In fact, H_λ is infinite active since $\nu((0, \infty)) = \infty$. Throughout this paper, we consider the IG-OU type BNS model.

Next, we discuss the minimal martingale measure (MMM). First of all, we denote by $M = \{M_t\}_{0 \leq t \leq T}$ the martingale part of S. That is, it is described as

$$dM_t = S_{t-} \left(\sigma_t dW_t + \int_0^\infty (e^{\rho x} - 1)\widetilde{N}(dt, dx) \right), \quad M_0 = 0.$$

An equivalent martingale measure $\mathbb{P}^*$ is called the MMM if any square integrable $\mathbb{P}$-martingale orthogonal to M is also a $\mathbb{P}^*$-martingale. Now, we define a martingale $Z = \{Z_t\}_{0 \leq t \leq T}$ as

$$Z_t := \mathbb{E}\left[\frac{d\mathbb{P}^*}{d\mathbb{P}} | \mathcal{F}_t \right],$$

and call it the density process of $\mathbb{P}^*$. Note that $Z_0 = 1$ and $Z_T = d\mathbb{P}^*/d\mathbb{P}$. Besides, Z is a solution to the following SDE:

$$dZ_t = -Z_{t-}\Lambda_t dM_t, \tag{2.4}$$

3

where
$$\Lambda_t := \frac{1}{S_{t-}} \frac{\alpha}{\sigma_t^2 + C_2^\rho}$$
and
$$C_2^\rho := \int_0^\infty (e^{\rho x} - 1)^2 \nu(dx) = 2\rho\lambda a \left(\frac{1}{\sqrt{b^2 - 4\rho}} - \frac{1}{\sqrt{b^2 - 2\rho}} \right). \qquad (2.5)$$

Assumption 2.1. *Throughout this paper, we assume that*
$$\frac{b^2}{2} > 2 \left(\frac{1 - e^{-\lambda T}}{\lambda} \vee |\rho| \right) \quad and \quad \frac{\alpha}{e^{-\lambda T}\sigma_0^2 + C_2^\rho} > -1. \qquad (2.6)$$

The first and second conditions in (2.6) ensure the martingale property of the product process ZS and the positivity of the solution to the SDE (2.4), respectively. For more details on this matter, see [1].

Solving (2.4) under Assumption 2.1, we have
$$Z_t = \exp \left\{ - \int_0^t u_s dW_s - \frac{1}{2} \int_0^t u_s^2 ds + \int_0^t \int_0^\infty \log(1 - \theta_{s,x}) \widetilde{N}(ds, dx) \right.$$
$$\left. + \int_0^t \int_0^\infty (\log(1 - \theta_{s,x}) + \theta_{s,x})\nu(dx)ds \right\}, \qquad (2.7)$$

where
$$u_t := \Lambda_t S_{t-}\sigma_t = \frac{\alpha\sigma_t}{\sigma_t^2 + C_2^\rho} \quad and \quad \theta_{t,x} := \Lambda_t S_{t-}(e^{\rho x} - 1) = \frac{\alpha(e^{\rho x} - 1)}{\sigma_t^2 + C_2^\rho}.$$

Thus, the process $W^* = \{W_t^*\}_{0 \leq t \leq T}$ defined by
$$dW_t^* := dW_t + u_t dt$$

is a one-dimensional standard Brownian motion under $\mathbb{P}^*$. Similarly, we define
$$\nu_t^*(dx) := (1 - \theta_{t,x})\nu(dx), \qquad (2.8)$$

which is corresponding to the Lévy measure under $\mathbb{P}^*$, that is,
$$\mathbb{E}_{\mathbb{P}^*}[N(dt, dx)|\mathcal{F}_t] = \nu_t^*(dx)dt$$

holds. Roughly speaking, defining
$$\widetilde{N}^*(dt, dx) := N(dt, dx) - \nu_t^*(dx)dt,$$

we can see that stochastic integrals with respect to $\widetilde{N}^*$ are a $\mathbb{P}^*$-martingale. Note that $\nu_t^*(dx)$ depends on t and has randomness. Thus, the Carr-Madan method is not available when S is not a $\mathbb{P}$-martingale. Using W^*, $\widetilde{N}^*$ and ν_t^*, we can describe S as
$$S_t = S_0 \exp \left\{ - \int_0^t \frac{1}{2}\sigma_s^2 ds + \int_0^t \sigma_s dW_s^* \right.$$
$$\left. + \int_0^t \int_0^\infty \rho x \widetilde{N}^*(ds, dx) + \int_0^t \int_0^\infty (\rho x + 1 - e^{\rho x})\nu_s^*(dx)ds \right\}. \qquad (2.9)$$

4

3   First Algorithm

We illustrate an algorithm to compute the distributions of Z_T and S_T under $\mathbb{P}$ in order to simulate S_T under $\mathbb{P}^*$. We discretize the time interval $[0, T]$ into M time steps and denote $\delta := T/M$ and $t_k := k\delta$ for $k = 0, \ldots, M$. To simulate Z_T and S_T, we compute sequentially $\log Z_{t_{k+1}}\big|_{\mathbb{P}} S_{t_k}, \sigma_{t_k}^2$ and $\log S_{t_{k+1}}\big|_{\mathbb{P}} S_{t_k}, \sigma_{t_k}^2$ for $k = 0, \ldots, M-1$, where $X\big|_{\mathbb{Q}} Y_1, Y_2$ denotes the conditional distribution of X under the probability measure $\mathbb{Q}$ given Y_1 and Y_2.

First, we discuss how to simulate $\log S_{t_{k+1}}$ given S_{t_k} and $\sigma_{t_k}^2$. (2.1) implies that

$$\log S_{t_{k+1}} = \log S_{t_k} + \int_{t_k}^{t_{k+1}} \left(\mu - \frac{1}{2}\sigma_s^2 \right) ds + \int_{t_k}^{t_{k+1}} \sigma_s dW_s + \rho \Delta H_{\lambda t_k},$$

where $\Delta H_{\lambda t_k} := H_{\lambda t_{k+1}} - H_{\lambda t_k}$. From the view of the SDE (2.2), we can approximate $\Delta H_{\lambda t_k}$ as

$$\Delta H_{\lambda t_k} \approx \Delta \sigma_{t_k}^2 + \lambda \sigma_{t_k}^2 \delta,$$

where $\Delta \sigma_{t_k}^2 := \sigma_{t_{k+1}}^2 - \sigma_{t_k}^2$. Note that we can compute $\sigma_{t_{k+1}}^2\big|_{\mathbb{P}}\sigma_{t_k}^2$ using Algorithm 1 in [7] described in Algorithm A.1. Thus, $\log S_{t_{k+1}}$ can be approximated as follows:

$$\log S_{t_{k+1}} \approx \log S_{t_k} + \left(\mu - \frac{1}{2}\sigma_{t_k}^2 \right)\delta + \sigma_{t_k}\Delta W_{t_k} + \rho \Delta H_{\lambda t_k}$$

$$\approx \log S_{t_k} + \left(\mu - \frac{1}{2}\sigma_{t_k}^2 + \rho\lambda\sigma_{t_k}^2 \right)\delta + \sigma_{t_k}\Delta W_{t_k} + \rho \Delta \sigma_{t_k}^2,$$

where $\Delta W_{t_k} := W_{t_{k+1}} - W_{t_k}$.

Next, we mention an approximation of $\log Z_{t_{k+1}}$. By (2.7), we have

$$\log Z_{t_{k+1}} = \log Z_{t_k} - \int_{t_k}^{t_{k+1}} u_s dW_s + \int_{t_k}^{t_{k+1}} \int_0^\infty \log(1 - \theta_{s,x}) N(ds, dx)$$

$$+ \int_{t_k}^{t_{k+1}} \left(-\frac{1}{2}u_s^2 + \int_0^\infty \theta_{s,x}\nu(dx) \right) ds.$$

Now, denoting

$$K_t := \frac{\alpha}{\sigma_t^2 + C_2^\rho},$$

we have $u_t = K_t\sigma_t$ and $\theta_{t,x} = K_t(e^{\rho x} - 1)$. Thus, using the approximation

$$\int_{t_k}^{t_{k+1}} \int_0^\infty \log(1 - \theta_{s,x}) N(ds, dx) \approx \log(1 - \theta_{t_k, \Delta H_{\lambda t_k}}), \qquad (3.1)$$

we can approximate $\log Z_{t_{k+1}}$ as follows:

$$\log Z_{t_{k+1}} \approx \log Z_{t_k} - K_{t_k}\sigma_{t_k}\Delta W_{t_k} + \log(1 - \theta_{t_k, \Delta H_{\lambda t_k}})$$

$$+ \left(-\frac{1}{2}K_{t_k}^2\sigma_{t_k}^2 + K_{t_k}C_1^\rho \right)\delta,$$

where

$$C_1^\rho := \int_0^\infty (e^{\rho x} - 1)\nu(dx) = \frac{\rho\lambda a}{\sqrt{b^2 - 2\rho}}. \qquad (3.2)$$

Remark that (3.1) implements an approximation by combining all jumps occurring in the time interval $[t_k, t_{k+1}]$ into a single jump. This indicates that as the value of K_{t_k} increases, i.e., as the value of α increases, the accuracy worsens, as shown in the numerical experiments presented later.

The algorithm discussed here can be summarized as follows:

Algorithm 3.1. 1. *Set the values of the parameters* $S_0, \sigma_0, \lambda, a, b > 0, \rho \leq 0$, $\alpha \in \mathbb{R}$ *and* $T > 0$ *so that satisfy the assumption (2.6).*

2. *Set the values of the number of time steps* $M \in \mathbb{N}$ *and the number of paths* $L \in \mathbb{N}$. *Set* $\delta = T/M$.

3. *Compute* C_1^ρ *and* C_2^ρ *based on (3.2) and (2.5), respectively. Set* $\mu = \alpha - C_1^\rho$.

4. *For* $l = 1, \dots, L$, *do*

   (a) $S_{t_0,l} = S_0$, $\sigma_{t_0,l} = \sigma_0$ *and* $Z_{t_0,l} = 1$.

   (b) *For* $k = 0, \dots, M-1$, *do*

      i. *Compute* $\sigma_{t_{k+1},l}^2$ *by Algorithm A.1.*

      ii. *Generate a random number* $W \sim N(0, \delta)$.

      iii. $\log S_{t_{k+1},l} = \log S_{t_k,l} + \sigma_{t_k,l} W + \rho \Delta \sigma_{t_k,l}^2$
         $\quad + \left(\mu - \frac{1}{2}\sigma_{t_k,l}^2 + \rho\lambda\sigma_{t_k,l}^2 \right)\delta$,
         where $\Delta\sigma_{t_k,l}^2 = \sigma_{t_{k+1},l}^2 - \sigma_{t_k,l}^2$.

      iv. *Set* $K_{t_k,l} = \dfrac{\alpha}{\sigma_{t_k,l}^2 + C_2^\rho}$ *and* $\theta_{t_k,l} = K_{t_k,l}(e^{\rho\Delta H} - 1)$, *where*
         $\Delta H = \Delta\sigma_{t_k,l}^2 + \lambda\sigma_{t_k,l}^2\delta$.

      v. $\log Z_{t_{k+1},l} = \log Z_{t_k,l} - K_{t_k,l}\sigma_{t_k,l} W + \log(1 - \theta_{t_k,l})$
         $\quad + \left(-\frac{1}{2}K_{t_k,l}^2\sigma_{t_k,l}^2 + K_{t_k,l}C_1^\rho \right)\delta$.

   (c) $S_{T,l} = S_{t_M,l}$ *and* $Z_{T,l} = Z_{t_M,l}$.

4 Second Algorithm

This section develops another simulation method. Here we aim to compute S_T under $\mathbb{P}^*$ directly. As in the previous section, we discretize the time interval $[0, T]$ into M time steps. Unless otherwise noted, the notation defined in Sections 2 and 3 is also used here.

4.1 Computation of $\sigma^2_{t_{k+1}}\big|_{\mathbb{P}^*}\sigma^2_{t_k}$

We begin with the computation of $\sigma^2_{t_{k+1}}\big|_{\mathbb{P}^*}\sigma^2_{t_k}$. From the view of Proposition 3.1 of [6], (2.3) and (2.8), we have

$$
\mathbb{E}_{\mathbb{P}^*}\left[\exp\left\{-v\sigma^2_{t_{k+1}}\right\}\Big|\sigma^2_{t_k}\right]
$$

$$
= \exp\left\{-vw\sigma^2_{t_k}\right\}\exp\left\{-\frac{1}{\lambda}\int_{vw}^{v}\frac{1}{u}\int_0^{\infty}(1-e^{-ux})\nu^*_{t_k}(dx)du\right\}
$$

$$
= \exp\left\{-vw\sigma^2_{t_k}\right\}\exp\left\{-\frac{1}{\lambda}\int_{vw}^{v}\frac{1}{u}\int_0^{\infty}(1-e^{-ux})\nu(dx)du\right\}
$$

$$
\times \exp\left\{-K_{t_k}\frac{1}{\lambda}\int_{vw}^{v}\frac{1}{u}\int_0^{\infty}(1-e^{-ux})(1-e^{\rho x})\nu(dx)du\right\}
$$

$$
= \mathbb{E}\left[\exp\left\{-v\sigma^2_{t_{k+1}}\right\}\Big|\sigma^2_{t_k}\right]
$$

$$
\times \exp\left\{-K_{t_k}\int_{vw}^{v}\frac{1}{u}\int_0^{\infty}(1-e^{-ux})(1-e^{\rho x})\frac{a\beta}{\sqrt{2\pi}}x^{-\frac{1}{2}}e^{-\beta x}dxdu\right\}
$$

$$
\times \exp\left\{-K_{t_k}\int_{vw}^{v}\frac{1}{u}\int_0^{\infty}(1-e^{-ux})(1-e^{\rho x})\frac{a}{2\sqrt{2\pi}}x^{-\frac{3}{2}}e^{-\beta x}dxdu\right\} \quad (4.1)
$$

for any $v > 0$, where $w := e^{-\lambda\delta}$, $\beta := b^2/2$, and ν^*_t is decomposed into

$$
\nu^*_t(dx) = (1 - K_t(e^{\rho x} - 1))\,\nu(dx) = (1 + K_t(1 - e^{\rho x}))\,\nu(dx).
$$

For $m =$1,3, converting u and x into $y = ux/v$ and $z = v/u$, we obtain

$$
\int_{vw}^{v}\frac{1}{u}\int_0^{\infty}(1-e^{-ux})(1-e^{\rho x})x^{-\frac{m}{2}}e^{-\beta x}dxdu
$$

$$
= \int_0^{\infty}(1-e^{-vy})\int_1^{\frac{1}{w}}(1-e^{\rho yz})(yz)^{-\frac{m}{2}}e^{-\beta yz}dzdy,
$$

and define a function f_m and a constant C_m as

$$
f_m(y) := \int_1^{\frac{1}{w}}(1-e^{\rho yz})(yz)^{-\frac{m}{2}}e^{-\beta yz}dz, \quad y > 0,
$$

and

$$
C_m := \int_0^{\infty}f_m(y)dy = \frac{5-3m}{2}\sqrt{\pi}\lambda\delta\left(\beta^{\frac{m}{2}-1} - (\beta-\rho)^{\frac{m}{2}-1}\right).
$$

Since $f_m(y) > 0$ for any $y > 0$, $\widetilde{f}_m(y) := f_m(y)/C_m$ gives a probability density function. Consequently, we obtain

$$
\mathbb{E}_{\mathbb{P}^*}\left[\exp\left\{-v\sigma^2_{t_{k+1}}\right\}\Big|\sigma^2_{t_k}\right]
$$

$$
= \mathbb{E}\left[\exp\left\{-v\sigma^2_{t_{k+1}}\right\}\Big|\sigma^2_{t_k}\right]\exp\left\{-K_{t_k}\frac{a\beta}{\sqrt{2\pi}}\int_0^{\infty}(1-e^{-vy})C_1\widetilde{f}_1(y)dy\right\}
$$

$$\times \exp\left\{-K_{t_k}\frac{a}{2\sqrt{2\pi}}\int_0^\infty (1-e^{-vy})C_3\widetilde{f}_3(y)dy\right\}$$

for any $v > 0$, which implies that $\sigma^2_{t_{k+1}}\big|_{\mathbb{P}*}\sigma_{t_k}$ is given by

$$\sigma^2_{t_{k+1}}\big|_{\mathbb{P}}\sigma_{t_k} + \sum_{i=1}^{N_1} X_i^{(1)} + \sum_{i=1}^{N_3} X_i^{(3)}, \tag{4.2}$$

where N_1 and N_3 are random variables following the Poisson distribution with parameters $K_{t_k}\dfrac{a\beta}{\sqrt{2\pi}}C_1$ and $K_{t_k}\dfrac{a}{2\sqrt{2\pi}}C_3$, respectively, and $\{X_i^{(1)}\}_{i\geq 1}$ and $\{X_i^{(3)}\}_{i\geq 1}$ are i.i.d. sequences with density functions $\widetilde{f}_1$ and $\widetilde{f}_3$, respectively.

4.2 Computation of $X_i^{(m)}$

In order to generate random variables $X_i^{(m)}$, $m =1,3$, we use the acceptance/rejection (A/R) scheme. See Glasserman [4] for details on the A/R scheme. Define

$$g_1(y) := \frac{1}{C_1}y^{-\frac{1}{2}}\int_1^{\frac{1}{w}} z^{-\frac{1}{2}}dz e^{-\beta y} = \frac{2}{C_1}y^{-\frac{1}{2}}e^{-\beta y}\left(\frac{1}{\sqrt{w}}-1\right), \quad y > 0.$$

We have then $\widetilde{f}_1(y) \leq g_1(y)$ for any $y > 0$. Moreover, normalizing g_1 as

$$\widetilde{g}_1(y) := \frac{g_1(y)}{\int_0^\infty g_1(z)dz} = \sqrt{\frac{\beta}{\pi}}y^{-\frac{1}{2}}e^{-\beta y}, \quad y > 0,$$

which is the density function of the Gamma distribution with shape parameter $1/2$ and scale parameter $1/\beta$. Similarly, we define

$$g_3(y) := \frac{|\rho|}{C_3}y^{-\frac{1}{2}}\int_1^{\frac{1}{w}} z^{-\frac{1}{2}}dz e^{-\beta y} = \frac{2|\rho|}{C_3}y^{-\frac{1}{2}}e^{-\beta y}\left(\frac{1}{\sqrt{w}}-1\right), \quad y > 0.$$

Since $1 - e^{\rho yz} \leq |\rho|yz$, $\widetilde{f}_3(y) \leq g_3(y)$ holds for any $y > 0$, and the normalization

$$\widetilde{g}_3(y) := \frac{g_3(y)}{\int_0^\infty g_3(z)dz}, \quad y > 0,$$

coincides with $\widetilde{g}_1$. Under the above preparation, we describe an algorithm for computing $X_i^{(m)}$, $m =1,3$.

Algorithm 4.1.   *1. Generate a random variable Y following the Gamma distribution with shape parameter $1/2$ and scale parameter $1/\beta$.*

2. Generate a random variable U uniformly distributed on $[0,1]$.

3. If $U \leq \widetilde{f}_m(Y)/g_m(Y)$ holds, then $X_i^{(m)} = Y$. Otherwise, reject Y and go back to 1.

8

Remark 1. By Section 2.2.2 of [4], the probability that Y is accepted in 3 of Algorithm 4.1, affecting computation time, is given by

$$\frac{1}{\int_0^\infty g_m(z)dz} = \begin{cases} \frac{\lambda\delta}{2\left(e^{\frac{1}{2}\lambda\delta}-1\right)}\frac{\sqrt{\beta-\rho}-\sqrt{\beta}}{\sqrt{\beta-\rho}}, & \text{for } m=1, \\[2ex] \frac{\lambda\delta}{|\rho|\left(e^{\frac{1}{2}\lambda\delta}-1\right)}\left(\sqrt{\beta(\beta-\rho)}-\beta\right), & \text{for } m=3. \end{cases}$$

We further summarize the algorithm for computing $\sigma^2_{t_{k+1}}\big|_{\mathbb{P}^*}\sigma^2_{t_k}$ based on (4.2).

Algorithm 4.2. *1. Generate a random variable N_1 following the Poisson distribution with parameter $K_{t_k}\dfrac{a\beta}{\sqrt{2\pi}}C_1$.*

2. If $N_1 = 0$, set $\Sigma_1 = 0$ and jump to 4. Otherwise, for $i = 1,\ldots,N_1$, generate a random variable $X_i^{(1)}$ by using Algorithm 4.1.

3. $\Sigma_1 = \sum_{i=1}^{N_1} X_i^{(1)}$.

4. Generate a random variable N_3 following the Poisson distribution with parameter $K_{t_k}\dfrac{a}{2\sqrt{2\pi}}C_3$.

5. If $N_3 = 0$, set $\Sigma_3 = 0$ and jump to 7. Otherwise, for $i = 1,\ldots,N_3$, generate a random variable $X_i^{(3)}$ by using Algorithm 4.1.

6. $\Sigma_3 = \sum_{i=1}^{N_3} X_i^{(3)}$.

7. Compute $\sigma^2_{t_{k+1}}\big|_{\mathbb{P}}\sigma_{t_k}$ by Algorithm A.1.

8. Set $\sigma^2_{t_{k+1}} = \sigma^2_{t_{k+1}}\big|_{\mathbb{P}}\sigma_{t_k} + \Sigma_1 + \Sigma_3$.

4.3   Computation of $\log S_{t_{k+1}}\big|_{\mathbb{P}^*}S_{t_k}$

By a similar argument with Section 3 and (2.9), we approximate $\log S_{t_{k+1}}$ as follows:

$$\begin{aligned} \log S_{t_{k+1}} &= \log S_{t_k} - \int_{t_k}^{t_{k+1}} \frac{1}{2}\sigma_s^2 ds + \int_{t_k}^{t_{k+1}} \sigma_s dW_s^* \\ &\quad + \int_{t_k}^{t_{k+1}}\int_0^\infty \rho x N(ds,dx) + \int_{t_k}^{t_{k+1}}\int_0^\infty (1 - e^{\rho x})\,\nu_{t_k}^*(dx)ds \\ &\approx \log S_{t_k} - \frac{1}{2}\sigma_{t_k}^2\delta + \sigma_{t_k}\Delta W_{t_k}^* + \rho\Delta H_{\lambda t_k} - C_1^\rho\delta + K_{t_k}C_2^\rho\delta, \\ &\approx \log S_{t_k} + \left(-\frac{1}{2}\sigma_{t_k}^2 + \rho\lambda\sigma_{t_k}^2 - C_1^\rho + K_{t_k}C_2^\rho\right)\delta + \sigma_{t_k}\Delta W_{t_k}^* + \rho\Delta\sigma_{t_k}^2, \end{aligned}$$

where $\Delta W^*_{t_k} := W^*_{t_{k+1}} - W^*_{t_k}$ and

$$\Delta H_{\lambda t_k} := H_{\lambda t_{k+1}} - H_{\lambda t_k} \approx \Delta \sigma^2_{t_k} + \lambda \sigma^2_{t_k} \delta.$$

The algorithm for $\log S_{t_{k+1}}$ is described as follows:

Algorithm 4.3. *1. Conduct 1-3 in Algorithm 3.1.*

2. For $l = 1, \ldots, L$, do

    (a) $S_{t_0,l} = S_0$ and $\sigma_{t_0,l} = \sigma_0$.

    (b) For $k = 0, \ldots, M - 1$, do

        i. $K_{t_k,l} = \dfrac{\alpha}{\sigma^2_{t_k,l} + C^\rho_2}.$

        ii. Generate a random number $W \sim N(0, \delta)$.

        iii. Compute $\sigma^2_{t_{k+1},l}$ by Algorithm 4.2.

        iv. $\log S_{t_{k+1},l} = \log S_{t_k,l} + \left(-\dfrac{1}{2}\sigma^2_{t_k,l} + \rho\lambda\sigma^2_{t_k,l} - C^\rho_1 + K_{t_k,l}C^\rho_2 \right)\delta$
            $+ \sigma_{t_k,l}W + \rho(\sigma^2_{t_{k+1},l} - \sigma^2_{t_k,l}).$

    (c) $S_{T,l} = S_{t_M,l}.$

5 Numerical results

We execute numerical experiments for Algorithms 3.1 and 4.3. Throughout our experiments, we use the parameter set calibrated in [5], that is, $S_0 = 468.40$, $\sigma^2_0 = 0.0041$, $\lambda = 2.4958$, $a = 0.0872$, $b = 11.98$, $\rho = -4.7039$, and vary the drift coefficient α from 0.01 to 100. Furthermore, fix $T = 1$ and set the number of time steps M and the number of paths L to 100 and 100,000, respectively. The values of M and L are varied depending on the situation. Note that Assumption 2.1 is always satisfied in the above setting. Here, we simulate the values of $\mathbb{E}_{\mathbb{P}^*}[S_T]$ and $\mathbb{E}_{\mathbb{P}^*}[\overline{S}_T]$, where

$$\overline{S}_T := \frac{1}{T} \int_0^T S_t dt,$$

and confirm if the algorithms developed here are available to compute plain vanilla and Asian options. As for $\mathbb{E}_{\mathbb{P}^*}[S_T]$, we compute

$$\frac{1}{L} \sum_{l=1}^L S_{T,l} Z_{T,l} \quad \text{and} \quad \frac{1}{L} \sum_{l=1}^L S_{T,l}$$

by Algorithms 3.1 and 4.3, respectively. In addition, $\mathbb{E}_{\mathbb{P}^*}[\overline{S}_T]$ is simulated computing

$$\frac{1}{L(M+1)} \sum_{l=1}^L \sum_{k=0}^M S_{t_k,l} Z_{t_k,l} \quad \text{and} \quad \frac{1}{L(M+1)} \sum_{l=1}^L \sum_{k=0}^M S_{t_k,l}$$

10

by Algorithms 3.1 and 4.3, respectively. Since $S_0 = \mathbb{E}_{\mathbb{P}^*}[S_T] = \mathbb{E}_{\mathbb{P}^*}[\overline{S}_T]$, we define

$$\text{Error} := \frac{S_0 - \text{simulation result}}{S_0} \times 100$$

and calculate it to evaluate the performance of our experiments. Tables 1 and 2 give the results of Algorithms 3.1 and 4.3, respectively. Note that the top and bottom rows of the fourth columns in Tables 1 and 2 represent the errors for the simulations of $\mathbb{E}_{\mathbb{P}^*}[S_T]$ and $\mathbb{E}_{\mathbb{P}^*}[\overline{S}_T]$, respectively. All the experiments have been conducted using Matlab R2022a with MacBook Air(2022) Apple M2 CPU, 24GB.

α	M	L	Error %	Time sec.
0.01	100	100,000	0.012664767 0.066203396	30.785927
0.05	100	100,000	0.649362749 0.802834253	30.699075
0.1	100	100,000	0.795886643 0.874496856	30.954113
0.1	500	100,000	0.309957572 0.339321885	149.456563
1	10,000	10,000	95.28883837 96.11253977	349.566446

Table 1: Results of Algorithm 3.1.

α	M	L	Error %	Time sec.
0.1	100	100,000	-0.110181005 -0.059691138	82.117931
1	100	100,000	-0.467036359 -0.32856946	185.662105
5	100	100,000	-2.944984824 -2.125720502	383.790356
5	500	100,000	-0.482692379 -0.337696412	604.576945
10	100	100,000	-6.645643187 -4.989554592	537.079512
10	500	100,000	-1.163916967 -0.887464098	740.789968
10	1,000	100,000	-0.825495611 -0.577846827	1042.507264
100	20,000	10,000	-0.791441992 -0.455007171	1341.982969

Table 2: Results of Algorithm 4.3.

As for Algorithm 3.1, when $\alpha \leq 0.1$, the errors are less than 1%, which ensures sufficient accuracy. As α increases, the accuracy becomes worse, but when $\alpha = 0.1$, increasing M to 500 improves accuracy. Algorithm 3.1 is accurate for small α despite including the rough approximation in (3.1). However, when $\alpha = 1$, the accuracy does not improve even if M is increased. In other words, it does not work for large α. On the other hand, Algorithm 4.3 is sufficiently accurate for larger values of α if M is increased. For example, even in the extreme case of $\alpha = 100$, the errors can be kept below 1% taking $M = 20{,}000$. Note that when $M \geq 10{,}000$, we reduce L to 10,000 to save computation time. However, it is a little bit more time-consuming than Algorithm 3.1. The use of the A/R scheme probably causes this. Indeed, the acceptance probabilities in our experiments with $\delta = 0.01$ are 0.0311 for $m = 1$ and 0.978 for $m = 3$; that is, The A/R scheme for $m = 3$ is very efficient but not efficient for $m = 1$. Finally, the above results show that Algorithms 3.1 and 4.3 are useful for calculating prices of plain vanilla and Asian options.

6 Concluding remarks

Two simulation algorithms for the non-martingale IG-OU type BNS model under the MMM have been developed in this paper. Algorithm 3.1 only works when the drift coefficient α is small, but it is faster than Algorithm 4.3. On the other hand, Algorithm 4.3 is sufficiently accurate by increasing M the number of time steps for any value of α. Using the simulation algorithms developed here, option price calculations for the BNS model can also be implemented easily. Hence, we can develop supervised deep learning to compute option prices by creating training samples with our simulation algorithms. We left it to future work.

A Computation of $\sigma^2_{t_{k+1}} \big|_{\mathbb{P}} \sigma_{t_k}$

We illustrate the algorithm for $\sigma^2_{t_{k+1}} \big|_{\mathbb{P}} \sigma_{t_k}$ developed by Sabino and Petroni [7]. Suppose the values of parameters $\lambda, a, b > 0$ are given. Fix $T > 0$ and $M \in \mathbb{N}$, and set $\delta = T/M$ and $t_k = k\delta$ for $k = 0, \ldots, M$. Suppose that the value of σ_{t_k} is given for some $k \in \{0, \ldots, M-1\}$. The following provides an algorithm for computing $\sigma^2_{t_{k+1}}$ under $\mathbb{P}$.

Algorithm A.1 (Algorithm 1 in [7]).   *1. Generate a random variable X_1 following the inverse Gaussian distribution with scale parameter $a(1 - \sqrt{w})/b$ and shape parameter $a^2(1 - \sqrt{w})^2$, where $w = e^{-\lambda\delta}$.*

2. Generate a random variable N following the Poisson distribution with parameter $ab(1 - \sqrt{w})$.

3. If $N = 0$, set $X_2 = 0$ and jump to 5. Otherwise, for $n = 1, \ldots, N$, do

   (a) Generate a random variable U distributed uniformly on $[0, 1]$.

(b) *Set* $V = \left(1 + 2 \left(\frac{1}{\sqrt{w}} - 1 \right) U \right)^2$, *and* $\widetilde{\beta} = \beta V$.

(c) *Generate a random variable* J_n *following the Gamma distribution with shape parameter* $1/2$ *and scale parameter* $1/\widetilde{\beta}$.

4. $X_2 = \displaystyle\sum_{n=1}^{N} J_n$.

5. $\sigma_{t_{k+1}}^2 = w\sigma_{t_k}^2 + X_1 + X_2$.

Acknowledgments

References

[1] Arai, T., Imai, Y. and Suzuki, R. (2017). Local risk-minimization for Barndorff-Nielsen and Shephard models, Finance & Stochastics, 21 , pp.551-592.

[2] Barndorff-Nielsen, O. E., & Shephard, N. (2001). Modelling by Lévy processes for financial econometrics. In Lévy processes (pp.283-318). Birkhäuser, Boston, MA.

[3] Barndorff-Nielsen, O. E., & Shephard, N. (2001). Non-Gaussian Ornstein-Uhlenbeck-based models and some of their uses in financial economics. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2), pp.167-241.

[4] Glasserman, P. (2004). Monte Carlo methods in financial engineering (Vol. 53, pp. xiv+-596). New York: Springer.

[5] Nicolato, E. and Venardos, E. (2003). Option pricing in stochastic volatility models of the Ornstein-Ühlenbeck type, Mathematical Finance, 13 , pp.445-466.

[6] Qu, Y., Dassios, A., & Zhao, H. (2021). Exact simulation of Ornstein-Uhlenbeck tempered stable processes. Journal of Applied Probability, 58(2), pp.347-371.

[7] Sabino, P., & Petroni, N. C. (2022). Fast simulation of tempered stable Ornstein-Ühlenbeck processes. Computational Statistics, 37(5), pp.2517-2551.

[8] Schoutens, W. (2003). Lévy processes in finance: pricing financial derivatives, Wiley.