Fast exact simulation of the first passage of a tempered stable subordinator across a non-increasing function

Jorge Ignacio González Cázares[*]   Feng Lin[†]   Aleksandar Mijatović[‡]

April 30, 2024

Abstract

We construct a fast exact algorithm for the simulation of the first-passage time, jointly with the undershoot and overshoot, of a tempered stable subordinator over an arbitrary non-increasing absolutely continuous function. We prove that the running time of our algorithm has finite exponential moments and provide bounds on its expected running time with explicit dependence on the characteristics of the process and the initial value of the function. The expected running time grows at most cubically in the stability parameter (as it approaches either 0 or 1) and is linear in the tempering parameter and the initial value of the function. Numerical performance, based on the implementation in the dedicated GitHub repository, exhibits a good agreement with our theoretical bounds. We provide numerical examples to illustrate the performance of our algorithm in Monte Carlo estimation.

Key words: exact simulation; subordinator; first passage of subordinator; overshoot and undershoot of a subordinator; complexity.

AMS Subject Classification 2020: 60G51, 65C05 (Primary); 62E15, 60E07 (Secondary).

Contents

[*]Universidad Nacional Autónoma de México, IIMAS, `jorge.gonzalez@sigma.iimas.unam.mx`

[†]University of Warwick, `feng.lin.1996@outlook.com`

[‡]University of Warwick & The Alan Turing Institute, `a.mijatovic@warwick.ac.uk`

1   Introduction

The study of the first-passage event across a barrier is a classical subject that has long been of interest for many stochastic processes, including Lévy processes. For instance, first-passage events describe the ruin event (i.e. the ruin time and penalty function) in risk theory [5, Ch. 12], are used to describe the law of the steady-state waiting time and workload in queuing theory [3] and arise in the probabilistic representation of the solutions to fractional partial differential equations (FPDEs) (see e.g. [18] and the references therein),

as well as in financial mathematics (in payoffs of barrier-type derivative securities). In all such areas, simulation algorithms and Monte Carlo methods are of great interest and widely used (see [4, Ex. 5.15]). However, direct biased Monte Carlo methods, based on random walk approximations of the crossing time, may lead to errors that are often difficult and computationally expensive to control, see e.g. Subsection 3.3.1 below. Fast exact simulation of the crossing time of a level by a subordinator is thus needed for a stable Monte Carlo algorithm in these contexts. For example, in [22], exact simulation of the first-passage time of a stable subordinator was used to solve numerically an FPDE with a Caputo fractional derivative [21]. Generalising [21] beyond Caputo fractional derivatives requires an exact simulation algorithm for more general subordinators, along with an *a priori* bound on its expected running time. The exact simulation algorithm in [8], applicable to a broad class of subordinators, typically turns out to have an infinite expected running time (see Subsection 2.7 below), making it hard to exploit the general algorithm in [8] in this context.

The present paper develops an exact simulation algorithm for the first-passage event of a tempered stable subordinator across a non-increasing function, such that its random running time has finite exponential moments with explicit (in terms of model parameters) control over the expected running time. While our algorithm is widely applicable to problems in applied probability discussed above when the underlying model is a finite variation tempered stable process (cf. [20] for examples in financial mathematics), our main interest in this algorithm stems from our aim to generalise [22] to tempered fractional derivatives and possibly, in future work, to more general non-local operators using ideas from [8, 9].

1.1 The complexity of the main exact simulation algorithm

A subordinator $S = (S_t)_{t \in \mathbb{R}_+}$ (where $\mathbb{R}_+ := [0, \infty)$) is a non-decreasing process with independent and stationary increments and right-continuous paths, started at zero $S_0 = 0$. It is well known that $S_t = \mu t + \sum_{s \in (0,t]} J_s$, where $\mu \in \mathbb{R}_+$ is a non-negative drift parameter and the (positive) jumps $\{(s, J_s) : s > 0, J_s > 0\}$ form a Poisson point process with mean measure $\mathrm{d}t \otimes \nu(\mathrm{d}x)$ on $(0, \infty) \times (0, \infty)$, where ν is the *Lévy measure* of S, satisfying $\int_0^\infty \min\{1, x\}\nu(\mathrm{d}x) < \infty$ (see details in [25, Def. 8.2]). In this work we consider a *driftless tempered stable subordinator* S: $\mu = 0$ and $\nu(\mathrm{d}x) := \theta(\alpha/\Gamma(1 - \alpha))\mathrm{e}^{-qx}x^{-\alpha-1}\mathrm{d}x$ for some (fixed) $\theta > 0$, $\alpha \in (0, 1)$ and $q \in [0, \infty)$. The first-passage event of S is described by the random element

$$\big(\tau_b, S_{\tau_b-}, S_{\tau_b}\big), \qquad \text{where} \quad \tau_b := \inf\{t > 0 : S_t > b(t)\}$$

is the first crossing time of S over a non-increasing absolutely continuous function $b : [0, \infty) \to [0, \infty)$, with $b(0) \in (0, \infty)$. Since $S_0 = 0$, we have $0 < \tau_b < \infty$ almost surely and the variable $S_{\tau_b-} := \lim_{t \uparrow \tau_b} S_t$, referred to as the *undershoot* in this paper, gives the position of the subordinator S just before it crosses the function b. The position of the subordinator S at first crossing time τ_b over b is given by S_{τ_b}. TSFFP-Alg below, which is the main simulation algorithm of this paper, samples from the law of the vector $(\tau_b, S_{\tau_b-}, S_{\tau_b})$.

Denote by $\mathcal{T}$ the (random) running time of our main algorithm TSFFP-Alg. Here and throughout we assume that elementary operations such as addition, multiplication and evaluation of elementary functions sin, cos, exp all have constant computational cost. Further, we assume all operations use floating point precision and denote by $N \in \mathbb{N}$ the desired number of precision bits of the output. Then $\mathcal{T}$ has finite exponential moments (up to some order) and its mean is bounded above by

$$\kappa_{TSFFP}(1 + qb(0)/\alpha)((1 - \alpha)^{-3} + |\log \alpha| + \log N)/\alpha,$$

where the constant κ_{TSFFP} does not depend on $\alpha \in (0,1)$, $q \in [0,\infty)$ or b (see Corollary 2.2 below). Moreover, κ_{TSFFP} can be made explicit in terms of the costs of elementary operations listed above.

Since the law of the vector $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ degenerates in any of the limiting regimes, $\alpha \uparrow 1$, $\alpha \downarrow 0$, $q \to \infty$ or $b(0) \to \infty$, the deterioration in the performance of the algorithm in these regimes is expected. Indeed, the subordinator S converges weakly to a linear drift as $\alpha \uparrow 1$ (see the stable characteristic function in (2.1) below), making the law of the undershoot S_{τ_b-} increasingly harder to simulate from. If instead $\alpha \downarrow 0$ or $q \to \infty$, then the subordinator S converges weakly to the trivial process 0 for which the first-passage time τ_b is infinite. Finally, if $b \equiv b(0)$ increases, then the crossing time τ_b increases, extending the running time of several loops in the algorithm. The main point of the formula in the display above is that it provides an explicit insight into the deterioration of the performance of the algorithm in any of the limiting regimes. There is a good match between this theoretical upper bound on the performance and what we observe through the implementation found in the repository [15], see Subsection 3.1 below for details.

The key step in the exact simulation of the first-passage triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ for a general tempering parameter $q \in [0,\infty)$ (TSFFP-Alg below) is the stable case $q = 0$, given in SFP-Alg below. The general case $q \in (0,\infty)$ in TSFFP-Alg requires only an additional accept-reject step based on the Esscher transform described below. In turn, the simulation of the first-passage event of a stable process in SFP-Alg is essentially reduced to the simulation of the undershoot S_{τ_b-} on the event $\{S_{\tau_b-} < b(\tau_b)\}$ when the first passage did not occur continuously (i.e. by *creeping*[1], which occurs on the event $\{S_{\tau_b-} = b(\tau_b)\}$, cf. Subsection 4.1 below). Sampling the undershoot efficiently, which turns out to be deceivingly hard (see Subsection 2.5 below), is the main technical contribution of this paper. Although the density of the undershoot S_{τ_b-}, conditional on $\{\tau_b = t\}$, admits a semi-analytic explicit form (see (4.5) below), it is unbounded and does not lend itself well to direct accept-reject methods. Our main strategy to circumvent this issue is to break up the state space $(0, b(t))$ of $S_{\tau_b-}|\{\tau_b = t\}$ into subintervals, depending on the values of the (possibly random) parameters, and on each subinterval use an appropriate accept-reject algorithm. The algorithm in [11], see LC-Alg in Appendix C.2 below, plays a key role precisely in the subinterval where the density of S_{τ_b-} decreases sharply from a very high value to super-exponentially small values, making the standard accept-reject bounds hard to find.

Most algorithms developed and used in this paper are based on a combination of two components: (I) a numerical search step in the form of binary search (also known as the bisection method), possibly followed by an application of the Newton–Raphson method (rigorous care is taken to verify the assumptions that guarantee quadratic convergence of the Newton–Raphson algorithm) and (II) rejection (or accept-reject) sampling. For both components, we do a careful complexity analysis accounting for all computational operations. This is crucial because parameter values themselves are often random, requiring control over their distribution as well as the impact on the computational complexity when these random parameters take extreme values. Note that the infinite expected running time of the algorithm in [8] arises essentially due to an accept-reject step having a random acceptance probability with large mass close to zero, see Subsection 2.7 below for details.

---

[1]Note that creeping occurs with positive probability $\mathbb{P}[S_{\tau_b-} = b(\tau_b)] > 0$ when the boundary function b is not constant, even though the tempered stable subordinator has no drift, see [7, 8].

1.2   Comparison with the literature

Exact simulation of the first-passage event of a tempered stable subordinator is a central topic of two articles: [8] and [9]. [8] was the first to develop a simulation algorithm for the first-passage event for a wide class of subordinators over a general non-decreasing absolutely continuous boundary functions b. The case of the tempered stable subordinator plays a prominent role in [8], because the entire general class of subordinators considered in [8] can be viewed as a modification of the tempered stable case. Indeed, the main algorithm in [8] relies on this structure. As discussed above, the algorithm for the tempered stable first-passage event in [8] typically has infinite expected running time (see Subsection 2.7), making it hard to use in Monte Carlo estimation.

[9] consider a tempered stable subordinator and a constant function b. They develop an exact simulation algorithm [9, Alg. 4.1] for the pair $(\tau_b, S_{\tau_b} - b(\tau_b))$. In this case, the problem is first reduced to the stable case via the Esscher transform. Then the constant barrier makes the density of $(\tau_b, S_{\tau_b} - b)$ explicit with a stable scaling property. The scaling property in turn allows for a change of variables (in the form of the variable $r = y^{-\alpha/(1-\alpha)} t^{1/(1-\alpha)}$ in [9, l.11, §4.2]) that makes it easy to sample from the resulting density. This technique, which depends on the explicit density of the random vector $(\tau_b, S_{\tau_b} - b)$, does not generalise easily to a non-constant function b, including a linear function $b(t) = a_0 - a_1 t$ with $a_1 > 0$. Part of the difficulty in generalising such methodology is that a new possibility arises: the subordinator may *creep* (i.e. cross continuously [7, 8]) through the curve b, in which case $S_{\tau_b-} = b(\tau_b) = S_{\tau_b}$. Moreover, the probability of creeping is not uniform in time even for linear functions (see Proposition 4.1(b) below), making it hard to compute even the mass of the absolutely continuous component of the joint law.

1.3   Organisation of the paper

The main algorithms and the results on their respective computational complexities are provided in Section 2. In Section 3 we present numerical evidence for the theoretical bounds on the computational complexities of our algorithms. We also present two numerical applications of our simulation algorithms, using Monte Carlo estimation, to solve fractional partial differential equations and to price barrier options. The GitHub repository [15] contains the implementation (in Julia) of the algorithms in this paper, used in Section 3. Finally, the proofs of all our results on the validity and computational complexities of our algorithms are given in Section 4. See [16] for a short YouTube presentation on Prob-AM channel discussing the algorithm.

2   Sampling the first passage of a tempered stable subordinator

For any stability index $\alpha \in (0,1)$, intensity $\theta \in (0,\infty)$ and tempering parameter $q \in \mathbb{R}_+$, let $S = (S_t)_{t \in \mathbb{R}_+}$ be, under $\mathbb{P}_q$ (with expectation operator $\mathbb{E}_q$), a driftless tempered stable subordinator started at 0 with Laplace transform and Lévy measure

$$\mathbb{E}_q[\mathrm{e}^{-uS_1}] = \exp\left((q^\alpha - (u+q)^\alpha)\theta\right), \ \ u \geqslant 0 \ \ \& \ \ \nu_q(\mathrm{d}x) = (\alpha/\Gamma(1-\alpha))\theta\mathrm{e}^{-qx}x^{-\alpha-1}\mathrm{d}x, \ \ x > 0. \quad (2.1)$$

Note that, under $\mathbb{P} := \mathbb{P}_0$, S is a stable subordinator with Lévy measure $\nu := \nu_0$.

Figure 2.1: Dependence of algorithms ($B \leftarrow A$ means algorithm A calls algorithm B)
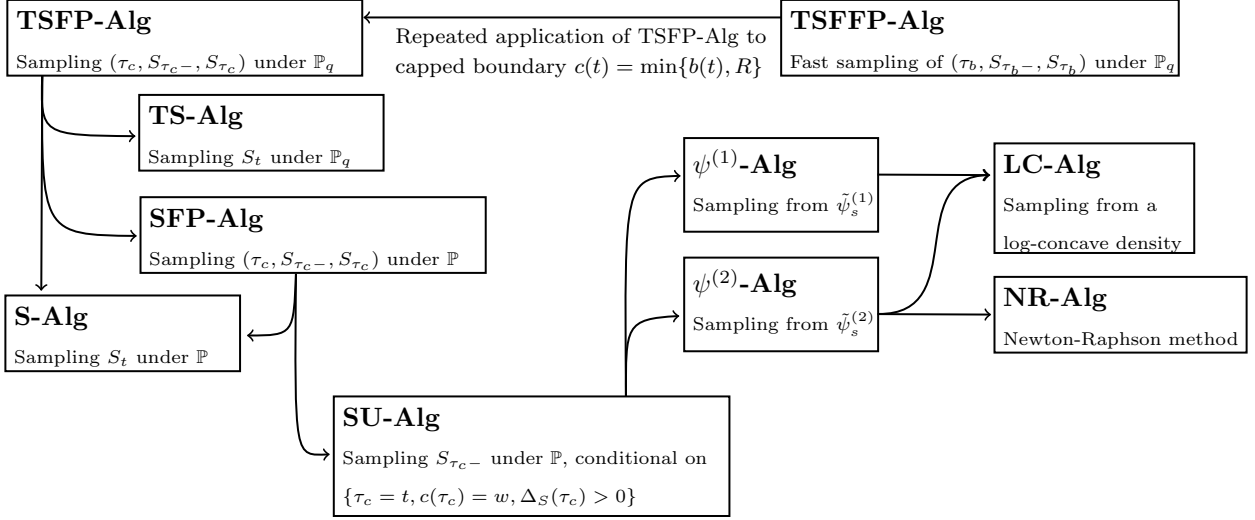
2.1 Dependence of the algorithms for the simulation of $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}_q$

Figure 2.1 above summarises the dependence between the algorithms used to simulate the triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ for the tempered stable subordinator S. Algorithm TSFFP-Alg essentially reduces the problem to the same problem in the stable case, dealt with by SFP-Alg. The hardest step of the stable first-passage algorithm SFP-Alg consists of the simulation of the undershoot S_{τ_b-}, performed by SU-Alg, which in turn relies on $\psi^{(1)}$-Alg and $\psi^{(2)}$-Alg. These two rejection-sampling algorithms are tailored to the specific densities $\tilde\psi_s^{(1)}$ and $\tilde\psi_s^{(2)}$ that arise in our problem (see Proposition 4.2 for the definition). Upon noting that a certain function (related to $\tilde\psi_s^{(1)}$) is log-concave, $\psi^{(1)}$-Alg essentially only calls LC-Alg, a very fast rejection-sampling procedure for log-concave densities. $\psi^{(2)}$-Alg requires a partitioning of the state space, which relies on numerical inversion of elementary functions, and finely tuned rejection sampling. The details are given in the remainder of this section.

2.2 Sampling the first passage of a tempered stable subordinator

Before presenting our simulation algorithm TSFP-Alg, we give a brief intuitive account. The measures $\mathbb{P}_q$ (for $q \geqslant 0$) are known to be equivalent to each other (see Appendix B). In fact, the law of the trajectory $(S_t)_{t \in [0,T]}$ (for any $T > 0$) under $\mathbb{P}_q$ is equal to the law under $\mathbb{P}_0$ weighted by the change-of-measure $\exp(-qS_T + q^\alpha\theta T)$. Since this change-of-measure is bounded, it can be used for rejection sampling of $(S_t)_{t \in [0,T]}$ under $\mathbb{P}_q$: propose a path on $[0,T]$ under $\mathbb{P}_0$ and accept it with probability e^{-qS_T}, which is on average equal to $\exp(-q^\alpha\theta T)$, see Lemma B.1 in Appendix B below for details. Thus, it is natural to expect that the simulation of the first-passage triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}_q$ can essentially be reduced to the case $q = 0$ on the stochastic interval $[0, \tau_b]$. However, since τ_b may have unbounded support, the change-of-measure $\exp(-qS_{\tau_b} + q^\alpha\theta\tau_b)$ may also be unbounded, making a direct rejection-sampling algorithm for $\mathbb{P}_q$ infeasible. TSFP-Alg makes use of the change-of-measure idea by introducing a time grid and sampling S on the grid until the compact interval in which τ_b lies is identified.

TSFP-Alg makes use of S-Alg and TS-Alg, well-known simple and fast procedures for sampling stable and tempered stable marginals, respectively (see Appendix C.1 below). In lines 2–7, TSFP-Alg consecu-

---

Algorithm 1 (TSFP-Alg) Tempered Stable First-Passage Algorithm: samples $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}_q$

Require: Parameters $\alpha \in (0,1)$, $\theta > 0$, $q \geqslant 0$ and function $b : [0, \infty) \to [0, \infty)$

1: Set $t_* \leftarrow (2qb(0) + 1 - 2^{-\alpha})/((2^\alpha - 1)q^\alpha\theta)$, $T \leftarrow 0$, $U \leftarrow 0$ and $c(\cdot) \leftarrow b(\cdot)$

2: **repeat**

3:     Sample s from the law of S_{t_*} under $\mathbb{P}_q$ via TS-Alg

4:     **if** $s < c(t_*)$ **then**

5:         Update $T \leftarrow T + t_*$, $U \leftarrow U + s$ and $c(\cdot) \leftarrow c(\cdot + t_*) - s$

6:     **end if**

7: **until** $s \geqslant c(t_*)$

8: **repeat**

9:     Sample (τ, u, v) via SFP-Alg with α, θ, t_*, c ▷ Law of $(\tau_c, S_{\tau_c-}, S_{\tau_c})|\{\tau_c \leqslant t_*\}$ under $\mathbb{P}$

10:     Sample w from the law of $S_{t_*-\tau}$ under $\mathbb{P}$ via S-Alg and $E \sim \mathrm{Exp}(1)$

11: **until** $E \geqslant q(w + v)$

12: **return** $(T + \tau, U + u, U + v)$

---

tively samples S_T (and stores the value in U) over the grid $T \in \{t_*, 2t_*, \ldots\}$ (i.e., the variable T keeps track of time) under the tempered stable law $\mathbb{P}_q$ until the interval containing τ_b is identified, i.e., the smallest $T \in t_*\mathbb{N}$ with $S_{T+t_*} \geqslant b(T + t_*)$. When the interval $(T, T + t_*]$ containing τ_b under $\mathbb{P}_q$ is identified, in lines 8–11, TSFP-Alg samples $(\tau_b - T, S_{\tau_b-} - S_T, S_{\tau_b} - S_T)$ under $\mathbb{P}_q$ via rejection sampling: it proposes $(\tau_b - T, S_{\tau_b-} - S_T, S_{\tau_b} - S_T)$ and $S_{T+t_*} - S_{\tau_b}$ under $\mathbb{P}$, and accepts the sample (under $\mathbb{P}_q$) with probability $\exp(-q(S_{T+t_*} - S_T))$. The parameter t_* affects the following three components of TSFP-Alg: the number of iterations of the loop in lines 2–7, the complexity of SFP-Alg called in line 9 (cf. Theorem 2.3 below) and the acceptance probability in line 11. The choice of t_* in line 1 of TSFP-Alg is made to control the overall complexity of the algorithm, see details in (4.28) of Section 4 below.

Stable First-Passage algorithm SFP-Alg and its analysis constitute the main technical contribution of this paper. The complexity of TSFP-Alg is given in the following theorem. Proofs of all the results in this paper are in Section 4 below.

Theorem 2.1. *TSFP-Alg samples from the law of the triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ of a tempered stable subordinator S under $\mathbb{P}_q$. Moreover, the running time of TSFP-Alg has exponential moments with mean bounded by*

$$\kappa_{TSFP}\mathrm{e}^{2qb(0)/(2^\alpha-1)}((1-\alpha)^{-3} + |\log\alpha| + \log N)/\alpha,$$

where the constant κ_{TSFP} does not depend on the parameters $\alpha \in (0,1)$, $\theta \in (0, \infty)$ or $q \in [0, \infty)$, and N specifies the required number of precision bits of the output.

The upper bound on the running time of TSFP-Alg has a term of order $\mathcal{O}(\exp(2qb(0)/(2^\alpha - 1)))$, which may be quite large if either $b(0)$, q or $1/\alpha$ are large. However, the exponential dependence on these parameters of the complexity of TSFP-Alg can be easily turned into a linear dependence (see Corollary 2.2) by picking a constant $R > 0$ and applying TSFP-Alg successively with the boundary function $t \mapsto \min\{b(t), R\}$ until the initial boundary b is crossed. Our main algorithm TSFFP-Alg, which we now describe, does precisely this. Indeed, variables T and U, introduced in line 1 of TSFFP-Alg keep track of time and the value S_T of S at this time. Line 3 of TSFFP-Alg, samples $(\tau_c - T, S_{\tau_c-} - S_T, S_{\tau_c} - S_T)$

under $\mathbb{P}_q$ via TSFP-Alg, where $c : t \mapsto \min\{b(t), S_T + R\}$. The algorithm updates T and $U = S_T$ in line 4. If $S_{\tau_c} \geqslant b(\tau_c)$ (line 5), then $\tau_b = \tau_c$, the algorithm stops and returns the triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ in line 6; otherwise (i.e., if $S_{\tau_c} < b(\tau_c)$), it repeats from line 2. The choice of R in TSFFP-Alg is fixed in line 1 and is chosen to minimise an upper bound on the expected running time of the algorithm, see details in Section 4 below.

Algorithm 2 (TSFFP-Alg) Tempered Stable Fast First-Passage Algorithm: samples $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}_q$

Require: Parameters $\alpha \in (0,1)$, $\theta > 0$, $q \geqslant 0$ and function $b : [0, \infty) \to [0, \infty)$

  1: Set $R \leftarrow (2^\alpha - 1)/(2q)$, $T \leftarrow 0$, $U \leftarrow 0$ and $c(\cdot) \leftarrow b(\cdot)$

  2: **repeat**

  3:     Sample (τ, u, v) via TSFP-Alg with function $\min\{c, R\}$

  4:     Update $T \leftarrow T + \tau$, $U \leftarrow U + v$, $c(\cdot) \leftarrow c(\cdot + \tau) - v$

  5: **until** $U \geqslant b(T)$

  6: **return** $(T, U - (v - u), U)$ ▷ The jump that takes S over b has size $v - u$

Corollary 2.2. *TSFFP-Alg samples from the law of $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}_q$. Moreover, the running time of TSFFP-Alg has exponential moments with mean bounded by*

$$\kappa_{TSFFP}(1 + qb(0)/\alpha)((1 - \alpha)^{-3} + |\log \alpha| + \log N)/\alpha,$$

where κ_{TSFFP} does not depend on $\alpha \in (0,1)$, $\theta \in (0, \infty)$ or on $q \in [0, \infty)$, and N is as in Theorem 2.1.

2.3 Sampling the first passage of a stable subordinator

The algorithm SFP-Alg for sampling $(\tau_b, S_{\tau_b-}, S_{\tau_b}) | \{\tau_b \leqslant t_*\}$ under the stable law $\mathbb{P}$ is based on the formulae in Proposition 4.1 below, describing the law of $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}$. Before presenting SFP-Alg, we give a brief account of its contents. First, SFP-Alg samples the crossing time τ_b of the stable subordinator, conditional on $\tau_b \leqslant t_*$, in line 1 and stores its value in the variable T. The algorithm then determines in lines 2 & 3 whether the crossing occurred via a jump $\Delta_S(\tau_b) := S_{\tau_b} - S_{\tau_b-}$ of positive size $\Delta_S(\tau_b) > 0$ or continuously (i.e., by creeping, with $\Delta_S(\tau_b) = 0$). In the latter case, the algorithm simply returns the crossing triple (line 4) and, in the former, the algorithm samples the undershoot and the overshoot and returns the triplet (lines 6–8).

Theorem 2.3. *SFP-Alg samples from the law of $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}$ conditional on $\tau_b \leqslant t_*$ (for $0 < t_* \leqslant \infty$). The running time of SFP-Alg has finite exponential moments and mean bounded by*

$$\kappa_{SFP}((1 - \alpha)^{-3} + |\log \alpha| + \log N)/\mathbb{P}[S_1 \geqslant B(t_*)],$$

where the constant κ_{SFP} does not depend on $\alpha \in (0,1)$, $\theta \in (0, \infty)$ or $t_* \in (0, \infty]$ and N specifies the number of precision bits of the output.

The proof of Theorem 2.3 is in Section 4.4 below. If $t_* = \infty$, then $B(\infty) = 0$, $\mathbb{P}[S_1 > 0] = 1$ and Theorem 2.3 implies that the expected running time of SFP-Alg may increase as fast as $(1 - \alpha)^{-3}$ as $\alpha \uparrow 1$. A deterioration in the limit as $\alpha \uparrow 1$ is expected since, in this limit, the stable subordinator S converges to

---

Algorithm 3 (SFP-Alg) Stable First-Passage Algorithm: samples $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ conditional on $\tau_b \leqslant t_*$ under $\mathbb{P}$

Require: Parameters $\alpha \in (0, 1)$, $\theta > 0$, $t_* \in (0, \infty]$, functions $b : [0, \infty) \to [0, \infty)$, $B : t \mapsto t^{-1/\alpha} b(t)$ and its inverse $B^{-1} : (0, \infty) \to (0, T_b)$ specified in Proposition 4.1 (with convention $B(\infty) = 0$)

1: Generate $V_1 \sim \mathcal{L}(S_1)$ via S-Alg until $V_1 \geqslant B(t_*)$ and set $T \leftarrow B^{-1}(V_1)$

2: Generate $U_1 \sim \mathrm{U}(0, 1)$

3: **if** $U_1 \leqslant -b'(T)/(-b'(T) + \alpha^{-1} T^{-1} b(T))$ **then**

4:     **return** $(T, b(T), b(T))$

5: **else**

6:     Generate $U_2 \sim \mathrm{U}(0, 1)$

7:     Generate $V_2 \sim \mathrm{SU}_\alpha(T, b(T))$ with law of $S_{\tau_b-}|\{\tau_b = T, b(\tau_b) = b(T), \Delta_S(\tau_b) > 0\}$ via SU-Alg

8:     **return** $(T, V_2, V_2 + (b(T) - V_2)U_2^{-1/\alpha})$

9: **end if**

---

a pure-drift subordinator θt, for which the distribution of the triplet degenerates to a delta distribution. Similarly, the expected running time of SFP-Alg may increase as fast as $|\log \alpha|$ as $\alpha \downarrow 0$. This is expected as well since the stable subordinator S converges to the trivial process 0 as $\alpha \downarrow 0$. Numerical evidence for these increases in complexity and the theoretical bounds in Theorem 2.3 can be found in Section 3.1 below. We stress that existing algorithms for the simulation of the triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ have infinite expected running time for any $\alpha \in (0, 1)$, see Subsection 2.5 below for more details.

As we said before, SFP-Alg has three main steps. First, the algorithm generates the first-passage time T in line 1. This requires evaluating, up to N bits of precision, the inverse B^{-1} of the function $B : t \mapsto t^{-1/\alpha} b(t)$, which is defined on $(0, T_b)$ where $T_b := \inf\{t > 0 : b(t) = 0\} \in (0, \infty]$. This step may require a numerical inversion of B (or, equivalently, numerically finding the root of $t \mapsto B(t) - V_1$). Such algorithms are standard in the literature (see Appendix C.3), making this a simple step. Second, the algorithm determines if the first passage occurred continuously (i.e., by creeping) in line 3. If the process crept, the triple equals $(T, b(T), b(T))$, but, if it did not, then we must sample the undershoot and overshoot of the process conditional on the crossing time. The third main step is precisely the simulation of the undershoot and overshoot in lines 7 and 8. By Proposition 4.1, the conditional law of the overshoot is a power law and hence easy to sample from, while the law of the undershoot, although expressible in terms of the stable law and Lévy measure, is complicated and hard to sample from. For this reason, the focus of the remainder of this section is to construct SU-Alg, which samples from the law of the undershoot.

2.4   Simulation of the stable undershoot via domination by a mixture of densities

Our main objective now is to present a simulation algorithm from the undershoot law $\mathrm{SU}_\alpha(t, w)$, $t, w > 0$, that is, the law of S_{τ_b-} under $\mathbb{P}$, conditional on $\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$. Our main tool is the general rejection-sampling principle. Unlike before, the tools and algorithms used here have little probabilistic interpretation and are mostly based on analytical properties and inequalities. The derivation of these inequalities, which validate SU-Alg, will be developed and explained in Section 4 below. For now, we only present the algorithms with a brief description of how it works. The algorithm has several nontrivial components that we will slowly introduce and explain. The reason for this is that the task of simulating

from the undershoot law $SU_\alpha(t, w)$ comes with multiple pitfalls, with several tempting approaches having severe problems, see Subsection 2.5 for an account of these pitfalls.

First, the density $f_{t,w}$ of $SU_\alpha(t, w)$ involves the density of the stable law, which can be written as a non-elementary integral of elementary functions (see Proposition 4.1). Thus, by extending the state space and applying an elementary transformation, the problem is converted into producing samples from a density ψ proportional to $(x, y) \mapsto (s - x^{-1/r})^{-\alpha} \sigma_\alpha(y) \mathrm{e}^{-\sigma_\alpha(y)x}$ on $(s^{-r}, \infty) \times (0, 1)$, where $r := \alpha/(1 - \alpha)$, $s := (\theta t)^{-1/\alpha} w$ and σ_α is the convex function defined in terms of trigonometric functions in (4.2). The density ψ_s is difficult to sample from directly, so we first bound it with (and propose from) the mixture $\widetilde{\psi}_s = p\widetilde{\psi}_s^{(1)} + (1 - p)\widetilde{\psi}_s^{(2)}$ of two simpler densities $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$ on $(s^{-r}, \infty) \times (0, 1)$, which are respectively proportional to $(x, y) \mapsto \sigma_\alpha(y) \mathrm{e}^{-\sigma_\alpha(y)x}$ and $(x, y) \mapsto \sigma_\alpha(y) \mathrm{e}^{-\sigma_\alpha(y)x}(x - s^{-r})^{-\alpha}$ (see details in Subsection 4.3 below).

Algorithm 4 (SU-Alg) Stable Undershoot Algorithm: samples $S_{t-}|\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$ under $\mathbb{P}$

Require: Parameters $\alpha \in (0, 1)$ and $\theta, t, w > 0$

  1: Set $s \leftarrow (\theta t)^{-1/\alpha} w$ and $r \leftarrow \alpha/(1 - \alpha)$, compute p' in (4.9) and set $p \leftarrow p'/(p' + 1)$

  2: **repeat**

  3:      Generate $V_1, V_2 \sim \mathrm{U}(0, 1)$

  4:      **if** $V_1 < p$ **then**

  5:          Generate (ζ_s, Y_s) with density $\widetilde{\psi}_s^{(1)}$ via $\psi^{(1)}$-Alg

  6:      **else**

  7:          Generate (ζ_s, Y_s) with density $\widetilde{\psi}_s^{(2)}$ via $\psi^{(2)}$-Alg

  8:      **end if**

  9: **until** $V_2 \leqslant (s - \zeta_s^{-1/r})^{-\alpha}/((1 - 2^{\alpha-1})^{-\alpha} s^{-\alpha} + (2r)^\alpha s^{-r}(\zeta_s - s^{-r})^{-\alpha})$

 10: **return** $(\theta t)^{1/\alpha} \zeta_s^{-1/r}$

SU-Alg proposes from the mixture $\widetilde{\psi}_s$ in lines 2–8 and accepts the sample with probability proportional to $\psi_s/\widetilde{\psi}_s$. Lines 3 & 4 of SU-Alg decide whether the sample from the mixture $\widetilde{\psi}_s$ will come from $\widetilde{\psi}_s^{(1)}$ (sampled in line 5) or $\widetilde{\psi}_s^{(2)}$ (sampled in line 7). The simulation algorithms corresponding to the simpler densities $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$ will be discussed later.

Remark 2.1. The proportion p' (and hence p) can be efficiently evaluated to arbitrary precision using Gauss–Legendre quadrature. In fact, the numerator in the formula in (4.9) equals the distribution function of a stable law while the integrand in the denominator is a convex geometric combination of the integrands of the density and distribution functions of a stable law. The computation time for p' is thus assumed to be constant for any $s = (\theta t)^{-1/\alpha} w$, and to only depend on the number of precision digits N, see fast algorithms in e.g. [23, 1]. The error of the quadrature is known to decay exponentially fast in the number of nodes but may also be sensitive to large values of the high order derivatives of the integrand [13]. In our case, the derivatives of the integrand also grow quickly in the order of differentiation, implying a locally large relative error in the numerical integral. Since, however, these derivatives are large precisely when the function f is nearly vanishing, the overall relative error appears to be small, resulting in a complexity that is only a function of the precision N, as has been confirmed many times in the literature and by our own simulation times, see Section 3.1 below.

Remark 2.2. Line 7 of SFP-Alg calls SU-Alg on random input, making the parameter $s = (\theta t)^{-1/\alpha} w$ in SU-Alg equal to $\theta^{-1/\alpha} V_1$ (where $V_1 \sim \mathcal{L}(S_1)$ is as in SFP-Alg), which follows the stable law of $S_{1/\theta}$ conditioned on $S_{1/\theta} \geqslant B(t_*)$. In particular, the density of s is proportional to the restriction to $[B(t_*), \infty)$ of φ_α, defined in (4.3), making s heavy tailed and its law dependent only on α and $B(t_*)$.

For any $a, x > 0$, define $\log_a^+(x) := \max\{0, \log(x)/\log(a)\}$.

Proposition 2.4. *SU-Alg samples from the law $\mathrm{SU}_\alpha(t, w)$ of the undershoot S_{t-} under $\mathbb{P}$, conditional on the event $\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$. The random running time of SU-Alg has exponential moments. Moreover, the expected running time of SU-Alg is bounded above by*

$$\kappa_{SU}\Big[(1-\alpha)^{-1}\Big(1 + \alpha(1-\alpha)^{-1}\log_2^+(s^{-1}) + \log_2^+\big(1/(s^{\alpha/(1-\alpha)} - \alpha^{(1-2\alpha)/(1-\alpha)}(1-\alpha))\big)$$
$$+ \alpha \log_2^+(s) + |\log \alpha| + (1-\alpha)^{-2}\Big) + \log N\Big],$$

where the constant $\kappa_{SU} > 0$ does not depend on $s = (\theta t)^{-1/\alpha} w \in (0, \infty)$ (see line 1 of SU-Alg), $t > 0$, $\theta \in (0, \infty)$ or $\alpha \in (0, 1)$.

The proof of Proposition 2.4 is given in Subsection 4.3.4 below. In the following two subsections we describe how to sample from the laws given by the densities $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$.

2.4.1 Simulation from the density $\widetilde{\psi}_s^{(1)}$.

The density $\widetilde{\psi}_s^{(1)}(x, y)$ is proportional to

$$(x, y) \mapsto \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}\mathbb{1}_{(s^{-r}, \infty)}(x)\mathbb{1}_{(0,1)}(y) = \mathrm{e}^{-\sigma_\alpha(y)s^{-r}}\mathbb{1}_{(0,1)}(y) \times \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)(x-s^{-r})}\mathbb{1}_{(s^{-r}, \infty)}(x).$$

The first factor (the marginal of the second component) is proportional to a log-concave density and can be sampled easily via LC-Alg by virtue of being log-concave (by Lemma 4.3 established below). The second factor (the conditional density of the first component given the second) is a shifted exponential with shift s^{-r} and scale $1/\sigma_\alpha(y)$, which is also easy to sample.

Algorithm 5 ($\psi^{(1)}$-Alg) Simulation from the density $\widetilde{\psi}_s^{(1)}$

Require: Parameters $\alpha \in (0, 1)$ and $s > 0$

1: Sample Y_s with density $y \mapsto \exp(-\sigma_\alpha(y)s^{-r})$ via LC-Alg
2: Sample $E \sim \mathrm{Exp}(1)$ and **return** $(s^{-r} + E/\sigma_\alpha(Y_s), Y_s)$

Proposition 2.5. *$\psi^{(1)}$-Alg samples from the density $\widetilde{\psi}_s^{(1)}$. Moreover, the running time of $\psi^{(1)}$-Alg is bounded by a multiple of $G + \alpha(1-\alpha)^{-1}\log^+(s^{-1})$ where G is a geometric random variable with parameter $1/5$. In particular, the running time has exponential moments with mean bounded by*

$$\kappa_{(1)}\big(1 + \alpha(1-\alpha)^{-1}\log^+(s^{-1})\big),$$

where the constant $\kappa_{(1)} > 0$ depends neither on $s \in (0, \infty)$ nor $\alpha \in (0, 1)$.

The proof of Proposition 2.5 is given in Subsection 4.3.1 below. $\psi^{(1)}$-Alg does not rely on numerical inversion (cf. NR-Alg). Its expected running time does therefore not depend on the number of precision bits N. In fact, the computational complexity of $\psi^{(1)}$-Alg primarily comes from the pre-processing step (line 1 of LC-Alg), which depends on the values of the parameters in $\psi^{(1)}$-Alg.

2.4.2 Simulation from the density $\widetilde{\psi}_s^{(2)}$.

Simulating from the density $\widetilde{\psi}_s^{(2)}$ is a delicate matter, particularly because of the sensitivity of the function on $\alpha \in (0,1)$ and on s, which is heavy-tailed in line 7 of SFP-Alg (see Remark 2.2 above). $\psi^{(2)}$-Alg below samples from the density $\widetilde{\psi}_s^{(2)}(x,y)$, proportional to $\sigma_\alpha(y)e^{-\sigma_\alpha(y)x}(x-s^{-r})^{-\alpha}$, in two steps. The algorithm first samples, in lines 1–25, the second marginal of $\widetilde{\psi}_s^{(2)}$, with density proportional to $y \mapsto \sigma_\alpha(y)^\alpha e^{-\sigma_\alpha(y)s^{-r}}$ on $(0,1)$. Then, the algorithm samples, in line 26, the first marginal conditionally given the second, with conditional density proportional to $x \mapsto (x-s^r)^{-\alpha}e^{-\sigma_\alpha(y)(x-s^r)}$ on (s^{-r}, ∞), which is simply a translated (by s^{-r}) gamma random variable with shape α and scale $1/\sigma_\alpha(y)$ and hence easy to sample from. The simulation from the second marginal density is done by breaking up the support depending on the parameters α and s and using rejection sampling with respect to various dominating (proposal) densities on each subinterval. Finding appropriate dominating functions, from which we can sample, is hard since the function $y \mapsto \sigma_\alpha(y)^\alpha e^{-\sigma_\alpha(y)s^{-r}}$ may have a very large peak at level $e^{-\alpha}\alpha^\alpha s^{\alpha^2/(1-\alpha)}$ followed by a extremely steep descent to 0, even for reasonable values of α, if the parameter s (recall s is heavy-tailed, see Remark 2.2 above) takes moderately large values. Figure 2.2 below illustrates how peaked the corresponding density may be in such cases.

$\psi^{(2)}$-Alg needs to numerically invert several twice differentiable functions $f : [0,1] \to \mathbb{R}$ to produce samples of certain dominating (proposal) densities. This inversion is achieved via the Newton–Raphson root-finding method given in NR-Alg. Each call of NR-Alg requires the input of a specific target function f and an auxiliary function $M : [0,1] \times \mathbb{N} \to \mathbb{R}$ satisfying (C.1) (see Appendix C.3).

Remark 2.3. For the same reason we mentioned in Remark 2.1, the integral $\int_a^b \tilde{f}(y)\mathrm{d}u$ can be computed efficiently to arbitrary precision via Gaussian–Legendre quadrature. The computation time for Step 3 in SU-Alg is thus assumed to be constant.

Proposition 2.6. *$\psi^{(2)}$-Alg samples from the law given by the density $\widetilde{\psi}_s^{(2)}$. Moreover, the running time of $\psi^{(2)}$-Alg is bounded above by a constant multiple of*

$$G + \log^+\left(1/\left(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}\right)\right) + \alpha\log^+(s) + |\log\alpha| + \log N,$$

where N specifies the number of precision bits of the output and G is a geometric random variable with probability of success $\mathbb{P}(G=1)$ bounded below by a multiple of $(1-\alpha)^2$. In particular, the running time of $\psi^{(2)}$-Alg has exponential moments and its mean is bounded by

$$\kappa_{(2)}\left[(1-\alpha)^{-2} + \log^+\left(1/\left(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}\right)\right) + \alpha\log^+(s) + |\log\alpha| + \log N\right],$$

where the constant $\kappa_{(2)}$ depends neither on $s \in (0,\infty)$ nor $\alpha \in (0,1)$.

The proof of Proposition 2.6 is given in Subsection 4.3.3 below. The computational cost in $\psi^{(2)}$-Alg comes from various sources: (I) applications of the inversion NR-Alg, which may depend on the parameters $\alpha \in (0,1)$ and $s > 0$ (recall that the parameter s is in fact random in SU-Alg, which calls $\psi^{(2)}$-Alg); (II) in the cases $D = 0$ and $D = 1$, we also perform an accept-reject step where the acceptence probability depends on the parameters $\alpha \in (0,1)$ and $s > 0$; (III) in the case $D = 2$, the pre-processing (line 1 in LC-Alg) also adds to the final computational complexity since the function LC-Alg is applied to in line 5 of $\psi^{(2)}$-Alg depends both on $\alpha \in (0,1)$ and $s > 0$. We note that the accept-reject step in the case $D = 2$ is embedded in LC-Alg and has uniformly bounded (in parameter values) expected running time.

Algorithm 6 ($\psi^{(2)}$-Alg) Simulation from the density $\widetilde{\psi}_s^{(2)}$

Require: Parameters $\alpha \in (0,1)$ and $s > 0$

1: Set $z \leftarrow 0$ if $\alpha s^r < \sigma_\alpha(0+)$. Else $z \leftarrow \sigma_\alpha^{-1}(\alpha s^r)$ via NR-Alg with $f = \log \sigma_\alpha$ and M in (4.14)

2: Define the function $\tilde{f}(y) = \sigma_\alpha(y)^\alpha \exp(-\sigma_\alpha(y)s^{-r})$ and set $z_* \leftarrow \min\{z, 1/2\}$

3: Generate discrete variable D on $\{0,1,2\}$ with weights $(\int_0^{z_*} \tilde{f}(y)\mathrm{d}y, \int_{z_*}^z \tilde{f}(y)\mathrm{d}y, \int_z^1 \tilde{f}(y)\mathrm{d}y)$

4: **if** $D = 2$ **then**

5:     Generate Y_s from the density proportional to $\tilde{f}|_{[z,1]}$ via LC-Alg

6: **else if** $D = 0$ **then**

7:     **repeat**

8:         Generate Y_s with distribution f in (4.19) via NR-Alg with M in (4.15)

9:         Generate $V \sim \mathrm{U}(0,1)$

10:     **until** $V \leqslant \exp(-\sigma_\alpha(Y_s)s^{-r})/(1 + \alpha Y_s \sigma_\alpha'(Y_s)/\sigma_\alpha(Y_s))$

11: **else** $\triangleright$ (here, $D = 1$)

12:     **if** $\alpha \leqslant 1/2$ **then**

13:         Set $a \leftarrow \sin(\pi(1-\alpha))$ and $b \leftarrow \pi\alpha(1-\alpha)\cos(\pi\alpha)$

14:         **repeat**

15:             Generate Y_s with distribution f defined in (4.21) via NR-Alg with M in (4.17)

16:             Generate $V \sim \mathrm{U}(0,1)$

17:         **until** $V \leqslant \sigma_\alpha(Y_s)^\alpha e^{-\sigma_\alpha(Y_s)s^{-r}} a^{1-\alpha} 2^r (1-Y_s)^r/(a + b(1-Y_s))$

18:     **else**

19:         Set $c \leftarrow \rho'(1/2)/\rho(1/2)^2$, $a \leftarrow \rho(1/2)^{r-1}$ and $b \leftarrow \rho(z)^{r-1} - a$

20:         **repeat**

21:             Generate Y_s with distribution f defined in (4.24) via NR-Alg with M in (4.14)

22:             Generate $V \sim \mathrm{U}(0,1)$

23:         **until** $V \leqslant ce^{-\sigma_\alpha(Y_s)s^{-r}}\rho(Y_s)^2/\rho'(Y_s)$

24:     **end if**

25: **end if**

26: Generate $E \sim \mathrm{Gamma}(1-\alpha, 1)$, set $\zeta_s \leftarrow s^{-r} + E/\sigma_\alpha(Y_s)$ and **return** (ζ_s, Y_s)
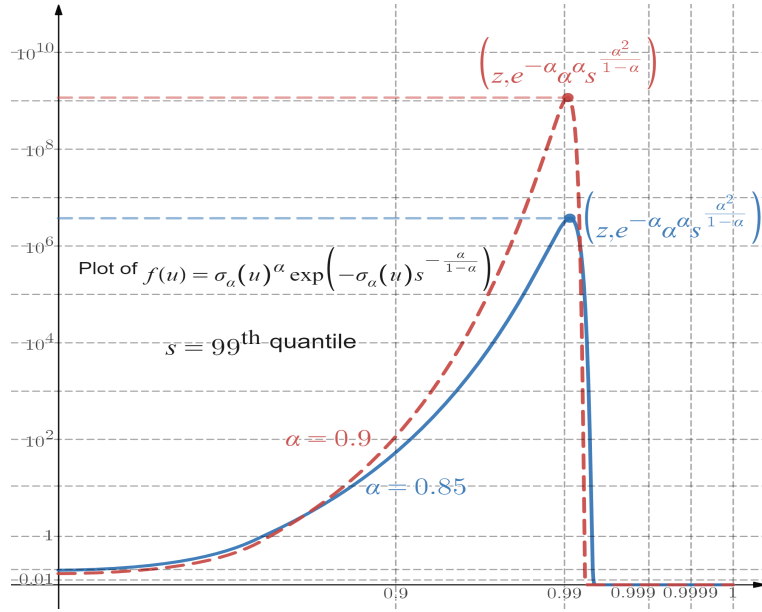

Figure 2.2: Density of the second marginal of $\widetilde{\psi}_s^{(2)}$. The figure shows the graphs of the density of the second marginal of $\widetilde{\psi}_s^{(2)}$, in a custom logarithmic-style scale, for two processes with $\alpha = 0.85$ and $\alpha = 0.9$, where the parameter s is at its 99th quantile. Thus, on average in one of every 100 runs of SFP-Alg, $\psi^{(2)}$-Alg is called to sample from a density which, at the mode $z = \sigma_\alpha^{-1}(\alpha s^r)$, takes extremely large value followed by a steep drop towards zero. Moreover, in this case the density is strictly increasing on the interval $(0, z)$ and strictly positive on $(0, 1)$. But since its derivatives of all orders are equal to zero at 1, the graphs appear to be equal to zero on most of the interval $(z, 1)$. Furthermore, the behaviour of the density on either side of the mode z is markedly different. Taking a uniform upper bound $\mathrm{e}^{-\alpha}\alpha^\alpha s^{\alpha^2/(1-\alpha)}$ would result in an extremely inefficient algorithm with infinite expected running time (in fact, such a bound was used to sample from this type of function in [8], see Subsection 2.7 below). For parameter values as in the graph above, $\psi^{(2)}$-Alg sets $z_* = 1/2$, splits the interval $(0, 1) = (0, z_*] \cup (z_*, z] \cup (z, 1)$ and uses appropriate proposal densities on each subinterval.

2.5 How *not* to sample the stable undershoot $S_{\tau_b-}|\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$

At a glance, there are several apparently feasible ways to produce a sample from the undershoot law $\mathrm{US}(t, w)$ of S_{τ_b-}, under $\mathbb{P}$, conditional on $\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$. However, most of these simulation methods are either infeasible altogether or their computational complexity grows rapidly as either $\alpha \downarrow 0$ or $\alpha \uparrow 1$. In this section, we will briefly discuss some of these "approaches".

First recall that sampling from the undershoot law is, by Proposition 4.1(c), essentially reduced to producing samples from the density proportional to $x \mapsto \varphi_\alpha(x)(s - x)^{-\alpha}\mathbb{1}_{\{x < s\}}$ where $s = (\theta t)^{-1/\alpha}w = (\theta\tau_b)^{-1/\alpha}b(\tau_b) > 0$ and φ_α is the density of $S_{1/\theta}$ and given in (4.3). A first guess would be to employ rejection sampling on this density. Indeed, since the function $x \mapsto (s - x)^{-\alpha}$ is unbounded, we cannot propose from the density φ_α. However, the density φ_α *is* bounded and hence we may propose from the density proportional to $x \mapsto (s - x)^{-\alpha}$ and accept the sample with probability proportional to $\varphi_\alpha(x)$. For a fixed s, this algorithm has a running time with finite mean; however, we require an algorithm that performs well for $s = (\theta\tau_b)^{-1/\alpha}b(\tau_b) \stackrel{d}{=} S_{1/\theta}$. Since $x \mapsto (s - a)^{-\alpha}$ puts most of its mass close to s (which is stable distributed and thus heavy tailed), the acceptance probability of such an algorithm is asymptotically

14

equivalent to $\varphi_\alpha(s)/\sup_{u\in(0,\infty)}\varphi_\alpha(u) \sim Cs^{-\alpha-1}$ as $s \to \infty$ for some $C > 0$. In turn, this means that the running time is bounded below by $C' \max\{1, S_1^{\alpha+1}\}$, for some $C' > 0$, which does not have a finite mean (in fact, it has a finite moment $p > 0$ if and only if $p < \alpha/(\alpha+1)$).

Another possibility would be to perform a numerical inversion of the distribution function F_α proportional to $x \mapsto \int_0^x \varphi_\alpha(y)(s-y)^{-\alpha}\mathrm{d}y$. Since

$$\frac{F_\alpha'(x)}{F_\alpha(x)} = \frac{\varphi_\alpha(x)(s-x)^{-\alpha}}{\int_0^x \varphi_\alpha(y)(s-y)^{-\alpha}\mathrm{d}y},$$

where φ_α can be computed to arbitrary precision using numerical quadrature, one may expect Newton–Raphson's method to perform well in inverting F_α numerically. It is, however, computationally intensive. Every step in Newton–Raphson method and in the binary search required prior to the quadratic convergence of Newton–Raphson's method evaluates the double integral in F_α (recall that φ_α is itself given in terms of an integral). If the numerical quadrature for computing φ_α accurately requires n nodes, the corresponding quadrature for F_α needs approximately n^2 nodes. Since we typically have $n \in [10^4, 10^5]$ for double-precision floating-point numbers, each numerical evaluation of F_α requires evaluating at least 10^8 functions (sums, products, exponentiation and trigonometric functions). This makes this approach infeasible despite the good properties of Newton–Raphson method and the boundedness and smoothness of the function φ_α.

A third option is to maintain parts of our current methodology and modify the parts of the algorithm that sample from a density of the form $u \mapsto \sigma_\alpha(u)^p \exp(-\sigma_\alpha(u)s)$ for some $p \in (0, \infty) \setminus \{1\}$. For instance, one may attempt a rejection-sampling algorithm via elementary bounds on σ_α, such as those found in Lemma 4.9 below. Such algorithms are feasible and, in fact, we use some of them for certain parameter combinations. The drawback of those approaches is that the acceptance probability becomes tiny as $\alpha \to 1$, which is why we introduce the auxiliary parameter z_* and other simulation algorithms for certain parameter combinations. Indeed, the acceptance probability is often upper bounded by c^r for some $c \in (0, 1)$ where we recall that $r = \alpha/(1-\alpha)$. This probability is incredibly small even for moderate values of α. For instance, the quotient between the lower and upper bounds on σ_α in Lemma 4.9 is proportional to $(\pi/4)^{r+1}(1-\alpha)^r$, as $\alpha \to 1$, which is approximately 4.28×10^{-6} (resp. 8.93×10^{-11}) when $\alpha = 0.85$ (resp. $\alpha = 0.9$).

Our SU-Alg (and its subalgorithms) are in fact informed by the pitfalls described above. In particular, the choices made in $\psi^{(2)}$-Alg may appear arbitrary but are, in fact, designed to control the computational complexity. It remains an open problem to design an exact simulation algorithm with uniformly bounded expected running time on the entire parameter space $\alpha \in (0, 1)$. Figure 3.1b below appears to suggest that the complexity of our algorithm grows as $(1-\alpha)^{-0.2}$ as $\alpha \uparrow 1$, which is much smaller than $(1-\alpha)^{-3}$, as suggested by our theoretical bounds.

2.6 Simulation of $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$ via numerical inversion with numerical integrals

It is natural to enquire[2] about the performance of a simulation method based on a direct numerical inversion of the distribution functions corresponding to the marginals $y \mapsto \int \widetilde{\psi}_s^{(1)}(x, y)\mathrm{d}x$ and $y \mapsto \int \widetilde{\psi}_s^{(2)}(x, y)\mathrm{d}x$. Using direct numerical inversion via Householder's method with $k = 4$ (see H-Alg in Appendix C.3 below), we can sample from these marginals of $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$.

Recall that $\psi^{(1)}$-Alg requires no numerical integration, while $\psi^{(2)}$-Alg uses 3 such integrations (see line 3), each over an interval where the integrand is monotone. In contrast, since every step of H-Alg

---

[2]We thank an anonymous referee for drawing our attention to this questions.

evaluates the distribution function, the numerical integration is used much more extensively over regions where the integrands are not monotone. Thus the direct inversion method is more sensitive to large values of the number N of precision digits and to instabilities in the numerical integration appearing for such large N.

We compared numerically the outputs of $\psi^{(1)}$-Alg and $\psi^{(2)}$-Alg with the outputs of their direct inversion counterparts: Figure 2.3 shows that our algorithms and the direct inversion alternatives sample from the same laws. We found that algorithms $\psi^{(1)}$-Alg and $\psi^{(2)}$-Alg were on average approximately 32.01 and 14.82 times faster, respectively than the direct inversion alternatives over the range of $\alpha \in \{.1, .3, .5, .7, .9\}$.
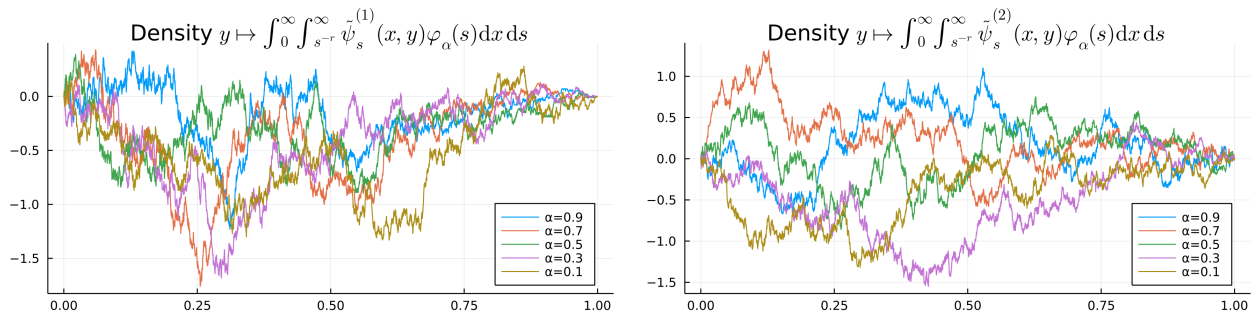
Figure 2.3: Numerical inversion with numerical integrals. The graph on the left (resp. right) show the function $y \mapsto \sqrt{n}(F_n(y) - G_n(y))$ where F_n and G_n are the empirical distribution functions of the samples produced using $\psi^{(1)}$-Alg (resp. $\psi^{(2)}$-Alg) and direct numerical inversion, respectively, for $n = 10^4$. Such functions indeed resemble Brownian bridges time-changed by the corresponding distribution functions. All samples pass the two-sample Kolmogorov-Smirnov test, confirming that our algorithms and direct numerical inversion produce samples from the same laws.

2.7 Infinite expected running time of the algorithm in [8, §7]

In [8, §7], an algorithm that produces a sample of the first-passage triplet of a tempered stable process is given. Such an algorithm has almost surely finite running time. However, it can be seen as follows, that its expected running time is infinite.

Suppose for simplicity that $\theta = 1$ and b is bounded by the variable denoted by r in the algorithm in [8, §7]. Consider Step 3 of the algorithm in [8, §7], which uses rejection sampling to simulate the pair $(S_{\tau_b-}, \Delta_S(\tau_b))$ given $\{\tau_b = t\}$ (recall from Proposition 4.1 that $\tau_b \overset{d}{=} B^{-1}(S_1)$, where B^{-1} is the inverse of $t \mapsto t^{-1/\alpha}b(t)$). The proposal for $s = S_{\tau-}$ is $s = \beta b(t)$ for some $\beta \sim \text{Beta}(1, 1-\alpha)$, with acceptance probability $p = h(t^{-1/\alpha}s, U)/M_\alpha$, where $U \sim \text{U}(0,1)$ is an auxiliary variable, $h(x, u) := \sigma_\alpha(u)x^{-r-1}e^{-\sigma_\alpha(u)x^{-r}}$ and $M_\alpha := (1-\alpha)^{1-1/\alpha}\alpha^{-1-1/\alpha}e^{-1/\alpha}$ is a global bound on h. Thus, the expected runnig time of this step equals $\mathbb{E}[C]$, where $C := 1/p \overset{d}{=} M_\alpha/h(\beta S_1, U)$ and the variables U, β and S_1 are independent. Hence, recalling that $r = \alpha/(1-\alpha)$, we have

$$\begin{aligned} \mathbb{E}[C] &= M_\alpha \mathbb{E}[\sigma_\alpha(U)^{-1}(S_1\beta)^{r+1} \exp(\sigma_\alpha(U)(S_1\beta_1)^{-r})] \geqslant M_\alpha \mathbb{E}[\sigma_\alpha(U)^{-1}(S_1\beta)^{r+1}] \\ &= M_\alpha \mathbb{E}[\sigma_\alpha(U)^{-1}]\mathbb{E}[S_1^{r+1}]\mathbb{E}[\beta^{r+1}] \end{aligned} \tag{2.2}$$

However, $\mathbb{E}[S_1^\eta] < \infty$ if and only if $\eta < \alpha$ by (4.26) below. Since $r = \alpha/(1-\alpha) > 0$ for all $\alpha \in (0,1)$, the expected running time of the algorithm in [8, §7] is indeed infinite.

16

We note that, similarly, Step 5 of the algorithm in [8, §7] also has infinite expected running time. Since Step 5 is more complex than Step 3, we made the phenomenon above explicit only for Step 3.

2.8 Can an implemented simulation algorithm be exact?

The answer depends on the definition of exact simulation. Since our algorithms are implemented inside computers with finite resources, the output of any algorithm cannot have the same law as the variable it is simulating if the variable has a density. However, we may consider an algorithm to be *practically exact* if a chosen distance between the law of the ideal variable I and the algorithm's output A can be controlled within a given multiple of *machine precision* (see Appendix D for further discussion of this question and various natural suggestions for the distance). Our algorithms are practically exact with respect to the Prokhorov metric (which metrises weak convergence) because we can create a coupling under which, with high probability, the distance $|I - A|$ is bounded by a constant multiple of machine precision, and this multiplicative constant is only dependent on the parameters $(\alpha, \theta q)$ and $b(0)$, see details in Appendix D below.

3 Applications

The main objective of this section is to present some applications of our algorithms. First, we present some numerical evidence for the theoretical bounds on the expected running time of TSFFP-Alg and SFP-Alg. We then show how TSFFP-Alg can be used to sample the first-passage event of a two-sided tempered stable process, which in turn can be used to price barrier options via Monte Carlo. Next, we use the probabilistic representation of the solution of a fractional partial differential equation (FPDE) [18] along with TSFFP-Alg to produce Monte Carlo estimators of the solution of such a FPDE.

3.1 Implementation

In this section, we showcase an implementation of our algorithms in Julia computing language. Our first goal is to show that such an implementation makes the simulation of the first-passage event feasible and fast in practice. Our second goal is to show that, in practice, our bounds on the expected running time of TSFFP-Alg and SFP-Alg overestimate the true complexity.

Observe that the expected running time in Figure 3.1 appears to be smaller for $\alpha = 0.999$ (close to 1) than for $\alpha = 0.001$ (close to 0). Although this is surprising since our upper bounds are of order $\mathcal{O}((1-\alpha)^{-3})$ for large α and of order $\mathcal{O}(|\log \alpha|)$ for small α, it is important to consider a few things. First, our key $\psi^{(2)}$-Alg, used to simulate the stable undershoot, uses a very different simulation method for $\alpha \leqslant 1/2$ and for $\alpha > 1/2$. In particular, the multiplicative constants in the $\mathcal{O}$ notation can be significantly different. Second, the bounds in Theorem 2.3 are only bounds, and need not be sharp. In particular, this does not necessarily mean that the asymptotic cost as $\alpha \uparrow 1$ is truly smaller than the asymptotic cost as $\alpha \downarrow 0$.

The behaviour of the expected running time of TSFFP-Alg, as a function of $q > 0$, is in good agreement with our upper bound in Theorem 2.1, see Figure 3.2 for details.

Mean running time $\mathbb{E}[T]$ (seconds per 10^4 samples)

Mean running time $\mathbb{E}[T]$ (seconds per 10^4 samples)
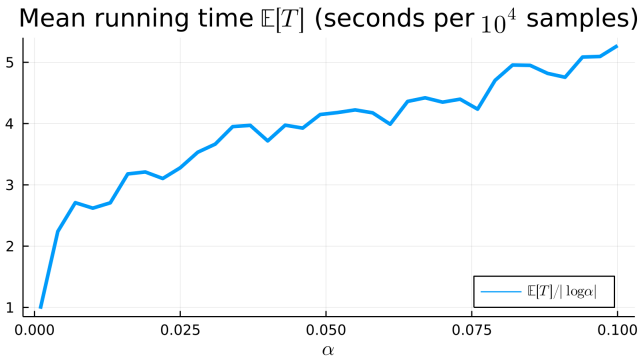
(a) By Theorem 2.3, $\mathbb{E}[T] = \mathcal{O}(|\log \alpha|)$ as $\alpha \downarrow 0$. The picture confirms this: $\mathbb{E}[T]/|\log \alpha|$ decreases to a number close to 1 as $\alpha \downarrow 0$ (the reported value of $\mathbb{E}[T]$ for $\alpha = 0.001$ is 9.30 seconds per 10^4 samples).
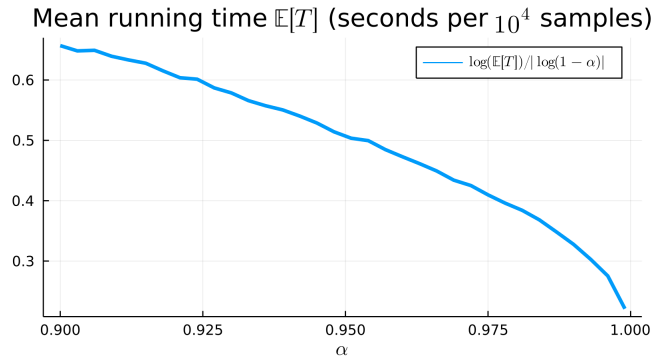
(b) By Theorem 2.3, $\mathbb{E}[T] = \mathcal{O}((1 - \alpha)^{-3})$ as $\alpha \uparrow 1$. The picture confirms this: $\log(\mathbb{E}[T])/|\log(1 - \alpha)|$ decreases as $\alpha \uparrow 1$ to a number close to 0.2 (the reported value of $\mathbb{E}[T]$ for $\alpha = 0.999$ is 6.28 seconds per 10^4 samples).

Figure 3.1: Implementation of SFP-Alg with $t_* = \infty$ on Julia. The average time $\mathbb{E}[T]$ is measured in seconds taken for every 10^4 samples.

3.2 Extensions to Lévy processes of bounded variation

We start by recalling the algorithm in [8, §5.1]. Let Z be a Lévy process with non-positive drift, such that its Lévy measure satisfies

$$\nu(\mathbb{R}_+) = \infty \quad \text{and} \quad \int_{\mathbb{R}} (|x| \wedge 1)\nu(\mathrm{d}x) < \infty.$$

Decompose Z as $Z^+ - Z^-$, where Z^+ and Z^- are independent subordinators with Levy measures ν^+ and ν^-, respectively, and where Z^+ is driftless. Fix some $c > 0$ and let $\tau_c^Z := \inf\{t > 0 : Z_t > c\}$. The following algorithm provides a method to simulate τ_c^Z under the assumption that the first-passage triplets $(\tau_x^+, Z_{\tau_x^+ -}, Z_{\tau_x^+})$, where $\tau_x^+ = \inf\{t > 0 : Z_t^+ > x\}$, as well as the increments Z_t^+ and Z_t^- can be sampled for any $x > 0$ and $t > 0$. Further consider some $T \in (0, \infty]$. According to [8, Proposition 5.1], if either $\limsup_{t \to \infty} Z_t = \infty$ a.s. (which can be determined, e.g., via Rogozin's criterion [17, Thm 2.7]) or $T < \infty$, then BVFP-Alg below stops a.s. and it samples the triplet $(\min\{\tau_c^Z, T\}, Z_{\min\{\tau_c^Z, T\}-}, Z_{\min\{\tau_c^Z, T\}})$.

3.2.1 Pricing barrier options.

Let Z be a Lévy process of bounded variation started from 0 and let $R_t := R_0 e^{Z_t}$, $t \geqslant 0$, model the risky asset for some initial value $R_0 > 0$. The simulation of the first-passage event can be used to obtain Monte Carlo estimators of the price of a barrier option. Indeed, fix $K < M$ and consider the payoff of the up-and-out barrier call option with payoff

$$g(T, R) = \max\{R_T - K, 0\} \mathbb{1}\left\{\sup_{t \in [0,T]} R_t < M\right\} = \max\{R_0 e^{Z_T} - K, 0\} \mathbb{1}\left\{\sup_{t \in [0,T]} Z_t < \log(M/R_0)\right\}$$

$$= \max\{R_0 e^{Z_T} - K, 0\} \mathbb{1}\{\tau_{\log(M/R_0)} > T\} = \max\{R_0 e^{Z_{\tau_*}} - K, 0\} \mathbb{1}\{\tau_* = T\},$$

where $\tau_* := \min\{\tau_{\log(M/R_0)}^Z, T\}$ and $\tau_x^Z = \inf\{t > 0 : Z_t > x\}$. Then the price of the option is given by $G(T, R_0) := e^{-\delta T} \mathbb{E}[g(T, R)]$, where $\delta \in \mathbb{R}$ is the discount rate.

Let $(\tau_*^i, Z_{\tau_*^i}^i)$, $i \in \mathbb{N}$, be iid samples with the law of (τ_*, Z_{τ_*}) generated via BVFP-Alg. Then, for every

18

Mean running time $\mathbb{E}[T]$ (seconds per 10^2 samples)
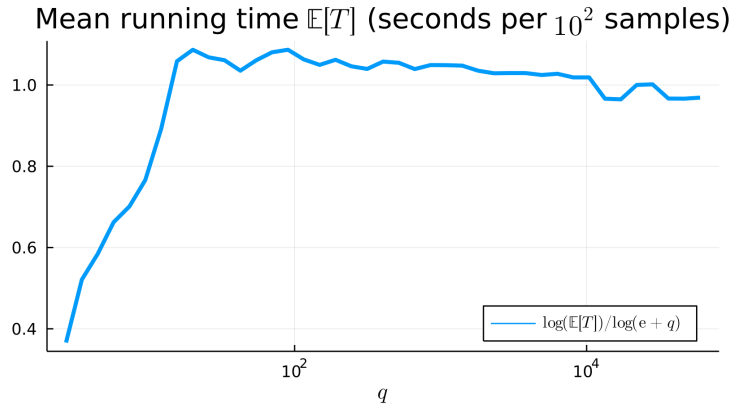
Figure 3.2: Implementation of TSFFP-Alg. The picture depicts an implementation of TSFFP-Alg with parameters $\alpha = 0.55$, $\theta = 1$, $b(t) \equiv 1$ and $q \in \{e^0, e^{1/4}, \dots, e^{11}\}$. The mean running time $\mathbb{E}[T]$ is measured in seconds taken for every 10^2 samples. By Corollary 2.2, $\mathbb{E}[T] = \mathcal{O}(q)$ as $q \to \infty$. This is supported by the picture: the quotient $\log(\mathbb{E}[T])/\log(e + q)$ appears to converge to 1 as $q \to \infty$.

n, we consider the Monte Carlo estimator

$$G_n(T, R_0) := \frac{1}{n} \sum_{k=1}^{n} \max\{R_0 e^{Z^i_{\tau_*}} - K, 0\} \mathbb{1}\{\tau^i_* = T\},$$

of $G(T, R_0)$. Figure 3.3 shows such an implementation for the case where Z is the difference of two tempered stable subordinators. To obtain smoother graphs, we apply BVFP-Alg to successive values of the barrier level $\log(M/R_0)$ (i.e., by decreasing values of $R_0 \in \{98, 98.031, \dots, 101.999\}$), obtaining strongly correlated samples. This is a well-known procedure in Monte Carlo estimation that retains the accuracy of each estimate as if we had used independent samples for each value of R_0, but reduces the random fluctuations between any two estimated values, see [14, §4.2].
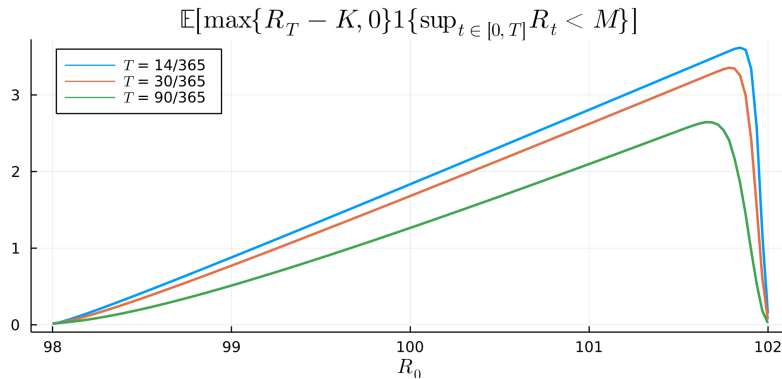
$\mathbb{E}[\max\{R_T - K, 0\} \mathbb{1}\{\sup_{t \in [0, T]} R_t < M\}]$

Figure 3.3: Estimated values of $e^{-\delta T} \mathbb{E}[\max\{R_T - K, 0\} \mathbb{1}\{\sup_{t \in [0,T]} R_t < M\}]$ using $n = 10^4$ samples. We considered the time horizons $T \in \{14/365, 30/365, 90/365\}$ and parameters $\delta = 0$, $K = 98$ and $M = 102$. We assume $Z = Z^+ - Z^-$ where Z^+ and Z^- are both driftless tempered stable subordinators with parameters $(\alpha^+, \theta^+, q^+) = (0.66, 0.1305, 6.5022)$ and $(\alpha^-, \theta^-, q^-) = (0.66, 0.0615, 3.3088)$, respectively, as calibrated from the USD/JPY currency pair, see [2, Table 3].

19

Algorithm 7 (BVFP-Alg) Bounded Variation First-Passage Algorithm: for a bounded variation Lévy process Z, samples $(\min\{\tau_c^Z, T\}, Z_{\min\{\tau_c^Z, T\}-}, Z_{\min\{\tau_c^Z, T\}})$

Require: Process $Z = Z^+ - Z^-$ with non-positive drift, barrier level $c \in (0, \infty)$, time horizon $T \in (0, \infty]$, such that either $\limsup_{t \to \infty} Z_t = \infty$ a.s. or $T < \infty$.

1: Set $t \leftarrow 0$, $h \leftarrow 0$, $v \leftarrow 0$ and $b \leftarrow c$
2: **repeat**
3:      Sample $(s, u, v) \sim \mathcal{L}(\tau_b^+, Z_{\tau_b^+-}^+, Z_{\tau_b^+}^+)$ and $w \sim \mathcal{L}(Z_s^-)$
4:      **if** $s + t \geqslant T$ **then**
5:          Sample $u \sim \mathcal{L}(Z_{T-t}^+)$ until $u < b$ and sample $w \sim \mathcal{L}(Z_{T-t}^-)$
6:          **return** $(T, h + u - w, h + u - w)$
7:      **else**
8:          Set $t \leftarrow t + s$, $h \leftarrow h + v - w$, $b \leftarrow b - v + w$
9:      **end if**
10: **until** $b < 0$
11: **return** $(t, h + u - v, h)$

3.3 Solution of fractional partial differential equations

Let $g : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}$ be a continuous function that vanishes at infinity and $\phi : \mathbb{R}^d \to \mathbb{R}$. For $a < b$, consider the following fractional partial differential equation (FPDE):

$$(-_t D_a + A_x)u(t, x) = -g(t, x) \quad (t, x) \in (a, b] \times \mathbb{R}^d,$$
$$u(a, x) = \phi(x), \quad x \in \mathbb{R}^d, \tag{3.1}$$

where A_x is a generator of a Feller semigroup on $C_\infty(\mathbb{R}^d)$ acting on x, $\phi \in \text{Dom}(A_x)$, the operator $-_t D_a$ is a generalised differential operator of Caputo type of order less than 1 acting on the time variable $t \in [a, b]$. The solution u of the FPDE problem in (3.1) exists and its stochastic representation is given by (see [18]):

$$u(t, x) = \mathbb{E}\left[\phi(X_{T_t}^x) + \int_0^{T_t} g(Y_s^{a,t}, X_s^x) \mathrm{d}s\right]$$

where $\{X_s^x\}_{s \geqslant 0}$ is the stochastic process generated by A_x and $T_t = \inf\{s > 0, Y_s^{a,t} < a\}$ where $\{Y_s^{a,t}\}_{s \geqslant 0}$ is the decreasing $[a, b]$-valued process started at $t \in [a, b]$ generated by $-_t D_a$. Note that T_t is the first-passage time of the subordinator $(t - Y_s^{a,t})_{s \in \mathbb{R}_+}$ over the constant level $t - a$.

In the special case $g \equiv 0$, $\phi(x) = x^2$, $A_x = (x^2/2)\Delta$ (where Δ is the Laplacian on $\mathbb{R}$) and the subordinator is tempered stable, we can simulate the Monte Carlo estimator

$$u_n(t, x) = \frac{1}{n} \sum_{k=1}^n \phi(X_{T_t^k}^{x,k}) \tag{3.2}$$

of the solution u of the FPDE in (3.1). In particular, the variables T_t^k are an iid samples of T_t, obtained by TSFFP-Alg, while

$$X_s^{x,k} := x \exp\left(\sqrt{s} N^k - s/2\right),$$

where N^k are iid standard normal random variables. Since X^x is a geometric Brownian motion and TSFFP-Alg is exact, the variables $\phi(X_{T_t^k}^{x,k})$ in the estimator in (3.2) have the same law as $\phi(X_{T_t}^x)$. A plot

of $u_n(t,x)$, based on $n = 10^4$ samples, is presented in Figure 3.4. In this case, the dependence of the estimator $u_n(t,x)$ for fixed t (as a function of x) is explicit. The section of $u_n(t,x)$ as a function of t (for fixed x) is obtained by sampling consecutive crossing times for increasing barriers. This is a well-know procedure in Monte Carlo estimation that retains the accuracy of each estimate $u_n(t,x)$ but reduces random fluctuations between the values of u_n at consecutive time points and a given level x by introducing correlation [14, Sec.4.2]. The spacing between values of t (resp. x) where $u_n(t,x)$ is evaluated in Figure 3.4 is $1/20$ (resp. $1/100$).

3.3.1 Comparison with a biased approximation.

We stress that being able to simulate exactly from the law of T_t is essential for the stability of the estimator $u_n(t,x)$. It is tempting to simulate a random walk approximation (i.e. the skeleton) of the subordinator $(t - Y_s^{a,t})_{s \in \mathbb{R}_+}$ on a time grid with mesh $h > 0$ and approximate the law of T_t with the first time $T_t^{(h)}$ the random walk crosses level $t - a$. Unlike $u_n(t,x)$, the estimator

$$u_n^{(h)}(t,x) = \frac{1}{n} \sum_{k=1}^{n} \phi\left(X_{T_t^{(h),k}}^{x,k}\right), \quad (t,x) \in \mathbb{R}_+^2,$$

is biased. Thus, in order for the L^2-error to be at most $\epsilon^2 > 0$, its bias and variance need to be of order ϵ and ϵ^2, respectively. In the specific case considered here, it is easy to see that, for fixed (t,x), the bias of $u_n^{(h)}(t,x)$ is proportional to h. In particular, the computational complexity of evaluating $\phi(X_{T_t^{(h),k}}^{x,k})$ is proportional to $1/h$, making the total complexity of $u_n^{(h)}(t,x)$ proportional to $n/h \propto \epsilon^{-3}$. In contrast, the computational complexity of evaluating $u_n(t,x)$ based on our exact simulation algorithm is proportional to $n \propto \epsilon^{-2}$. This yields an improvement in computational complexity proportional to $1/h \propto \epsilon^{-1}$.

The analysis outlined in the previous paragraph can be made precise for all Lipschitz functions ϕ. We note that, if ϕ is only locally Lipschitz (as is the case in the example above), the error of the naive random walk algorithm may deteriorate significantly as a function of (t,x). For instance, in the example above, the bias is proportional to $hx^2\mathbb{E}_q[\exp(\sqrt{T_t}N^1 - T_t/2)] = hx^2\mathbb{E}_q[\exp(T_t)]$, where $\mathbb{E}_q[\exp(T_t)] \geqslant \mathbb{E}[\exp(T_t)]$ (because stable paths dominate tempered stable paths) and, by Proposition 4.1(a) and (4.26),

$$\mathbb{E}\left[e^{T_t}\right] = \mathbb{E}\left[e^{t^\alpha S_1^{-\alpha}}\right] = \sum_{k=0}^{\infty} \frac{t^{k\alpha}\mathbb{E}[S_1^{-k\alpha}]}{k!} = \sum_{k=0}^{\infty} \frac{t^{k\alpha}}{\Gamma(1+k\alpha)} \geqslant \frac{1}{\Gamma(1+\alpha)} + \sum_{k=1}^{\infty} \frac{t^{k\alpha}}{k!} = e^{t^\alpha} - 1 + \frac{1}{\Gamma(1+\alpha)}.$$

Thus, the bias of the naive approximation $u_n^{(h)}(t,x)$ is proportional to $hx^2e^{t^\alpha}$. This would require h to be proportional to $\epsilon x^{-2}e^{-t^\alpha}$ if the L^2-error is to be smaller than ϵ^2. Thus, the improvement in computational complexity of $u_n(t,x)$ over $u_n^{(h)}(t,x)$ is proportional to $1/h \propto \epsilon^{-1}x^2e^{t^\alpha}$.

4 Analyses of the algorithms

For any $\theta > 0$ and $\alpha \in (0,1)$, let $S = (S_t)_{t \in \mathbb{R}_+}$ (where $\mathbb{R}_+ := [0,\infty)$) be the stable subordinator (for background on Lévy, including stable, processes see monograph [6]) with Laplace transform and Lévy measure

$$\mathbb{E}[e^{-uS_t}] = \exp(-t\theta u^\alpha), \quad u \in \mathbb{R}_+, \quad \& \quad \nu(dx) := \theta w_\alpha^{-1} x^{-\alpha-1} dx, \quad x \in (0,\infty), \tag{4.1}$$

21
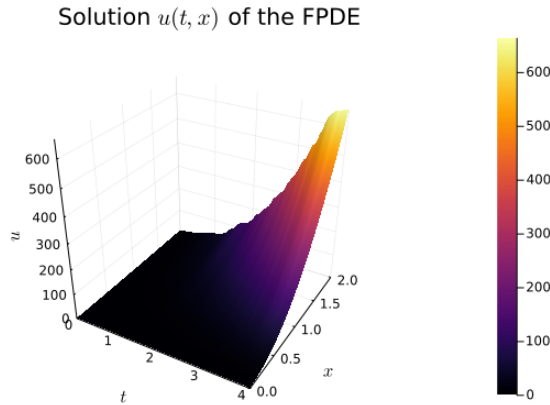
Figure 3.4: Estimated values $u_n(t, x)$ of $u(t, x)$ using $n = 10^4$ samples. We chose $\phi(x) = x^2$. Note T_t has the law of the first-passage time of a tempered stable subordinator with parameters $\alpha = 0.4$, $\theta = 1$, $q = 1$ crossing level $b(t) \equiv 1$ and $X = (X_t)_{t \in \mathbb{R}_+}$ is geometric Brownian motion $X_t = x \exp(B_t - t/2)$ with $X_0 = x > 0$.

respectively, where $w_\alpha := \Gamma(1 - \alpha)/\alpha > 0$ (Γ denotes the Gamma function). The constant factor $1/w_\alpha$ in the Lévy measure $\nu(\mathrm{d}x)$ is chosen so that the Lévy-Khinchin formula for the Laplace exponent yields the desired Laplace exponent $u \mapsto \theta u^\alpha$. Let $\overline{\nu} : (0, \infty) \to (0, \infty)$ be the tail function of the Lévy measure ν:

$$\overline{\nu}(x) := \nu((x, \infty)) = x^{-\alpha}\theta/\Gamma(1 - \alpha), \qquad x \in (0, \infty).$$

In particular, S starts at zero, has zero drift and satisfies $S_t \stackrel{d}{=} t^{1/\alpha}S_1$ for all $t \in (0, \infty)$. Denote by $g_t : \mathbb{R}_+ \to \mathbb{R}_+$ the density of S_t (for $t > 0$). Introduce the functions

$$\rho(u) := \frac{\sin(\alpha\pi u)^\alpha \sin((1 - \alpha)\pi u)^{1-\alpha}}{\sin(\pi u)} \quad \text{and} \quad \sigma_\alpha(u) := \rho(u)^{r+1} = \rho(u)^{1/(1-\alpha)}, \quad u \in (0, 1), \tag{4.2}$$

where $r := \alpha/(1 - \alpha) \in (0, \infty)$. Note $r/\alpha = r + 1$ and hence $\sigma_\alpha(0+) := \lim_{x \downarrow 0} \sigma_\alpha(x) = (1 - \alpha)\alpha^{\alpha/(1-\alpha)}$. The density g_t of S_t can be expressed as follows [26, §4.4]: for any $x > 0$,

$$g_t(x) = \varphi_\alpha((\theta t)^{-1/\alpha}x)(\theta t)^{-1/\alpha}, \quad \text{where} \quad \varphi_\alpha(x) := r \int_0^1 \sigma_\alpha(u)x^{-r-1}\mathrm{e}^{-\sigma_\alpha(u)x^{-r}}\mathrm{d}u. \tag{4.3}$$

In particular, the n-th derivative $g_t^{(n)}$ exists on $(0, \infty)$ for any $n \in \mathbb{N} \cup \{0\}$ and satisfies $\lim_{x \downarrow 0} g_t^{(n)}(x) = 0$.

4.1   Distribution of the first-passage event of a stable subordinator

The paths of S are assumed to be right-continuous with left limits, where for $t > 0$ we define $S_{t-} := \lim_{s \uparrow t} S_s$. The jump at time t is given by $\Delta_S(t) := S_t - S_{t-}$. The first-passage event for S over a non-increasing, absolutely continuous function $b : [0, \infty) \to [0, \infty)$, with $b(0) \in (0, \infty)$, is described by the random vector

$$\left(\tau_b, S_{\tau_b-}, S_{\tau_b}\right), \qquad \text{where} \quad \tau_b := \inf\{t > 0 : S_t > b(t)\} \tag{4.4}$$

is the crossing time of S over the function b and $S_{\tau_b-} = \lim_{s \uparrow \tau_b} S_s$ is the left limit of S at the crossing time of b. The size $\Delta_S(\tau_b) := S_{\tau_b} - S_{\tau_b-}$ of the jump of S at the crossing time τ_b may be zero with positive

probability (i.e. it does not jump at time τ_b) if b is not constant. In this case we say that the subordinator S *creeps* over b.

Theorem A.1, stated and proved in Appendix A.1 below for completeness but originally established in [8] (see also [7]), describes the law of the triplet $(\tau_b, S_{\tau_b-}, \Delta_S(\tau_b))$ (and hence of (4.4), since $S_{\tau_b} = S_{\tau_b-} + \Delta_S(\tau_b)$) for a general class of subordinators. In particular, as a consequence of (A.1a), the stable subordinator S does not jump onto or out of the function b at τ_b:

$$\mathbb{P}(S_{\tau_b-} < b(\tau_b), S_{\tau_b} = b(\tau_b)) = 0 \qquad \& \qquad \mathbb{P}(S_{\tau_b-} = b(\tau_b), S_{\tau_b} > b(\tau_b)) = 0.$$

Note that, since b is monotone and absolutely continuous, it is differentiable Lebesgue-a.e. on $(0, \infty)$ and its derivative is a version of its density. We denote its derivative by b' and, on the Lebesgue zero measure set where it is not defined, we set it equal to -1 (arbitrary choice, without ramifications). The next proposition is key for the sampling of $(\tau_b, S_{\tau_b-}, S_{\tau_b})$.

Proposition 4.1. *Let S be a stable subordinator and $b : \mathbb{R}_+ \to \mathbb{R}_+$ a non-increasing absolutely continuous function with $b(0) > 0$. Define $T_b := \inf\{t > 0 : b(t) = 0\} \in (0, \infty]$ (with convention $\inf \emptyset := \infty$). The following statements hold.*

(a) $\tau_b \overset{d}{=} B^{-1}(S_1)$, *where $B(t) := t^{-1/\alpha} b(t)$ is a strictly decreasing continuous function $B : (0, T_b) \to (0, \infty)$, with inverse $B^{-1} : (0, \infty) \to (0, T_b)$. Hence, $\mathbb{P}(0 < \tau_b < T_b) = 1$.*

(b) *The probability of S creeping at b, conditional on at time $\tau_b = t \in (0, T_b)$, equals*

$$\mathbb{P}[S_{\tau_b} = b(\tau_b) | \tau_b = t] = \mathbb{P}[\Delta_S(\tau_b) = 0 | \tau_b = t] = -b'(t)/(-b'(t) + \alpha^{-1} t^{-1} b(t)).$$

(c) *Conditional on S not creeping at the crossing time $\tau_b = t \in (0, T_b)$, we have*

$$\mathbb{P}[S_{\tau_b-} \in \mathrm{d}u | \Delta_S(\tau_b) > 0, \tau_b = t] = \frac{\theta t}{w_\alpha b(t) g_t(b(t))} (b(t) - u)^{-\alpha} g_t(u) \mathrm{d}u, \quad u \in (0, b(t)); \tag{4.5}$$

$$\mathbb{P}[\Delta_S(\tau_b) \le v | \Delta_S(\tau_b) > 0, \tau_b = t, S_{\tau_b-} = u] = 1 - ((b(t) - u)/v)^\alpha, \quad v \in (b(t) - u, \infty). \tag{4.6}$$

Remark 4.1. (i) The representation of the law of τ_b in Proposition 4.1(a) is based on the scaling property of the stable subordinator S. In particular, Proposition 4.1(a) implies that, at the crossing time τ_b of a stable subordinator over a boundary function b, the law of the variable $\tau_b^{-1/\alpha} b(\tau_b)$ is stable for any non-increasing absolutely continuous function b.

(ii) The expression of the probability of creeping, conditional on $\tau_b = t$, in Proposition 4.1(b) does not depend on the density of S_t. This makes the event of creeping easy to sample, see SFP-Alg below.

(iii) Conditional on not creeping, the jump $\Delta_S(\tau_b)$ is easy to sample by inverting its distribution function in (4.6). In contrast, sampling the undershoot from the law in (4.5) in finite expected running time is delicate, see SU-Alg below.

Proof of Proposition 4.1. (a) By the scaling property of $S_t \overset{d}{=} t^{1/\alpha} S_1$, note $\mathbb{P}[\tau_b \le t] = \mathbb{P}[S_t \ge b(t)] = \mathbb{P}[S_1 \ge B(t)] = \mathbb{P}[B^{-1}(S_1) \le t]$ for any $t > 0$.

(b) We claim that τ_b is absolutely continuous with density given by the function $K : (0, T_b) \to (0, \infty)$,

$$K(t) = g_t(b(t)) \left(\alpha^{-1} t^{-1} b(t) - b'(t) \right) \quad \text{for all } t \in (0, T_b). \tag{4.7}$$

23

In fact, $\mathbb{P}[\tau_b \leqslant t] = \mathbb{P}[B^{-1}(S_1) \leqslant t] = \mathbb{P}[S_1 \geqslant B(t)] = \int_{B(t)}^{\infty} g_1(s)\mathrm{d}s$, which is absolutely continuous since b (and hence B) is absolutely continuous. Differentiating this identity in t and applying the scaling property to the density $g_t(s) = t^{-1/\alpha}g_1(t^{-1/\alpha}s)$ yields

$$K(t) := \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{P}[\tau_b \leqslant t] = \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{P}[B^{-1}(S_1) \leqslant t] = -B(t)'g_1(B(t))$$
$$= g_1(t^{-1/\alpha}b(t))(t^{-1/\alpha-1}b(t)/\alpha - t^{-1/\alpha}b'(t)) = g_t(b(t))\left(\alpha^{-1}t^{-1}b(t) - b'(t)\right).$$

Since $\mathbb{P}[S_{\tau_b} = b(\tau_b)|\tau_b = t] = \mathbb{P}[\Delta_S(\tau_b) = 0|\tau_b = t] = (1/K(t))\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{P}[\Delta_S(\tau_b) = 0, \tau_b \leq t]$, the representation in (4.7) and (A.1b) imply $\mathbb{P}[S_{\tau_b} = b(\tau_b)|\tau_b = t] = -b'(t)/(-b'(t) + \alpha^{-1}t^{-1}b(t))$.

(c) By (A.1a) in Theorem A.1, for $t \in (0, T_b)$ and $u \in (0, b(t))$, we have

$$\mathbb{P}[\tau_b \in \mathrm{d}t, S_{\tau_b-} \in \mathrm{d}u, \Delta_S(\tau_b) > 0] = \mathbb{1}\{0 < b(t) - u\}g_t(u)\bar{\nu}(b(t) - u)\mathrm{d}t\,\mathrm{d}u.$$

Thus, $\mathbb{P}[\tau_b \in \mathrm{d}t, \Delta_s(\tau_b) > 0] = \int_0^{b(t)} \bar{\nu}(b(t) - u)g_t(u)\mathrm{d}u\mathrm{d}t$ and

$$\mathbb{P}[S_{\tau_b-} \in \mathrm{d}u|\Delta_s(\tau_b) > 0, \tau_b = t] = \mathbb{1}\{0 < b(t) - u\}\frac{g_t(u)\bar{\nu}(b(t) - u)}{\int_0^{b(t)} g_t(u')\bar{\nu}(b(t) - u')\mathrm{d}u'}\mathrm{d}u.$$

To evaluate the denominator, let us recall that $K(t)$ is the density of τ_b. Together with (A.1b) we have

$$K(t) = \mathbb{P}[\tau_b \in \mathrm{d}t, \Delta_s(\tau_b) = 0] + \mathbb{P}[\tau_b \in \mathrm{d}t, \Delta_s(\tau_b) > 0] = -b'(t)g_t(b(t)) + \int_0^{b(t)} g_t(u')\bar{\nu}(b(t) - u')\mathrm{d}u'.$$

Combining (4.7) we get $\int_0^{b(t)} g_t(u')\bar{\nu}(b(t) - u')\mathrm{d}u' = g_t(b(t))\alpha^{-1}t^{-1}b(t)$ and (4.5) follows. Also from (A.1a) we know that

$$\mathbb{P}[\Delta_S(\tau_b) \in \mathrm{d}v|\Delta_S(\tau_b) > 0, \tau_b = t, S_{\tau_b-} = u] = \frac{\mathbb{P}[S_{\tau_b-} \in \mathrm{d}u, \Delta_S(\tau_b) \in \mathrm{d}v|\Delta_S(\tau_b) > 0, \tau_b = t]}{\mathbb{P}[S_{\tau_b-} \in \mathrm{d}u|\Delta_S(\tau_b) > 0, \tau_b = t]}$$
$$= \alpha(b(t) - u)^{\alpha}v^{-\alpha-1}\mathrm{d}v \quad \text{for } v \in (b(t) - u, \infty)$$

and (4.6) follows. $\qquad\square$

4.2 Simulation of the first-passage event of a stable process

The task is to justify the validity of SFP-Alg, which samples the triplet $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ of the stable subordinator S under $\mathbb{P}$. The analysis of its expected computational complexity has some technical prerequisites and will thus be deferred to Subsection 4.4.

By Proposition 4.1(a), τ_b has the same law as $B^{-1}(S_1)$ and can be sampled exactly in constant time (i.e. *not* requiring a random number of steps) via the widely used S-Alg given in Appendix C.1 below. Moreover, by Proposition 4.1(b), the probability of the stable subordinator S creeping over the function b (i.e. $\Delta_S(\tau_b) = 0$), given $\tau_b = t$, equals $-b'(t)/\left(-b'(t) + \alpha^{-1}t^{-1}b(t)\right)$. SFP-Alg samples a Bernoulli variable taking the value 1 with probability $-b'(t)/\left(-b'(t) + \alpha^{-1}t^{-1}b(t)\right)$. If the Bernoulli equals 1, then, since the probability of S jumping onto the function b is zero $\mathbb{P}(S_{\tau_b-} < b(\tau_b), S_{\tau_b} = b(\tau_b)) = 0$, SFP-Alg returns $(\tau_b, S_{\tau_b-}, S_{\tau_b}) = (t, b(t), b(t))$. If, conditional on $\tau_b = t$, creeping does not occur, SFP-Alg samples the undershoot S_{τ_b-} from the law $\mathbb{P}[S_{\tau_b-} \in \mathrm{d}u|\Delta_S(\tau_b) > 0, \tau_b = t]$, given in (4.5) of Proposition 4.1(c), using SU-Alg given below. Then, conditional on $S_{\tau_b-} = u$, the jump size $\Delta_S(t)$ has by (4.6) the same law

24

as $(b(t) - u)U^{-1/\alpha}$, where U is an independent uniform on $(0, 1)$. Thus, SU-Alg returns $(\tau_b, S_{\tau_b-}, S_{\tau_b}) = \left(t, u, u + (b(t) - u)U^{-1/\alpha}\right)$.

By the previous paragraph, for SFP-Alg to be valid, it suffices to justify that SU-Alg produces samples from the law $\mathrm{SU}_\alpha(t, w)$ of $S_{\tau_b-}|\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$, which is described analytically in (4.5) of Proposition 4.1(c). Note that the simulation from the law $\mathrm{SU}_\alpha(t, w)$ via SU-Alg is the only step in SFP-Alg that has random running time. In fact, the design of SU-Alg is one of the main contributions of this paper and will be discussed in detail in the following subsection.

4.3   Simulation from the law of the stable undershoot $S_{\tau_b-}|\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$

We now describe and justify SU-Alg and analyse its computational complexity. Recall from Subsection 4.1 above that the function $b : [0, \infty) \to [0, \infty)$ is absolutely continuous and non-decreasing and hence almost everywhere differentiable with derivative b' almost everywhere equal to the density of b and set to be equal to -1 at the points of non-differentiability of b. Further recall that $r = \alpha/(1 - \alpha)$, $w_\alpha = \Gamma(1 - \alpha)/\alpha > 0$ and the definition $T_b > 0$, as well as that of the functions σ_α and φ_α defined in (4.2) and (4.3), respectively.

By (4.5) in Proposition 4.1(c) and the representation of the stable density $g_t(x)$ in (4.3), the density of $\mathrm{SU}_\alpha(t, w)$ is given by

$$f_{t,w}(x) := \frac{1}{w_\alpha w \varphi_\alpha((\theta t)^{-1/\alpha} w)} \varphi_\alpha((\theta t)^{-1/\alpha} x)((\theta t)^{-1/\alpha} w - (\theta t)^{-1/\alpha} x)^{-\alpha}, \qquad x \in (0, w),$$

while the density of $\xi \sim \mathrm{SU}_\alpha(1/\theta, (\theta t)^{-1/\alpha} w)$ equals

$$f_{1/\theta,(\theta t)^{-1/\alpha} w}(x) = \frac{1}{w_\alpha w \varphi_\alpha((\theta t)^{-1/\alpha} w)} (\theta t)^{1/\alpha} \varphi_\alpha(x)((\theta t)^{-1/\alpha} w - x)^{-\alpha}, \qquad x \in (0, (\theta t)^{-1/\alpha} w).$$

Clearly, $(\theta t)^{-1/\alpha} f_{1/\theta,(\theta t)^{-1/\alpha} w}((\theta t)^{-1/\alpha} x) = f_{t,w}(x)$ for $x \in (0, w)$. Thus, $(\theta t)^{1/\alpha} \xi \sim \mathrm{SU}_\alpha(t, w)$. It suffices to sample ξ from the law $\mathrm{SU}_\alpha(1/\theta, s)$ with density $x \mapsto \varphi_\alpha(x)(s - x)^{-\alpha}/(w_\alpha s \varphi_\alpha(s))$, $x \in (0, s)$ and $s = (\theta t)^{-1/\alpha} w$. Sampling efficiently from a density on $(0, s)$, proportional to $x \mapsto \varphi_\alpha(x)(s - x)^{-\alpha}$, is hard to do directly, see Section 2.5. We use the integral representation of $\varphi_\alpha(x)$ in (4.3) to extend the state space: consider (ξ, Y_s) with density

$$(x, y) \mapsto (s - x)^{-\alpha} \sigma_\alpha(y) r x^{-r-1} e^{-\sigma_\alpha(y) x^{-r}}/(w_\alpha s \varphi_\alpha(s)), \quad (x, y) \in (0, s) \times (0, 1).$$

Define a random variable $\zeta_s := \xi^{-r}$. Note that $\zeta_s^{-1/r} \sim \mathrm{SU}_\alpha(1/\theta, s)$, and hence $(\theta t)^{1/\alpha} \zeta_s^{-1/r} \sim \mathrm{SU}_\alpha(t, w)$. Moreover, the law of the random vector (ζ_s, Y_s) is given by the density

$$\psi_s(x, y) := (s - x^{-1/r})^{-\alpha} \sigma_\alpha(y) e^{-\sigma_\alpha(y) x}/(w_\alpha s \varphi_\alpha(s)), \quad (x, y) \in (s^{-r}, \infty) \times (0, 1). \tag{4.8}$$

Fast exact simulation from the law given by the density ψ_s is non-trivial but possible, see SU-Alg. The first step, given in Proposition 4.2, is to dominate the density ψ_s by a mixture of densities that can both be simulated using rejection sampling.

Proposition 4.2. *Fix any $s > 0$. Let $\widetilde{\psi}_s^{(1)}$ (resp. $\widetilde{\psi}_s^{(2)}$) be the density on $(s^{-r}, \infty) \times (0, 1)$ proportional to $(x, y) \mapsto \sigma_\alpha(y) e^{-\sigma_\alpha(y) x}$ (resp. $(x, y) \mapsto \sigma_\alpha(y) e^{-\sigma_\alpha(y) x}(x - s^{-r})^{-\alpha}$). Define the mixture density*

$$\widetilde{\psi}_s := p \widetilde{\psi}_s^{(1)} + (1 - p) \widetilde{\psi}_s^{(2)},$$

where $p := p'/(p'+1) \in (0,1)$ and

$$p' := (2 - 2^\alpha)^{-\alpha} r^{-\alpha} s^{\alpha r} \frac{\int_0^1 \exp(-\sigma_\alpha(y)s^{-r})\mathrm{d}y}{\Gamma(1-\alpha)\int_0^1 \sigma_\alpha(y)^\alpha \exp(-\sigma_\alpha(y)s^{-r})\mathrm{d}y}. \tag{4.9}$$

Then there exists a constant $c_\psi > 0$ such that, for all $(x,y) \in (s^{-r}, \infty) \times (0,1)$, we have

$$\frac{1-\alpha}{2} \leqslant \frac{c_\psi \psi_s(x,y)}{\widetilde{\psi}_s(x,y)} = \frac{(s - x^{-1/r})^{-\alpha}}{(1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha} + (2r)^\alpha s^{-r}(x - s^{-r})^{-\alpha}} \leqslant 1. \tag{4.10}$$

Remark 4.2. The numerical value of the constant c_ψ in Proposition 4.2 is not required in SU-Alg, which samples from the density ψ_s in (4.8). The inequalities in (4.10) enable us to apply rejection sampling to a proposal from $\widetilde{\psi}_s$ to obtain an exact sample from the law given by the density ψ_s. The first inequality in (4.10) explicitly states a decay of the acceptance probability as $\alpha \uparrow 1$. Sampling from $\widetilde{\psi}_s$ requires a mixture of samples from $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$. The respective algorithms are constructed and analysed in Subsections 4.3.1 and 4.3.3 below, respectively.

Proof of Proposition 4.2. Denote the total masses of the positive functions $(x,y) \mapsto \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}$ and $(x,y) \mapsto \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}(x - s^{-r})^{-\alpha}$ by

$$M_1 := \int_0^1 \int_{s^{-r}}^\infty \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}\mathrm{d}x\mathrm{d}y = \int_0^1 \exp(-\sigma_\alpha(y)s^{-r})\mathrm{d}y \qquad \text{and}$$

$$M_2 := \int_0^1 \int_{s^{-r}}^\infty \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}(x - s^{-r})^{-\alpha}\mathrm{d}x\mathrm{d}y = \Gamma(1-\alpha)\int_0^1 \sigma_\alpha(y)^\alpha \exp(-\sigma_\alpha(y)s^{-r})\mathrm{d}y.$$

Then, $\widetilde{\psi}_s^{(1)}(x,y) = \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}/M_1$ and $\widetilde{\psi}_s^{(2)}(x,y) = \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}(x - s^{-r})^{-\alpha}/M_2$. Moreover,

$$p' = \frac{(1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha}M_1}{(2r)^\alpha s^{-r}M_2}.$$

Note that $p' = p/(1-p)$, making $\widetilde{\psi}_s$ proportional to

$$(2r)^\alpha s^{-r}M_2(p'\widetilde{\psi}_s^{(1)}(x,y) + \widetilde{\psi}_s^{(2)}(x,y)) = (1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha}M_1\widetilde{\psi}_s^{(1)}(x,y) + (2r)^\alpha s^{-r}M_2\widetilde{\psi}_s^{(2)}(x,y)$$
$$= \sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}\big((1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha} + (2r)^\alpha s^{-r}(x - s^{-r})^{-\alpha}\big).$$

Let $c_\psi > 0$ be the normalizing constant, such that for $(x,y) \in (s^{-r}, \infty) \times (0,1)$,

$$\widetilde{\psi}_s(x,y) = \frac{c_\psi}{w_\alpha s \varphi_\alpha(s)}\sigma_\alpha(y)\mathrm{e}^{-\sigma_\alpha(y)x}\big((1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha} + (2r)^\alpha s^{-r}(x - s^{-r})^{-\alpha}\big). \tag{4.11}$$

Thus $c_\psi \psi_s(x,y)/\widetilde{\psi}(x,y) = (s - x^{-1/r})^{-\alpha}/\big((1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha} + (2r)^\alpha s^{-r}(x - s^{-r})^{-\alpha}\big)$ in (4.10) follows.

We now prove the inequalities in (4.10). Since $x \mapsto (1-\alpha)s^{r+1}x + \alpha x^{-1/r}$ is increasing on (s^{-r}, ∞), we have $(1-\alpha)s^{r+1}x + \alpha x^{-1/r} \geqslant s$, equivalently $r^\alpha s^{-r}(x - s^{-r})^{-\alpha} \leqslant (s - x^{-1/r})^{-\alpha}$, for all $x > s^{-r}$. This inequality and a direct calculation imply $((1 - 2^{\alpha-1})^{-\alpha} + 2^\alpha)^{-1} \leqslant c_\psi \psi_s(x,y)/\widetilde{\psi}_s(x,y)$ for all $(x,y) \in (s^{-r}, \infty) \times (0,1)$. Since $(1-\alpha)/2 \leq ((1 - 2^{\alpha-1})^{-\alpha} + 2^\alpha)^{-1}$, the inequality on the left side of (4.10) follows.

Since $(1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha} \geqslant (s - x^{-1/r})^{-\alpha}$ for $x \geqslant 2^\alpha s^{-r}$ (note that $x \mapsto (s - x^{-1/r})^{-\alpha}$ is decreasing and equal to $(1 - 2^{\alpha-1})^{-\alpha}s^{-\alpha}$ at $x = 2^\alpha s^{-r}$), the inequality on the right side of (4.10) holds for $x \geqslant 2^\alpha s^{-r}$. Consider the case $s^{-r} < x < 2^\alpha s^{-r}$. Since $x \mapsto (1-\alpha)s^{r+1}x + 2\alpha x^{-1/r}$ decreases on $[s^{-r}, 2^\alpha s^{-r}]$, after substituting $r = \alpha/(1-\alpha)$, we obtain $(1-\alpha)s^{1/(1-\alpha)}x + 2\alpha x^{-(1-\alpha)/\alpha} \leqslant (1+\alpha)s$. This inequality is equivalent to $(2\alpha/(1-\alpha))^\alpha s^{-\alpha/(1-\alpha)}(x - s^{-\alpha/(1-\alpha)})^{-\alpha} \geqslant (s - x^{-(1-\alpha)/\alpha})^{-\alpha}$. Substituting $r = \alpha/(1-\alpha)$, we obtain $(2r)^\alpha s^{-r}(x - s^{-r})^{-\alpha} \geqslant (s - x^{-1/r})^{-\alpha}$, and the right side of (4.10) follows. $\square$

4.3.1 Simulation from the density $\widetilde{\psi}_s^{(1)}$: proof of Proposition 2.5

The proof of Proposition 2.5 requires the following two elementary lemmas about certain properties of the function σ_α defined in (4.2).

Lemma 4.3. *The derivative of any order of $\log \sigma_\alpha$, and hence of σ_α, is positive on $(0, 1)$.*

Proof. Clearly, by definition (4.2), $\log(\sigma_\alpha(x)) = (1 - \alpha)^{-1}(\alpha \psi_\alpha(x) + (1 - \alpha)\psi_{1-\alpha}(x))$, where $\psi_a : x \mapsto \log(\sin(a\pi x)) - \log(\sin(\pi x))$ for $a \in (0, 1)$. Recall that $\cot(\pi x) = (\pi x)^{-1} - (2/\pi) \sum_{n=1}^{\infty} \zeta(2n)x^{2n-1}$ for $|x| \in (0, 1)$ where ζ is the Riemann zeta function. Thus, for any $a \in (0, 1)$, we have

$$\psi_a'(x) = a\pi \cot(a\pi x) - \pi \cot(\pi x) = 2\sum_{n=1}^{\infty} \zeta(2n)(1 - a^{2n})x^{2n-1}, \quad x \in (0, 1).$$

This shows that ψ_a' and all its derivatives are positive on $(0, 1)$, implying the same is true of $\log(\sigma_\alpha)$. Thus, $\sigma_\alpha'' = (\mathrm{e}^{\log(\sigma_\alpha)})'' = \mathrm{e}^{\log(\sigma_\alpha)}(\log(\sigma_\alpha)'^2 + \log(\sigma_\alpha)'') \geq 0$, making σ_α convex on $(0, 1)$. Similarly, a derivative of σ_α of any order is a linear combination with non-negative coefficients of derivatives of $\log \sigma_\alpha$, multiplied with $\mathrm{e}^{\log(\sigma_\alpha)}$, and thus non-negative itself on the interval $(0, 1)$. $\qquad\square$

Lemma 4.4. *For $x \in (0, 1/2)$, we have $0 \leqslant \sigma_\alpha(x) - \sigma_\alpha(0+) \leqslant (\pi - 2\mathrm{e}^{-1})(1 - \alpha)x$.*

Proof. By Lemma 4.3, σ_α is convex and increasing. Thus

$$0 \leqslant (\sigma_\alpha(x) - \sigma_\alpha(0+))/x \leqslant (\sigma_\alpha(1/2) - \sigma_\alpha(0+))/(1/2),$$

implying $\sigma_\alpha(x) - \sigma_\alpha(0+) \leqslant 2(\sigma_\alpha(1/2) - \sigma_\alpha(0+))x$. For $t \in (0, \pi)$, we have $\sin(t) \leqslant \min\{1, t\}$ and hence

$$\sigma_\alpha(1/2) = \sin(\alpha\pi/2)^r \sin((1 - \alpha)\pi/2) \leqslant (1 - \alpha)\pi/2. \tag{4.12}$$

By $\log(1/\alpha) \leqslant 1/\alpha - 1$ we have $(1 - \alpha)^{-1}\alpha \log \alpha \geqslant -1$. Thus $\sigma_\alpha(0+) = \alpha^r(1 - \alpha) \geqslant \mathrm{e}^{-1}(1 - \alpha)$ and (4.12) imply $\sigma_\alpha(x) - \sigma_\alpha(0+) \leq (1 - \alpha)\pi/2 - \mathrm{e}^{-1}(1 - \alpha)$, proving the lemma. $\qquad\square$

Proof of Proposition 2.5. Recall that σ_α is convex by Lemma 4.3. To sample from $\widetilde{\psi}_s^{(1)}$, we first sample Y_s from the marginal log-concave density proportional to $\xi(y) := \exp(-(\sigma_\alpha(y) - \sigma_\alpha(0+))s^{-r})$ on $(0, 1)$ via LC-Alg. Then, conditional on $Y_s = y$, we sample ζ_s from the law whose density is proportional to the function $x \mapsto \sigma_\alpha(y)\exp(-\sigma_\alpha(y)x)\mathbb{1}_{\{x>s^{-r}\}}$ on $x \in (s^{-r}, \infty)$. Note that this makes ζ_s, conditional on Y_s, into a shifted exponential random variable.

As explained in Section C.2 below, the computational cost of LC-Alg has two parts: (I) the cost of finding the value of a_1 in line 1 of LC-Alg and (II) the expected cost of the accept-reject step in LC-Alg. By [11, §4], the expected cost of (II) is bounded above by 5.

Note that cost (I) of finding a_1 within $\psi^{(1)}$-Alg is not constant in the variable s, which is itself random in SFP-Alg. Thus we need to analyse the cost of finding a_1 in LC-Alg as a function of s. By Lemma 4.4, for $y \in (0, 1/2)$, we have $\xi(y) \geqslant \exp(-(\pi - 2\mathrm{e}^{-1})(1-\alpha)ys^{-r})$. Thus we set $y := (\pi - 2\mathrm{e}^{-1})^{-1}(\log 4)(1 - \alpha)^{-1}s^r$ and note that the following inequality holds: $\xi(y) \geqslant 1/4$. Therefore the number of steps in the binary search in Step 1 of LC-Alg is bounded above by

$$1 + \log(1/y) \leqslant 1 + \log^+\left((\pi - 2\mathrm{e}^{-1})(\log 4)^{-1}(1 - \alpha)s^{-r}\right) \leqslant \kappa_{(1)}(1 + \alpha(1 - \alpha)^{-1}\log_2^+(s^{-1})),$$

where the constant in $\kappa_{(1)}$ depends neither on $s \in (0, \infty)$ nor $\alpha \in (0, 1)$, implying the claim. $\qquad\square$

Before proving Proposition 2.6, we first clarify in the next subsection the the time cost of applying the Newton–Raphson method in lines 1, 8, 15 and 21 of $\psi^{(2)}$-Alg. As we shall see, the costs depend on α and may, but need not, depend on s.

4.3.2   Time cost of applying the Newton–Raphson method in $\psi^{(2)}$-Alg.

Inverting σ_α is equivalent to inverting the convex increasing function

$$x \mapsto \log\left(\frac{\sin(\alpha\pi x)^\alpha \sin((1-\alpha)\pi x)^{1-\alpha}}{\sin(\pi x)}\right), \quad x \in (0,1).$$

Equivalently, for any $s > \alpha \log\alpha + (1-\alpha)\log(1-\alpha)$, we must find the root of the convex increasing function $x \mapsto \varsigma(x) - s$ where

$$\varsigma(x) := \alpha \log\sin(\alpha\pi x) + (1-\alpha)\log\sin((1-\alpha)\pi x) - \log\sin(\pi x), \quad x \in (0,1). \tag{4.13}$$

Proposition 4.5. *Define the auxiliary function*

$$M(x,k) := \frac{1}{2\pi\alpha(1-\alpha)(x - 2^{1-k})x^2(1-x)^2}. \tag{4.14}$$

For any $s > \alpha\log\alpha + (1-\alpha)\log(1-\alpha)$, the computational cost of finding the root of $\varsigma(x)$ via NR-Alg with auxiliary function $M(x,k)$ is $\mathcal{O}(|\log(x_*(1-x_*)) + \log(\alpha(1-\alpha))| + \log N)$ where $x_* = \varsigma^{-1}(s)$ is the solution and N specifies the number of precision bits.

Proof. Since $\varsigma(x)$ is convex, if our initial estimate x_0 (obtained via binary search) of the root is larger than the true root, then the Newton–Raphson sequence

$$x_{n+1} = x_n - \frac{\varsigma(x_n) - s}{\varsigma'(x_n)}, \quad n \in \mathbb{N},$$

decreases and has quadratic convergence to the root x_*. Using the power series of the cotangent function (as in the proof of Lemma 4.3) and the inequality $\sin(\pi x) \geqslant \pi x(1-x)$, for (x_0, k) satisfying $x_* \leqslant x_0 - 2^{1-k}$, we have

$$\begin{aligned}
\frac{1}{2}\frac{\sup_{x\in[x_*,x_0]}|\varsigma''(x)|}{\inf_{x\in[x_*,x_0]}|\varsigma'(x)|} &\leqslant \frac{\pi}{2}\left|\frac{\csc^2(\pi x_0) - \alpha^3\csc^2(\alpha\pi x_0) - (1-\alpha)^3\csc^2((1-\alpha)\pi x_0)}{\alpha^2\cot(\alpha\pi x_*) + (1-\alpha)^2\cot((1-\alpha)\pi x_*) - \cot(\pi x_*)}\right| \\
&\leqslant \frac{3\csc^2(\pi x_0)}{2(1-\alpha^3-(1-\alpha)^3)x_*} \leqslant \frac{3}{2\pi(1-\alpha^3-(1-\alpha)^3)x_* x_0^2(1-x_0)^2} \\
&\leqslant \frac{1}{2\pi\alpha(1-\alpha)(x_0 - 2^{1-k})x_0^2(1-x_0)^2} = M(x_0, k).
\end{aligned}$$

Since $(x_0 - 2^{1-k})x_0^2(1-x_0)^2 \geqslant x_*^3(1-x_* - 2^{-k})^2$, by the stopping condition of the binary search of line 7 in NR-Alg, we only need

$$2^{1-k} \cdot \frac{1}{2\pi\alpha(1-\alpha)x_*^3(1-x_* - 2^{-k})^2} \leqslant \frac{1}{2}.$$

implying $k = \mathcal{O}(|\log(x_*(1-x_*)) + \log(\alpha(1-\alpha))|)$. Thus, the binary search for this $x_0 > x_*$ requires $\mathcal{O}(|\log(x_*(1-x_*)) + \log(\alpha(1-\alpha))|)$ steps. $\qquad\square$

Proposition 4.6. *Define the auxiliary function*

$$M(x,k) := \frac{\sigma_\alpha(1/2)^\alpha}{\sigma_\alpha(0)^\alpha}\left(1 + \frac{x_0\alpha}{2}\left(\sigma_\alpha''(x)/\sigma_\alpha(x) + (\alpha-1)\sigma_\alpha'(x)^2/\sigma_\alpha(x)^2\right)\right). \tag{4.15}$$

The total cost of numerical inversion of the function $u \mapsto u\sigma_\alpha(u)^\alpha$ on the interval $[0, z_*] \subset [0, 1/2]$ via NR-Alg with auxiliary function $M(x,k)$ is $\mathcal{O}(|\log(1-\alpha)| + \log N)$ where N specifies the number of precision bits.

Proof. On $[0, z_*]$, the function is convex and increasing with derivative $\sigma_\alpha(u)^\alpha + u\alpha\sigma_\alpha'(u)\sigma_\alpha(u)^{\alpha-1}$. Thus, given $y \in (0, \sigma(z_*))$, if our initial estimate x_0 (obtained via binary search) of the root x_* of the function $u\sigma_\alpha(x)^\alpha - y$ is larger than x_* and sufficiently close to x_*, then the Newton–Raphson sequence

$$x_{n+1} = x_n - \frac{x_n\sigma_\alpha(x_n)^\alpha - y}{\sigma_\alpha(x_n)^\alpha + x_n\alpha\sigma_\alpha'(x_n)\sigma_\alpha(x_n)^{\alpha-1}} = x_n - \frac{x_n - y/\sigma_\alpha(x_n)^\alpha}{1 + x_n\alpha\sigma_\alpha'(x_n)/\sigma_\alpha(x_n)}, \quad n \in \mathbb{N},$$

is decreasing and has quadratic convergence to the root x_*. Note that $\log(\sigma_\alpha)'' = \sigma_\alpha''/\sigma_\alpha - (\sigma_\alpha'/\sigma_\alpha)^2$ and $\log(\sigma_\alpha)' = \sigma_\alpha'/\sigma_\alpha$ are increasing, and thus

$$\frac{1}{2}\frac{\sup_{x\in[x_*,x_0]}|(x\sigma_\alpha(x)^\alpha)''(x)|}{\inf_{x\in[x_*,x_0]}|(x\sigma_\alpha(x)^\alpha)'(x)|} = \frac{1}{2}\frac{\sup_{x\in[x_*,x_0]}(2\sigma_\alpha(x)^\alpha + x\alpha(\sigma_\alpha''(x)\sigma_\alpha(x)^{\alpha-1} + (\alpha-1)\sigma_\alpha'(x)^2\sigma_\alpha(x)^{\alpha-2}))}{\inf_{x\in[x_*,x_0]}(\sigma_\alpha(x)^\alpha + x\alpha\sigma_\alpha'(x)\sigma_\alpha(x)^{\alpha-1})}$$

$$\leqslant \frac{\sigma_\alpha(1/2)^\alpha}{\sigma_\alpha(0)^\alpha}\frac{1 + x_0\alpha(\sigma_\alpha''(x_0)/\sigma_\alpha(x_0) + (\alpha-1)\sigma_\alpha'(x_0)^2/\sigma_\alpha(x_0)^2)/2}{1 + (x_0 - 2^{1-k}z_*)\alpha\sigma_\alpha'(x_0 - 2^{1-k}z_*)/\sigma_\alpha(x_0 - 2^{1-k}z_*)}$$

$$\leqslant \frac{\sigma_\alpha(1/2)^\alpha}{\sigma_\alpha(0)^\alpha}(1 + \frac{x_0\alpha}{2}(\sigma_\alpha''(x_0)/\sigma_\alpha(x_0) + (\alpha-1)\sigma_\alpha'(x_0)^2/\sigma_\alpha(x_0)^2)) = M(x_0, k).$$

Since $z_* < 1/2$,

$$\frac{\sigma_\alpha(1/2)^\alpha}{\sigma_\alpha(0)^\alpha} = \left(\frac{\sin(\alpha\pi/2)\sin(1-\alpha)\pi/2}{\alpha^\alpha(1-\alpha)^{1-\alpha}}\right)^{\alpha^2/(1-\alpha)}$$

has uniform upperbound for $\alpha \in (0,1)$ and

$$1 + x_0\alpha(\sigma_\alpha''(x_0)/\sigma_\alpha(x_0) + (\alpha-1)\sigma_\alpha'(x_0)^2/\sigma_\alpha(x_0)^2)/2 \leqslant 1 + \tfrac{1}{4}\left((\log\sigma_\alpha)''(1/2) + \alpha((\log\sigma_\alpha)'(1/2)^2\right)$$

$$= 1 + \frac{1}{4}\left(-\frac{\alpha}{1-\alpha}\left(\frac{\alpha\pi}{\sin(\alpha\pi/2)}\right)^2 - \left(\frac{(1-\alpha)\pi}{\sin((1-\alpha)\pi/2)}\right)^2 + \pi^2 + \alpha\left(\frac{\alpha}{1-\alpha}\frac{\cos(\alpha\pi/2)}{\sin(\alpha\pi/2)} + \cos(\alpha\pi/2)\right)^2\right)^2$$

By the stopping condition of the binary search of line 7 in NR-Alg, we only need the final expression in the previous display to be smaller than 2^{k-2}, implying $k = \mathcal{O}(|\log(1-\alpha)|)$. Thus, the binary search for $x_0 > x_*$ requires $\mathcal{O}(|\log(1-\alpha)|)$ iterations. $\qquad\square$

For $z \in (1/2, 1)$, define the function

$$F(x) = \sin(\pi(1-\alpha))(1-r)^{-1}(1-x)^{1-r} + \pi\alpha(1-\alpha)\cos(\pi\alpha)(2-r)^{-1}(1-x)^{2-r}, \quad x \in (1/2, z). \tag{4.16}$$

Proposition 4.7. *Define auxiliary function*

$$M(x,k) := \frac{(1-z)^{-(r+1)}(r\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-r))}{(1-1/2)^{-r}\left(\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-z)\right)}. \tag{4.17}$$

For $\alpha \in (0, 1/2)$, the total cost of numerical inversion of the function F via NR-Alg with auxiliary function $M(x,k)$ is $\mathcal{O}(|\log(1-z)| + \log N)$ where N specifies the number of precision bits.

Proof. Note that $F''(x) = (1-x)^{-(r+1)}(r\sin(\pi(1-\alpha)) - \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-r)(1-x))$ for $x \in (1/2, z)$. Given $y \in (F(z), F(1/2))$, if our initial estimate x_0 (obtained via binary search) of the root x_* of the function $F(x) - y$ is larger than x_* and sufficiently close to x_*, then the Newton–Raphson sequence

$$x_{n+1} = x_n - \frac{\sin(\pi(1-\alpha))(1-r)^{-1}(1-x_n)^{1-r} + \pi\alpha(1-\alpha)\cos(\pi\alpha)(2-r)^{-1}(1-x_n)^{2-r} - y}{(1-x_n)^{-r}(\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-x_n))} \quad n \in \mathbb{N},$$

has quadratic convergence to the root x_*. Note that

$$\frac{1}{2}\frac{\sup_{x\in[x_*,x_0]}|F''(x)|}{\inf_{x\in[x_*,x_0]}|F'(x)|} = \frac{1}{2}\frac{\sup_{x\in[1/2,z]}|(1-x)^{-(r+1)}(r\sin(\pi(1-\alpha)) - \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-r)(1-x))|}{\inf_{x\in[1/2,z]}|(1-x)^{-r}(\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-x))|}$$

$$\leqslant \frac{(1-z)^{-(r+1)}(r\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-r))}{(1-1/2)^{-r}(\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-z))} = M(x_0, k)$$

By the stopping condition of the binary search of line 7 in NR-Alg, we only need

$$2^{1-k}\frac{(1-z)^{-(r+1)}(r\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-r))}{(1-1/2)^{-r}(\sin(\pi(1-\alpha)) + \pi\alpha(1-\alpha)\cos(\pi\alpha)(1-z))} \leqslant \frac{1}{2},$$

implying $k \geqslant \mathcal{O}(|\log(1-z)|)$. Thus the binary search for $x_0 > x_*$ requires $\mathcal{O}(|\log(1-z)|)$ iterations. $\qquad\square$

4.3.3 Simulation from the density $\widetilde{\psi}_s^{(2)}$: proof of Proposition 2.6.

Before proving Proposition 2.6, we need following lemmas, describing the properties of functions ρ and σ_α.

Lemma 4.8. *The function $1/\rho$, where ρ is defined in (4.2), is concave on $(0, 1)$.*

Proof. Clearly, by definition (4.2), $\log(\sigma_\alpha(x)) = (1-\alpha)^{-1}(\alpha\psi_\alpha(x) + (1-\alpha)\psi_{1-\alpha}(x))$, where $\psi_a : x \mapsto \log(\sin(a\pi x)) - \log(\sin(\pi x))$ for $a \in (0, 1)$. Recall that $\cot(\pi x) = (\pi x)^{-1} - (2/\pi)\sum_{n=1}^{\infty}\zeta(2n)x^{2n-1}$ for $|x| \in (0, 1)$ where ζ is the Riemann zeta function. Thus, for any $a \in (0, 1)$,

$$\psi_a'(x) = a\pi\cot(a\pi x) - \pi\cot(\pi x) = 2\sum_{n=1}^{\infty}\zeta(2n)(1 - a^{2n})x^{2n-1}, \quad x \in (0, 1).$$

This shows that ψ_a' and all its derivatives are positive on $(0, 1)$, implying the same is true of $\log(\sigma_\alpha)$. Thus, $\sigma_\alpha'' = (e^{\log(\sigma_\alpha)})'' = e^{\log(\sigma_\alpha)}(\log(\sigma_\alpha)'^2 + \log(\sigma_\alpha)'') \geq 0$, making σ_α convex on $(0, 1)$. Similarly, a derivative of σ_α of any order is a linear combination with non-negative coefficients of derivatives of $\log\sigma_\alpha$, multiplied with $e^{\log(\sigma_\alpha)}$, and thus non-negative itself on the interval $(0, 1)$.

To prove that $1/\rho$ is concave, let $f := \log\rho = (1-\alpha)\log\sigma_\alpha$ and note that it suffices to show that $f'' \geqslant (f')^2$. If $\alpha = 1/2$, then $x \mapsto f''(x) - (f'(x))^2$ is constant since $\frac{d}{dx}\cot(x) = -1 - \cot^2(x)$ and the double angle formula $2\cot(2x)\cot(x) = \cot(x)^2 - 1$ imply that $f''(x) = \pi^2(\frac{3}{4} + \cot^2(\pi x) - \frac{1}{4}\cot^2(\frac{\pi}{2}x))$, and

$$f'(x)^2 = \pi^2\big(\cot^2(\pi x) + \tfrac{1}{4}\cot^2(\tfrac{\pi}{2}x) - \cot(\pi x)\cot(\tfrac{\pi}{2}x)\big) = \pi^2\big(\tfrac{1}{2} + \cot^2(\pi x) - \tfrac{1}{4}\cot^2(\tfrac{\pi}{2}x)\big) = f''(x) - \tfrac{\pi^2}{4}.$$

To prove the general case $\alpha \in (0, 1)$, denote $a_n(\alpha) := \zeta(2n)(1 - \alpha^{2n+1} - (1-\alpha)^{2n+1})$, then we have $f'(x) = 2\sum_{n=1}^{\infty}a_n(\alpha)x^{2n-1}$, $x \in (0, 1)$, implying that

$$f''(x) = 4\sum_{n=1}^{\infty}(n - \tfrac{1}{2})a_n(\alpha)x^{2n-2} \quad \text{and} \quad f'(x)^2 = 4\sum_{n=2}^{\infty}\left(\sum_{k=1}^{n-1}a_k(\alpha)a_{n-k}(\alpha)\right)x^{2n-2}.$$

We will show that $f''(x) \geqslant f'(x)^2$ termwise. When $\alpha = 1/2$, we have $f'' = (f')^2 + \pi^2/4$ so all the coefficients of f'' and $(f')^2$ of x^{2n-2} for $n \geqslant 2$ agree:

$$\left(n - \tfrac{1}{2}\right)\zeta(2n) = \sum_{k=1}^{n-1} \zeta(2k)\zeta(2n-2k)\frac{(1-4^{-k})(1-4^{k-n})}{1-4^{-n}}, \quad n \geqslant 2.$$

To establish the termwise inequality for general $\alpha \in (0,1)$, it thus suffices to show that

$$\left(n - \tfrac{1}{2}\right)\zeta(2n) \geqslant \sum_{k=1}^{n-1} \zeta(2k)\zeta(2n-2k)p_{n,k}(\alpha), \quad \text{where}$$

$$p_{n,k}(\alpha) := \frac{(1 - \alpha^{2k+1} - (1-\alpha)^{2k+1})(1 - \alpha^{2(n-k)+1} - (1-\alpha)^{2(n-k)+1})}{(1 - \alpha^{2n+1} - (1-\alpha)^{2n+1})}.$$

Since the equality is attained when $\alpha = 1/2$, it suffices to show that $p_{n,k}$ attains its maximum at $\alpha = 1/2$. Since $p_{n,k}(1/2) > 0 = p_{n,k}(0+) = p_{n,k}(1-)$, it suffices to show that $1/2$ is the only critical point of $p_{n,k}$. This will follow once we show that the derivative

$$q_m(x) := \tfrac{\mathrm{d}}{\mathrm{d}x} \log(1 - x^m - (1-x)^m) = m\frac{(1-x)^{m-1} - x^{m-1}}{1 - x^m - (1-x)^m},$$

is positive and decreasing (resp. negative and increasing) in $m \geqslant 3$ for any $x \in (0, 1/2)$ (resp. $x \in (1/2, 1)$). Indeed, $\tfrac{\mathrm{d}}{\mathrm{d}x} \log(p_{n,k}(x)) = q_{2k+1}(x) + q_{2(n-k)+1}(x) - q_{2n+1}(x)$ so the only solution to $\tfrac{\mathrm{d}}{\mathrm{d}x}p_{n,k}(x) = 0$ is $x = 1/2$ (where $q_m(1/2) = 0$ for any $m \geqslant 3$).

For $3 \leqslant i < m$, the inequality $q_i(x) > q_m(x)$ for $x \in (0, 1/2)$ is equivalent to

$$A_{i,m}(c) := i(c^i - 1)((c+1)^m - c^m - 1) - m(c^m - 1)((c+1)^i - c^i - 1) > 0, \quad \text{where} \quad c := \frac{1-x}{x} > 1.$$

This inequality is further equivalent to

$$A_{i,m}(c) = i\sum_{j=1}^{m-1}\binom{m}{j}c^{j+i} + m\sum_{j=1}^{i-1}\binom{i}{j}c^j - \left(m\sum_{j=1}^{i-1}\binom{i}{j}c^{j+m} + i\sum_{j=1}^{m-1}\binom{m}{j}c^j\right) > 0.$$

It is easy to see that the positive and negative coefficients of the powers of c are the same (with different corresponding powers of c). An elementary but tedious calculation shows that $A_{i,m}(c)$ can be rewritten as

$$A_{i,m}(c) = \sum_{j=1}^{i-1}\left(\binom{m}{m-j}i - \binom{i}{j}m\right)(c^{m-j+i} - c^j)$$

$$+ \sum_{j=i}^{i+\lfloor(m-i)/2\rfloor}\left(\binom{m}{m-j}i - \binom{m}{m+i-j}i\right)(c^{m-j+i} - c^j).$$

Since $c > 1$ and $\binom{m}{j}i > \binom{i}{j}m$ for $j \leqslant i-1$ and $\binom{m}{j}i \geqslant \binom{m}{j-i}i$ for $i \leqslant j \leqslant i + (m-i)/2$, we deduce that $A_{i,m}(c) > 0$ completing the proof. $\square$

Lemma 4.9. *For $x \in (0,1)$, we have*

$$4^{-1/(1-\alpha)} \sin(\alpha\pi)^{1/(1-\alpha)}(1-x)^{-1/(1-\alpha)} \leqslant \sigma_\alpha(x) \leqslant \alpha^{\alpha/(1-\alpha)}(1-\alpha)(1-x)^{-1/(1-\alpha)}.$$

In particular, if $z \in (0,1)$ satisfies $\sigma_\alpha(z) = \alpha s^r$ for some $s > 0$ (recall $r = \alpha/(1-\alpha)$), we have

$$1/(1-z) \leqslant 4\alpha^{1-\alpha}s^\alpha/\sin(\alpha\pi). \tag{4.18}$$

Proof. By the concavity of function $x \mapsto \sin(x)$, we have $\sin(\alpha\pi x) \geqslant \sin(\alpha\pi)x$ and $\sin((1-\alpha)\pi x) \geqslant \sin((1-\alpha)\pi)x$ for $x \in [0,1]$. Moreover, we have $\sin(\alpha\pi x) \leqslant \alpha\pi x$ and $\pi x(1-x) \leqslant \sin(\pi x) \leqslant 4x(1-x)$. $\quad\square$

Lemma 4.10. *For* $x \in (0,1)$, $\sigma'_\alpha(x) \leqslant \frac{d}{dx}[\alpha^r(1-\alpha)(1-x)^{-(r+1)}] = \alpha^r(1-\alpha)(r+1)(1-x)^{-(r+2)}$.

Proof. Note that $\sigma'_\alpha(x) = \sigma_\alpha(x)(\log(\sigma_\alpha))'(x)$. By Lemma 4.9, we only need to show that

$$(r+1)^{-1}(\log(\sigma_\alpha))'(x) = \alpha^2\pi\cot(\alpha\pi x) + (1-\alpha)^2\pi\cot((1-\alpha)\pi x) - \pi\cot(\pi x) \leqslant (1-x)^{-1}.$$

The power series expansion of cot shows that the left-hand side is equivalent to

$$\sum_{n=1}^{\infty} 2\zeta(2n)(1 - \alpha^{2n+1} - (1-\alpha)^{2n+1})x^{2n-1}$$

where ζ is Riemann zeta function. Note that $1 - \alpha^{2n+1} - (1-\alpha)^{2n+1}$ attains its maximum value when $\alpha = 1/2$, so it suffices to establish the following inequality

$$\pi\cot(\pi x/2)/2 - \pi\cot(\pi x) \leqslant (1-x)^{-1}.$$

In fact, let $t = \cot(\pi x/2)$, then

$$\frac{\pi t}{2} - \frac{\pi(t^2-1)}{2t} = \frac{\pi}{2t} = \frac{\pi}{2}\tan(\pi x/2) = \frac{\pi}{2}\cot(\pi(1-x)/2) \leqslant \frac{1}{1-x}. \qquad\square$$

Now we can complete the proof of Proposition 2.6.

Proof of Proposition 2.6. In line 1, $\psi^{(2)}$-Alg sets $z := 0$ when $\alpha s^r < \sigma_\alpha(0+) = (1-\alpha)\alpha^{\alpha/(1-\alpha)}$. Otherwise the equation $\sigma_\alpha(x) = \alpha s^r$, $x \in (0,1)$, has a unique solution by Lemma 4.3 and we may define $z := \sigma_\alpha^{-1}(\alpha s^r)$ to be that solution. This inversion is carried out by the Newton–Raphson method, converging quadratically fast since σ_α is log-convex, see Section C.3 below for details. By Proposition 4.5 the computational cost of finding z is bounded above by $\mathcal{O}(\log(1/(z(1-z))) + |\log(\alpha(1-\alpha))| + \log N)$. Since $\sigma_\alpha(z) - \sigma_\alpha(0+) = \alpha s^r - \alpha^r(1-\alpha)$, if $z < 1/2$, Lemma 4.4 implies $1/z \leqslant (\pi - 2e^{-1})(1-\alpha)/(\alpha s^r - \alpha^r(1-\alpha))$. Hence, by (4.18) in Lemma 4.9, we obtain

$$\frac{1}{z}\frac{1}{1-z} \leqslant \frac{(\pi - 2e^{-1})(1-\alpha)}{\alpha s^r - \alpha^r(1-\alpha)}\frac{4\alpha^{1-\alpha}s^\alpha}{\sin(\alpha\pi)} \leqslant \frac{4(\pi - 2e^{-1})s^\alpha}{\pi\alpha^2(s^r - \alpha^{r-1}(1-\alpha))},$$

where the second inequality follows from $(1-\alpha)\alpha^{-\alpha}/\sin(\alpha\pi) \leqslant \alpha^{-2}/\pi$, which holds since for every $\alpha \in (0,1)$ we have $\sin(\pi\alpha) \geqslant \pi\alpha(1-\alpha)$. Thus, the computational cost of finding z is bounded above by

$$\mathcal{O}\left(\log^+\left(1/(s^r - \alpha^{r-1}(1-\alpha))\right) + \alpha\log^+(s) + |\log(\alpha)| + |\log(1-\alpha)| + \log N\right).$$

The simulation from $\widetilde{\psi}_s^{(2)}$ can be decomposed into two steps: first sample from the marginal density proportional to $\tilde{f}(y) = \sigma_\alpha(y)^\alpha \exp(-\sigma_\alpha(y)s^{-r})$ and then from the conditional law proportional to $x \mapsto \mathbb{1}\{x > s^{-r}\}(x - s^{-r})^{-\alpha}\exp(-\sigma_\alpha(y)(x - s^{-r}))$ (which is a gamma with shape $1 - \alpha$ and rate $\sigma_\alpha(y)$ and translated by s^{-r}). Now let us establish a simulation algorithm from the density $\tilde{f}$.

To sample from $\tilde{f}$, we first toss (at most two) coins to determine on which of the following (possibly degenerate) intervals lies the sample: on $[0, z_*]$, $[z_*, z]$ or $[z, 1]$, where $z_* = z \wedge (1/2)$. The time cost of this step is constant, see Remark 2.3.

a) Case $D = 2$ (the sample lies in $[z, 1]$). Recall that by Lemma 4.3 we have $\sigma_\alpha'', \sigma_\alpha', \sigma_\alpha > 0$ on $(0, 1)$ and that z is the unique solution to $\sigma_\alpha(z) = \alpha s^r$. By definition of $\tilde{f}$, the following equalities

$$(\log \tilde{f})' = (\alpha/\sigma_\alpha - s^{-r})\sigma_\alpha' \quad \& \quad (\log \tilde{f})'' = (\alpha/\sigma_\alpha - s^{-r})\sigma_\alpha'' - (\sigma_\alpha'/\sigma_\alpha)^2 < 0$$

hold on the interval $(z, 1)$. Since σ_α is strictly increasing and $\sigma_\alpha'' > 0$, $\tilde{f}|_{[z,1]}$ log-concave on $[z, 1]$ with mode at z. Thus we may apply LC-Alg to obtain a sample in the interval $[z, 1]$.

Again, the computational cost of LC-Alg has two parts: (I) the cost to finding the value of a_1 and (II) the expected cost of the accept-reject step. By [11, §4], the expected cost of (II) is bounded above by 5.

To analyse the cost of finding a_1 in LC-Alg, our idea is to find $y \in (z, (z+1)/2)$ such that $\tilde{f}(y)/\tilde{f}(z) \geqslant 1/4$ since $(z+1)/2$ is the first number we check in the binary search. And then we only need to estimate the upperbound of $\log(1/(y - z))$. Recall that $\tilde{f}(x) := \sigma_\alpha(x)^\alpha \exp(-\sigma_\alpha(x)s^{-r})$ and $\tilde{f}(z) = \alpha s^r$. In fact, for $y \in (z, 1)$ satisfying $\sigma_\alpha(y) = (\alpha + 2\log 2)s^r$, we have

$$\frac{\tilde{f}(y)}{\tilde{f}(z)} = \frac{(\alpha + 2\log 2)s^r \exp(-(\alpha + 2\log 2))}{\alpha s^r \exp(-\alpha)} \geqslant \frac{1}{4}.$$

By Lemma 4.10 and the fact that σ_α is convex, we have

$$(r + 1)\alpha^r(1 - \alpha)(1 - (1 + z)/2)^{-(r+2)} = \left(\alpha^r(1 - \alpha)(1 - x)^{-(r+1)}\right)'\Big|_{x = \frac{1+z}{2}}$$

$$\geqslant \sigma_\alpha'\left(\frac{z+1}{2}\right) \geqslant \frac{\sigma_\alpha(y) - \sigma_\alpha(z)}{y - z} = \frac{(\log 4)s^r}{y - z}.$$

Together with (4.18) in Lemma 4.9, we have

$$\frac{1}{y - z} \leqslant \frac{2^{r+2}\alpha^r}{(\log 4)s^r(1 - z)^{r+2}} \leqslant \frac{2^{(2-\alpha)/(1-\alpha)}\alpha^{\alpha/(1-\alpha)}}{(\log 4)s^{\alpha/(1-\alpha)}} \frac{4^{(2-\alpha)/(1-\alpha)}\alpha^{2-\alpha}s^{\alpha(2-\alpha)/(1-\alpha)}}{\sin(\alpha\pi)^{(2-\alpha)/(1-\alpha)}}$$

$$= s^\alpha \frac{8^{(2-\alpha)/(1-\alpha)}\alpha^{\alpha/(1-\alpha)+2-\alpha}}{\sin(\alpha\pi)^{(2-\alpha)/(1-\alpha)}}.$$

Therefore the steps of binary search in Devroye's algorithm is bounded by

$$1 + \log(1/(y - z)) \leqslant \mathcal{O}(1 + \alpha \log^+(s) + (1 - \alpha)^{-1}|\log(1 - \alpha)|).$$

b) Case $D = 0$ (the sample lies in $[0, z_*]$, where $z_* = z \wedge (1/2)$). We propose a sample Y_s from the distribution function

$$u \mapsto u\sigma_\alpha(u)^\alpha/(z_*\sigma_\alpha(z_*)^\alpha) \tag{4.19}$$

on $[0, z_*]$ via the inversion method. By Proposition 4.6, the computation cost of inversion is bounded above by $\mathcal{O}(|\log(1 - \alpha)| + \log N)$. Since Y_s has a strictly positive (by Lemma 4.3) density, proportional to $u \mapsto \sigma_\alpha(u)^\alpha + \alpha u\sigma_\alpha'(u)\sigma_\alpha(u)^{\alpha-1}$, we accept the sample Y_s with probability $h(Y_s)$, where

$$1 \geqslant h_b(u) := \frac{\sigma_\alpha(u)^\alpha e^{-\sigma_\alpha(u)s^{-r}}}{\sigma_\alpha(u)^\alpha + \alpha u\sigma_\alpha'(u)\sigma_\alpha(u)^{\alpha-1}} = \frac{e^{-\sigma_\alpha(u)s^{-r}}}{1 + \alpha u\sigma_\alpha'(u)/\sigma_\alpha(u)} \geqslant \frac{e^{-\alpha}}{1 + (\alpha/2)(\log \sigma_\alpha)'(\frac{1}{2})} \tag{4.20}$$

$$= e^{-\alpha}\left(1 + (r\pi/2)\left(\alpha^2 \cot\left(\alpha\frac{\pi}{2}\right) + (1 - \alpha)^2 \cot\left((1 - \alpha)\frac{\pi}{2}\right)\right)\right)^{-1} \quad \text{for } u \in [0, z_*].$$

The first inequality in (4.20) holds since both $\sigma'_\alpha > 0$ and $\sigma_\alpha > 0$ on $(0,1)$. The second inequality follows from Lemma 4.3 because $(\log \sigma_\alpha)'$ is increasing on $(0,1)$ and $z_* \leq 1/2$. A direct calculation using the definition of σ_α in (4.2) yields the final expression in (4.20), which is decreasing in α and approaching its infimum $e^{-1}(2 + \pi^2/4)^{-1} \approx 0.0823$ as $\alpha \to 1$.

c) The case $D = 1$ (where $z \in (1/2, 1)$ and the sample lies in $[1/2, z]$), is split in two subcases:

(i) Case $\alpha \leqslant 1/2$. We propose from the density proportional to

$$u \mapsto \mathbb{1}_{\{1/2 < u \leqslant z\}}(1 - u)^{-r}(\sin(\pi(1 - \alpha)) + \pi\alpha(1 - \alpha)\cos(\pi\alpha)(1 - u)), \tag{4.21}$$

When $\alpha = 1/2$, the density is proportional to $u \mapsto \mathbb{1}_{\{1/2 < u \leqslant z\}}(1 - u)^{-r}$. We can sample from this via inversion directly and the time cost is a constant. When $\alpha \in (0, 1/2)$, we can sample from the density via inversion via Newton–Raphson method: the CDF of our target density equals

$$x \mapsto (F(x) - F(1/2)/(F(z) - F(1/2)) \qquad x \in [1/2, z] \tag{4.22}$$

with F defined in (4.16). So to get a sample from the target density we only need to solve $F(x) = U$ where $U \sim \mathrm{U}(F(1/2), F(z))$. By Proposition 4.7 and Lemma 4.9, the time cost of inversion is bounded above by

$$\mathcal{O}(\log(1/1 - z)) + \log N) \leqslant \mathcal{O}\left(\log\left(\frac{4\alpha^{1-\alpha}s^\alpha}{\sin(\alpha\pi)}\right) + \log N\right) \leqslant \mathcal{O}(\alpha \log_2^+(s) + \log N).$$

Then we accept the sample U with probability $h_{c,1}(U)$, where $h_{c,1}$ is given by

$$1 \geqslant h_{c,1}(u) := \frac{\sigma_\alpha(u)^\alpha e^{-\sigma_\alpha(u)s^{-r}}\sin(\pi(1 - \alpha))^{1-\alpha}2^r(1 - u)^r}{\sin(\pi(1 - \alpha)) + \pi\alpha(1 - \alpha)\cos(\pi\alpha)(1 - u)} \geqslant \frac{4}{3\pi\sqrt{e}} \approx 0.25742. \tag{4.23}$$

We now prove the second inequality in (4.23). By concavity, for $u \in [1/2, 1]$, we have

$$\sin(\pi(1 - \alpha)u)^\alpha = \sin(\pi(1 - \alpha))^\alpha - \pi\alpha(1 - \alpha)\int_u^1 \sin(\pi(1 - \alpha)x)^{\alpha-1}\cos(\pi(1 - \alpha)x)\mathrm{d}x$$

$$\leqslant \sin(\pi(1 - \alpha))^{\alpha-1}\big(\sin(\pi(1 - \alpha)) + \pi\alpha(1 - \alpha)\cos(\pi\alpha)(1 - u)\big).$$

Moreover, again by concavity, the quotient satisfies

$$1 \geqslant \frac{\sin(\pi(1 - \alpha)u)^\alpha \sin(\pi(1 - \alpha))^{1-\alpha}}{\sin(\pi(1 - \alpha)) + \pi\alpha(1 - \alpha)\cos(\pi\alpha)(1 - u)} \geqslant \frac{\sin(\pi(1 - \alpha)/2)^\alpha \sin(\pi(1 - \alpha))^{1-\alpha}}{\sin(\pi(1 - \alpha)) + \pi\alpha(1 - \alpha)\cos(\pi\alpha)/2} \geqslant \frac{2}{3}.$$

Using the elementary inequalities $2(1 - x) \leqslant \sin(\pi x) \leqslant \min\{1, \pi(1 - x)\}$ for $x \in (1/2, 1)$, we have

$$1 \geqslant h_{c,1}(u) = \frac{\sigma_\alpha(u)^\alpha e^{-\sigma_\alpha(u)s^{-r}}2^r(1 - u)^r}{\sin(\pi(1 - \alpha)u)^\alpha} \geqslant e^{-\alpha}\frac{2^{r(\alpha+1)}}{\pi^r}\frac{2}{3}(1 - \alpha u)^{\alpha r} \geqslant e^{-\alpha}\left(\frac{2}{\pi}\right)^r\frac{2}{3} \geqslant \frac{4}{3\pi\sqrt{e}}.$$

(ii) Case $\alpha > 1/2$. We propose from the distribution function

$$u \mapsto (\rho(u)^{r-1} - \rho(1/2)^{r-1})/(\rho(z)^{r-1} - \rho(1/2)^{r-1}), \tag{4.24}$$

which can be sampled from via the inversion method. Note that now the solution $x_* \in (1/2, z)$ in the inversion method, by Proposition 4.5 and Lemma 4.9, the time cost of inversion method is bounded above by

$$\mathcal{O}\big(\log(1/(1-z)) + |\log(1-\alpha)| + \log N\big) \leqslant \mathcal{O}\Big(\log\Big(\frac{4\alpha^{1-\alpha}s^\alpha}{\sin(\alpha\pi)}\Big) + |\log(1-\alpha)| + \log N\Big)$$
$$= \mathcal{O}\big(\alpha\log_2^+(s) + |\log(1-\alpha)| + \log N\big).$$

Then accept the proposal with probability

$$h_{c,2}(u) := C_\alpha \frac{\sigma_\alpha(u)^\alpha e^{-\sigma_\alpha(u)s^{-r}}}{\rho'(u)\rho(u)^{r-2}} = C_\alpha e^{-\sigma_\alpha(u)s^{-r}}\frac{\rho(u)^2}{\rho'(u)}$$
$$= C_\alpha \frac{e^{-\sigma_\alpha(u)s^{-r}}\sin(\alpha\pi u)^\alpha \sin((1-\alpha)\pi u)^{1-\alpha}/\sin(\pi u)}{\pi(\alpha^2\cot(\alpha\pi u) + (1-\alpha)^2\cot((1-\alpha)\pi u) - \cot(\pi u))}, \tag{4.25}$$

where

$$C_\alpha := \frac{\rho'(\frac{1}{2})}{\rho(\frac{1}{2})^2} = \pi\frac{\alpha^2\cot(\alpha\frac{\pi}{2}) + (1-\alpha)^2\cot((1-\alpha)\frac{\pi}{2})}{\sin(\alpha\frac{\pi}{2})^\alpha\sin((1-\alpha)\frac{\pi}{2})^{1-\alpha}}$$
$$\geqslant (1-\alpha)(\alpha^2\pi + 2\sin(\alpha\frac{\pi}{2})) \geqslant (1-\alpha)(\frac{\pi}{4} + \sqrt{2}).$$

By Lemma 4.8, $\rho(u)^2/\rho'(u) = -(\frac{\mathrm{d}}{\mathrm{d}u}(\rho(u)^{-1}))^{-1}$ is decreasing, so $h_{c,2}(u) \leqslant 1$ and

$$\frac{h_{c,2}(u)}{C_\alpha} \geqslant e^{-\alpha}\frac{\rho(1)^2}{\rho'(1)} = e^{-\alpha}\frac{\sin(\alpha\pi)^\alpha\sin((1-\alpha)\pi)^{1-\alpha}}{\pi^2} \geqslant e^{-\alpha}\frac{1-\alpha}{\pi}.$$

This shows that $h_{c,2}(u)/(1-\alpha)^2$ is uniformly bounded away from 0 for $\alpha \in (1/2, 1)$ and $u \in [1/2, 1]$.

All in all, the expected running time is bounded above by

$$\kappa_{(2)}\Big[1/(1-\alpha)^2 + 1/\log^+(s^{\alpha/(1-\alpha)} - \alpha^{(1-2\alpha)/(1-\alpha)}(1-\alpha)) + \alpha\log^+(s) + |\log(\alpha)| + \log N\Big],$$

where the constant $\kappa_{(2)}$ depends on neither $s \in (0, \infty)$ nor $\alpha \in (0, 1)$. $\qquad\square$

4.3.4 Proof of Proposition 2.4.

In this subsection we prove Proposition 2.4 using Propositions 2.5, 2.6 and 4.2. Let t and w be as in the statement of Proposition 2.4 and recall that $\theta > 0$ is the "intensity" in the Lévy measure (4.1) of the stable subordinator S. Recall also that $s = (\theta t)^{-1/\alpha}w$. As explained in the beginning of Section 4.3, if we sample (ζ_s, Y_s) with density ψ_s given in (4.8), then the random variable $(\theta t)^{1/\alpha}\zeta_s^{-1/r}$ follows the law of $S_{t-}|\{\tau_b = t, b(\tau_b) = w, \Delta_S(\tau_b) > 0\}$. By the inequalities in (4.10) of Proposition 4.2, SU-Alg samples from $\widetilde{\psi}_s$ (defined in Proposition 4.2) and performs an accept-reject step to obtain a sample (ζ_s, Y_s) from the density ψ_s. This implies that SU-Alg indeed samples from the law $\mathrm{SU}_\alpha(t, w)$.

It remains to prove that the expected running time of SU-Alg is bounded as stated. As explained in Remark 2.1 above, the computation of p' is assumed to take a constant amount of time. By (4.10), the acceptance probability in line 9 of SU-Alg equals $c_\psi\psi_s(\zeta_s, Y_s)/\widetilde{\psi}_s(\zeta_s, Y_s)$ and is bounded below by $(1-\alpha)/2$, uniformly in s (and all other parameters except α). Thus the accept-reject step in SU-Alg on average takes no more than $2/(1-\alpha)$ steps. Since $\widetilde{\psi}_s$ is mixture of densities $\widetilde{\psi}_s^{(1)}$ and $\widetilde{\psi}_s^{(2)}$ (see Proposition 2.4), we sample a Bernoulli variable taking value 1 with probability p. If we get 1, then sample from $\widetilde{\psi}_s^{(1)}$ and otherwise from $\widetilde{\psi}_s^{(2)}$. Thus, the computational complexities of Algorithms $\psi^{(1)}$-Alg and $\psi^{(2)}$-Alg, given in Propositions 2.5 and 2.6), imply the bound in Proposition 2.4. $\qquad\square$

4.4 Proofs of Theorem 2.3

We begin by recalling a well-known property of the moments (or Mellin transform) of stable random variables, see [25, Example 25.10]. For $\eta \in (-\infty, \alpha)$ and $t > 0$, we have

$$\mathbb{E}[S_t^\eta] = (t\theta)^{\eta/\alpha}\Gamma(1 - \eta/\alpha)/\Gamma(1 - \eta) < \infty. \tag{4.26}$$

Since $s = (\theta t)^{-1/\alpha}w$ and $\tau_b = B^{-1}(S_1)$ by Proposition 4.1, where S_1 is the value of the stable subordinator at time 1 and B^{-1} is the inverse of the decreasing function $t \mapsto t^{-1/\alpha}b(t)$, we have

$$(\theta\tau_b)^{-1/\alpha}b(\tau_b) = \theta^{-1/\alpha}S_1 \stackrel{d}{=} S_{1/\theta}$$

and hence $s \stackrel{d}{=} S_{1/\theta}$. Note that the density of $S_{1/\theta}$ is φ_α, which does not depend on the value of θ (see the representation of density of S_t in (4.3)).

Lemma 4.11. *The following limit holds:* $(1 - \alpha)\sup_{x \in \mathbb{R}_+} \varphi_\alpha(x) \to 1/\pi^2$ *as* $\alpha \to 1$.

Proof. Note from (4.1) and the Lévy–Khintchine formula that the Fourier transform of φ_α is, for any $u \in \mathbb{R}$, given by

$$\mathbb{E}[e^{iuS_{1/\theta}}] = \exp(-(-iu)^\alpha) = \exp\left(-|u|^\alpha e^{-i\mathrm{sgn}(u)\pi\alpha/2}\right),$$

where $\mathrm{sgn}(u) := \mathbb{1}_{(0,\infty)}(u) - \mathbb{1}_{(-\infty,0)}(u)$ is the sign of u. Recall that $\varphi_\alpha(x) = \frac{1}{2\pi}\int_{\mathbb{R}} e^{-iux}\mathbb{E}[e^{iuS_{1/\theta}}]\mathrm{d}u$ for any $x \in \mathbb{R}$. Thus,

$$\sup_{x \in \mathbb{R}_+} \varphi_\alpha(x) \leqslant \frac{1}{2\pi}\int_{\mathbb{R}}\left|\exp\left(-|u|^\alpha e^{-i\mathrm{sgn}(u)\pi\alpha/2}\right)\right|\mathrm{d}u$$

$$= \frac{1}{2\pi}\int_{\mathbb{R}} e^{-|u|^\alpha \cos(\pi\alpha/2)}\mathrm{d}u = \frac{1}{2\pi}\cos(\pi\alpha/2)^{-1/\alpha}\int_{\mathbb{R}} e^{-|u|^\alpha}\mathrm{d}u,$$

where $\int_{\mathbb{R}} e^{-|u|^\alpha}\mathrm{d}u \to \int_{\mathbb{R}} e^{-|u|}\mathrm{d}u = 2$ as $\alpha \to 1$ by dominated convergence and $\cos(\pi\alpha/2)^{-1/\alpha}(1 - \alpha) \to 2/\pi$ as $\alpha \to 1$. $\square$

Lemma 4.12. *With s defined in line 1 of SU-Alg, we have:*

(a) $\mathbb{E}\left[\log^+\left(1/\left(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}(1 - \alpha)\right)\right)\right] = \mathcal{O}((1 - \alpha)^{-1})$ *as* $\alpha \uparrow 1$;

(b) $\mathbb{E}\left[\log^+\left(1/\left(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}(1 - \alpha)\right)\right)\right] \to 0$ *as* $\alpha \downarrow 0$.

Proof of Lemma 4.12. Denote $A := \alpha^{1-1/r}(1 - \alpha)^{1/r} = \alpha^{2-1/\alpha}(1 - \alpha)^{1/\alpha-1}$.

(a) When $\alpha > 1/2$, note that $\|\varphi_\alpha\|_\infty := \sup_{x \in \mathbb{R}_+} \varphi_\alpha(x) = \mathcal{O}((1 - \alpha)^{-1})$ as $\alpha \uparrow 1$ by Lemma 4.11. Thus,

$$\mathbb{E}\left[\log^+\left(\frac{1}{s^r - \alpha^{r-1}(1 - \alpha)}\right)\right] = -\int_A^{(A^r+1)^{1/r}} \varphi_\alpha(x)\log(x^r - A^r)\mathrm{d}x$$

$$\leqslant -\|\varphi_\alpha\|_\infty \int_A^{(A^r+1)^{1/r}} \log(x^r - A^r)\mathrm{d}x.$$

By substituting $x = (y + A^r)^{1/r}$ we get

$$-\int_A^{(A^r+1)^{1/r}} \log(x^r - A^r)\mathrm{d}x = -\frac{1}{r}\int_0^1 (y + A^r)^{1/r-1}\log(y)\mathrm{d}y \leqslant -\frac{A^{r(1/r-1)}}{r}\int_0^1 \log(y)\mathrm{d}y$$

$$= \alpha^{(2\alpha-1)(1-2\alpha)/(\alpha(1-\alpha))-1}(1 - \alpha)^{(1-2\alpha)/\alpha+1} \to 1, \quad \text{as } \alpha \uparrow 1.$$

Hence we have $\mathbb{E}[\log^+(1/(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}(1 - \alpha)))] = \mathcal{O}((1 - \alpha)^{-1})$ as $\alpha \uparrow 1$.

(b) Note that the function $y \mapsto y\mathrm{e}^{-y}$ attains its maximum when $y = 1$, implying $y\mathrm{e}^{-y} \leq 1/\mathrm{e}$ for all $y \in \mathbb{R}_+$. Thus

$$\varphi_\alpha(x) = r\int_0^1 \sigma_\alpha(u)x^{-r-1}\mathrm{e}^{-\sigma_\alpha(u)x^{-r}}\mathrm{d}u \leq r\int_0^1 x^{-1}\mathrm{e}^{-1}\mathrm{d}u = \frac{r}{\mathrm{e}x}.$$

The expectation in part (b) of the lemma can now be bounded as follows:

$$\mathbb{E}\left[\log^+\left(\frac{1}{s^r - \alpha^{r-1}(1-\alpha)}\right)\right] = -\int_A^{(A^r+1)^{1/r}}\varphi_\alpha(x)\log(x^r - A^r)\mathrm{d}x \leq -\frac{r}{\mathrm{e}}\int_A^{(A^r+1)^{1/r}}\log(x^r - A^r)\frac{\mathrm{d}x}{x}.$$

By substituting $x = (y + A^r)^{1/r}$ we get

$$-\int_0^1 (y + A^r)^{-1/r+1/r-1}\log(y)\mathrm{d}y \leq -A^{-r}\int_0^1 \log(y)\mathrm{d}y = (1-\alpha)^{-1}\alpha^{1-\alpha/(1-\alpha)} \to 0 \quad \text{as } \alpha \downarrow 0. \quad \square$$

We can now give a proof Theorem 2.3 on the expected running time of SFP-Alg.

Proof of Theorem 2.3. Case $t_* = \infty$. SFP-Alg samples $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}$ by Proposition 4.1. By Proposition 2.4, to find an upper bound on the average running time, we only need to take expectation of the average running time of SU-Alg in s. In other words, upper bound $\mathbb{E}[\alpha \log_2^+(s)]$, $\mathbb{E}[\alpha \log_2^+(s^{-1})]$ and $\mathbb{E}[\log^+(1/(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}(1 - \alpha)))]$.

By (4.26) (with η equal to either $\alpha/2$ or $-\alpha/2$), the inequality $\alpha \log(x) \leq 2x^{\alpha/2}$ for all $x > 0$ and the fact that $\inf_{x \in (0,2)} \Gamma(x) > 1/2$, we have

$$\mathbb{E}[\alpha \log^+(s)] \leq 2\mathbb{E}[S_{1/\theta}^{\alpha/2}\mathbb{1}\{S_{1/\theta} > 1\}] \leq 2\mathbb{E}[S_{1/\theta}^{\alpha/2}] \leq 4\Gamma(1/2),$$

$$\mathbb{E}[\alpha \log^+(s^{-1})] \leq 2\mathbb{E}[S_{1/\theta}^{-\alpha/2}\mathbb{1}\{S_{1/\theta} < 1\}] \leq 2\mathbb{E}[S_{1/\theta}^{-\alpha/2}] \leq 4\Gamma(3/2).$$

Together with Lemma 4.12, the average running time of SFP-Alg is bounded above by

$$\kappa_0\big((1 - \alpha)^{-3} + |\log \alpha| + \log N\big)$$

where the constant κ_0 does not depend on α.

To show the running time of SFP-Alg has exponential moments, we only need to verify that the running times of Algorithms $\psi^{(1)}$-Alg and $\psi^{(2)}$-Alg, with random parameter $s \overset{d}{=} S_{1/\theta}$, have exponential moments. By Propositions 2.5 and 2.6, we only need to show that $\log^+(s^{-1})$, $\log^+(s)$ and $\log^+(s^r - \alpha^{r-1}(1 - \alpha))^{-1})$ have exponential moments. Note from (4.26) that $\mathbb{E}[\exp(\eta \log^+(s^{-1}))] = \mathbb{E}[s^{-\eta}\mathbb{1}\{s < 1\}] < \infty$ for any $\eta \geq 0$, $\mathbb{E}[\exp(\eta \log^+(s))] = \mathbb{E}[s^\eta\mathbb{1}\{s > 1\}] < \infty$ for any $\eta < \alpha$, and

$$\mathbb{E}\big[\exp\big(\eta\big(\log^+(1/(s^{\alpha/(1-\alpha)} - \alpha^{(2\alpha-1)/(1-\alpha)}(1 - \alpha)))\big)\big)\big] = \mathbb{E}\left[\frac{1}{(s^r - A^r)^\eta}\mathbb{1}\{A < s < (A^r + 1)^{1/r}\}\right]$$

$$= \int_A^{(A^r+1)^{1/r}}\frac{\varphi_\alpha(x)}{(x^r - A^r)^\eta}\mathrm{d}x \leq \int_A^{(A^r+1)^{1/r}}\frac{\|\varphi_\alpha\|_\infty}{(x^r - A^r)^\eta}\mathrm{d}x = \|\varphi_\alpha\|_\infty\int_0^1 \frac{(y + A^r)^{1/r-1}}{y^\eta}\mathrm{d}y < \infty.$$

for any $\eta \in (0, 1)$, where we recall that $\|\varphi_\alpha\|_\infty = \sup_{x \in \mathbb{R}_+} \varphi_\alpha(x)$. Thus, these steps indeed have running times with exponential moments, completing the proof.

Case $t_* < \infty$. Again by Proposition 4.1, SFP-Alg samples $(\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}$ conditional on $\tau_b \leq t_*$ since we keep sampling τ_b until $\tau_b \leq t_*$. Indeed, note that $\tau_b \leq t_*$ is equivalent to $V_1 \overset{d}{=} S_1$ being larger or equal to $B(t_*)$, where B is specified in Proposition 4.1, and has acceptance probability equal to $\mathbb{P}[\tau_b \leq t_*]$. The other steps are the same as in the case $t_* = \infty$.

The running time of SFP-Alg in the case $t_* < \infty$ consists of two parts: (I) the naive sampling of $V_1 \overset{d}{=} S_1$ until $V_1 \geqslant B(t_*)$ in Line 1 of SFP-Alg and (II) the running time of SU-Alg with random input $t = \tau_b$ in Line 7 of SFP-Alg, where τ_b is conditioned to be smaller or equal to t_*. Let $T_\alpha(t)$ be the expected runtime of SU-Alg with parameter t. Then the running time of SFP-Alg in the case $t_* < \infty$ is bounded by

$$\frac{1}{\mathbb{P}[\tau_b \leqslant t_*]} + \mathbb{E}[T_\alpha(\tau_b) | \tau_b \leqslant t_*] = \frac{1 + \mathbb{E}[T_\alpha(\tau_b)\mathbb{1}_{\{\tau_b \leqslant t_*\}}]}{\mathbb{P}[\tau_b \leqslant t_*]} \leqslant \frac{1 + \mathbb{E}[T_\alpha(\tau_b)]}{\mathbb{P}[S_1 \geqslant B(t_*)]},$$

where $\mathbb{E}[T_\alpha(\tau_b)] \leqslant \kappa_{SFP}((1-\alpha)^{-3} + |\log \alpha| + \log N)$ by Theorem 2.3. It remains to lower bound the probability $\mathbb{P}[S_1 \geqslant B(t_*)]$.

Pick $\lambda > 0$ and note that

$$\mathbb{P}[S_{t_*} < b(t_*)] \leqslant \mathbb{E}[\mathrm{e}^{\lambda b(t_*) - \lambda S_{t_*}}\mathbb{1}\{S_{t_*} < b(t_*)\}] \leqslant \mathbb{E}[\mathrm{e}^{\lambda b(t_*) - \lambda S_{t_*}}] = \mathrm{e}^{\lambda b(t_*) - \lambda^\alpha \theta t_*}.$$

Suppose $\lambda^\alpha \theta t_* > \lambda b(t_*)$, then the elementary inequality $1 - \mathrm{e}^{-x} \geqslant x\mathrm{e}^{-x}$ for $x > 0$ gives

$$\mathbb{P}[S_{t_*} \geqslant b(t_*)] = 1 - \mathbb{P}[S_{t_*} < b(t_*)] \geqslant 1 - \mathrm{e}^{\lambda b(t_*) - \lambda^\alpha \theta t_*} \geqslant (\lambda^\alpha \theta t_* - \lambda b(t_*))\mathrm{e}^{\lambda b(t_*) - \lambda^\alpha \theta t_*}.$$

Since $x\mathrm{e}^{-x}$ attains its maximum e^{-1} when $x = 1$, it is optimal to let λ be the 'solution' to $\lambda^\alpha \theta t_* - \lambda b(t_*) = 1$. This solution, however, need not exist if $\theta t_*/b(t_*)^\alpha$ is small. Note that $\lambda \mapsto \lambda^\alpha \theta t_* - \lambda b(t_*)$ is increasing and continuous on $[0, (\alpha \theta t_*/b(t_*))^{r+1}]$ and decreasing thereafter, with maximal value $\alpha^r(1-\alpha)(\theta t_*)^{r+1}b(t_*)^r$. Thus, we may take λ to be the solution of

$$\lambda^\alpha \theta t_* - \lambda b(t_*) = \min\{1, \alpha^r(1-\alpha)(\theta t_*)^{r+1}b(t_*)^r\}.$$

This leads to the lower bound

$$\mathbb{P}[S_{t_*} \geqslant b(t_*)] \geqslant \min\{1, \alpha^r(1-\alpha)(\theta t_*)^{r+1}b(t_*)^r\}\mathrm{e}^{-1},$$

implying the claim and completing the proof. $\qquad\square$

4.5 Proofs of Theorem 2.1 and Corollary 2.2

We begin by introducing a construction of the probability measures $\mathbb{P}_q$ given the base probability measure $\mathbb{P}$ via the Esscher transform. Fix any $\theta > 0$ and $\alpha \in (0,1)$ and let S be a driftless stable subordinator under the probability measure $\mathbb{P}$ with Lévy measure ν. Denote by $(\mathcal{F}_t)_{t \geqslant 0}$ the right-continuous filtration generated by S. Given $q \geqslant 0$, the probability measure $\mathbb{P}_q$ can be constructed via its Radon–Nikodym derivative

$$\left.\frac{\mathrm{d}\mathbb{P}_q}{\mathrm{d}\mathbb{P}}\right|_{\mathcal{F}_t} := M_q(t) := \exp\left(-qS_t + q^\alpha \theta t\right), \quad t \geqslant 0.$$

Indeed, by [Sato, Thm 33.2], S is subordinator under $\mathbb{P}_q$ with Lévy measure $\nu_q(\mathrm{d}x) = \mathrm{e}^{-qx}\nu(\mathrm{d}x)$, making it a tempered stable process as desired.

Recall $\chi_b = (\tau_b, S_{\tau_b-}, S_{\tau_b})$, fix some $t > 0$ and recall that $\{\tau_b \leqslant t\} \in \mathcal{F}_t$. For any measurable $A \subset [0,t] \times [0,\infty)^2$ we have $\{\chi_b \in A\} \in \mathcal{F}_t$ and hence

$$\mathbb{P}_q[\chi_b \in A] = \mathbb{E}[\mathbb{1}_A(\chi_b)M_q(t)] = \mathbb{E}[\mathbb{1}_A(\chi_b)\mathbb{1}_{(qS_t, \infty)}(E)]\mathrm{e}^{q^\alpha \theta t} = \mathbb{P}[\chi_b \in A \,|\, E \geqslant qS_t],$$

where $E \sim \mathrm{Exp}(1)$ is independent of S under $\mathbb{P}$. In particular, we have

$$\mathbb{P}_q[\chi_b \in A \,|\, \tau_b \leqslant t] = \mathbb{P}[\chi_b \in A \,|\, \tau_b \leqslant t, E \geqslant qS_t]. \tag{4.27}$$

Proof of Theorem 2.1. For a fixed time t_*, TSFP-Alg first samples the event $\{\tau_b \leqslant t_*\}$ under $\mathbb{P}$. Indeed, this is done by sampling $s \sim \mathcal{L}(S_{t_*})$ under $\mathbb{P}_q$ and noting that the event $s \leqslant b(t_*)$ is equivalent to the corresponding tempered stable process, of which s is the final value, to not cross the boundary on the time horizon $[0, t_*]$. Thus, if $s \leqslant b(t_*)$, by the Markov property of S, we may simply start a new iteration from the initial time-space point (t_*, s). If instead $s < b(t_*)$, we need to sample $\chi_b | \{\tau_b \leqslant t_*\}$ under $\mathbb{P}_q$. Moreover, in that case, by (4.27), we need only repeatedly sample $\chi_b = (\tau_b, S_{\tau_b-}, S_{\tau_b})$ under $\mathbb{P}$ conditional on $\tau_b \leqslant t_*$ and an independent $E \sim \mathrm{Exp}(1)$, until $E \geqslant qS_{t_*}$. For that, SFP-Alg samples $(t, u, v) \sim \mathcal{L}(\chi_b)$ under $\mathbb{P}$ conditional on $\tau_b \leqslant t_*$. By the strong Markov property, to check if $E \geqslant qS_{t_*}$, we need only sample independently (and conditionally given $\tau_b = t$) $w \sim \mathcal{L}(S_{t_*} - S_t) = \mathcal{L}(S_{t_*-t})$ via S-Alg and set $S_{t_*} = v + w$. This justifies the validity of TSFP-Alg.

Now let us consider the computational complexity of TSFP-Alg. The expected number of iterations in Line 2 of TSFP-Alg is

$$\mathbb{E}_q[\lfloor \tau_b/t_* \rfloor + 1] \leqslant \mathbb{E}_q[\tau_b]/t_* + 1.$$

In each iteration we use TS-Alg with expected complexity uniformly bounded by $c_0 := 4.2154$ according to [24, Thm 3.1]. After that, we sample the triplet conditional on $\tau_b \leqslant t_*$ via SFP-Alg. By Theorem 2.3, the expected running time of SFP-Alg is bounded by

$$\kappa_{SFP}((1-\alpha)^{-3} + |\log \alpha| + \log N)/\mathbb{P}[S_{t_*} \geqslant b(t_*)].$$

The acceptance probability in Line 11 of TSFP-Alg is

$$\mathbb{P}[E \geqslant qS_{t_*} | S_{t_*} \geqslant b(t_*)] = \mathbb{E}[\mathrm{e}^{-qS_{t_*}} \mathbb{1}\{S_{t_*} \geqslant b(t_*)\}]/\mathbb{P}[S_{t_*} \geqslant b(t_*)].$$

All in all, the expected running time of TSFP-Alg is bounded by

$$c_0\left(1 + \frac{\mathbb{E}[\tau_b]}{t_*}\right) + \frac{\kappa_{SFP}((1-\alpha)^{-3} + |\log \alpha| + \log N)}{\mathbb{E}[\mathrm{e}^{-qS_{t_*}} \mathbb{1}\{S_{t_*} \geqslant b(t_*)\}]}. \tag{4.28}$$

It remains to upper bound $\mathbb{E}[\tau_b]$ and lower bound $\mathbb{E}[\mathrm{e}^{-qS_{t_*}} \mathbb{1}\{S_{t_*} \geqslant b(t_*)\}]$.

Recall that

$$\mathbb{E}_q[\mathrm{e}^{-uS_1}] = \exp((q^\alpha - (u+q)^\alpha)\theta), \quad u \geqslant 0.$$

Note that, by concavity, $(x+y)^\alpha - x^\alpha = \int_x^{x+y} \alpha z^{\alpha-1} \mathrm{d}z \geqslant \alpha y(x+y)^{\alpha-1}$. Thus, by Lemma A.2 with $u = 1/b(0)$, we have

$$\mathbb{E}_q[\tau_b] \leqslant \frac{\theta^{-1}\mathrm{e}^{ub(0)}}{(u+q)^\alpha - q^\alpha} = \frac{\theta^{-1}\mathrm{e}}{(q+1/b(0))^\alpha - q^\alpha} \leqslant \frac{\mathrm{e}(q+1/b(0))^{1-\alpha}b(0)}{\alpha\theta} = \frac{\mathrm{e}(1 + 1/(qb(0)))^{1-\alpha}qb(0)}{\alpha q^\alpha \theta}.$$

On the other hand, note that

$$\mathbb{E}[\mathrm{e}^{-qS_{t_*}} \mathbb{1}\{S_{t_*} < b(t_*)\}] \leqslant \mathbb{E}[\mathrm{e}^{-2qS_{t_*}+qb(t_*)} \mathbb{1}\{S_{t_*} < b(t_*)\}]$$
$$\leqslant \mathbb{E}[\mathrm{e}^{-2qS_{t_*}+qb(t_*)}] = \mathrm{e}^{-2^\alpha q^\alpha \theta t_* + qb(t_*)} \leqslant \mathrm{e}^{-2^\alpha q^\alpha \theta t_* + qb(0)}.$$

Hence, if t_* satisfies $(2^\alpha - 1)q^\alpha \theta t_* > qb(0)$, then the elementary inequality $1 - \mathrm{e}^{-x} \geqslant x\mathrm{e}^{-x}$ for $x \geqslant 0$ gives

$$\mathbb{E}[\mathrm{e}^{-qS_{t_*}} \mathbb{1}\{S_{t_*} \geqslant b(t_*)\}] = \mathrm{e}^{-q^\alpha \theta t_*} - \mathbb{E}[\mathrm{e}^{-qS_{t_*}} \mathbb{1}\{S_{t_*} < b(t_*)\}]$$
$$\geqslant \mathrm{e}^{-q^\alpha \theta t_*}(1 - \mathrm{e}^{-(2^\alpha-1)q^\alpha \theta t_* + 2qb(0)}) \geqslant ((2^\alpha - 1)q^\alpha \theta t_* - 2qb(0))\mathrm{e}^{-2^\alpha q^\alpha \theta t_* + 2qb(0)}.$$

Denote $T_1 := qb(0)/(2^\alpha - 1)$, $T_2 := qb(0)$ and set $t_* = (2qb(0)+1-2^{-\alpha})/((2^\alpha-1)q^\alpha\theta) = (2T_1+2^{-\alpha})/(q^\alpha\theta)$ and note that $2^\alpha - 1 \in [\alpha\log 2, \alpha]$. Then the expected running time of TSFP-Alg is bounded by

$$
\begin{aligned}
&c_0\left(1 + \mathrm{e}\frac{(1+1/(qb(0)))^{1-\alpha}qb(0)}{\alpha q^\alpha \theta t_*}\right) + \frac{1 + \kappa_{SFP}((1-\alpha)^{-3} + |\log\alpha| + \log N)}{((2^\alpha-1)q^\alpha\theta t_* - 2qb(0))\mathrm{e}^{-2^\alpha q^\alpha\theta t_* + 2qb(0)}} \\
={}&c_0\left(1 + \mathrm{e}\frac{(1+1/T_2)^{1-\alpha}T_2}{\alpha(2T_1+2^{-\alpha})}\right) + \frac{1 + \kappa_{SFP}((1-\alpha)^{-3} + |\log\alpha| + \log N)}{((2^\alpha-1)(2T_1+2^{-\alpha}) - 2T_2)\mathrm{e}^{-2^\alpha(2T_1+2^{-\alpha})+2T_2}} \\
={}&c_0\left(1 + \mathrm{e}\frac{(1+1/T_2)^{1-\alpha}T_2}{\alpha(2(2^\alpha-1)T_2+2^{-\alpha})}\right) + \mathrm{e}^{1+2T_1}\frac{1 + \kappa_{SFP}((1-\alpha)^{-3} + |\log\alpha| + \log N)}{1 - 2^{-\alpha}} \\
\leqslant{}&c_0\left(1 + \frac{\mathrm{e}}{\log 2}\mathbb{1}_{\{T_2 \geqslant 1\}} + \frac{4\mathrm{e}}{\alpha}\mathbb{1}_{\{T_2 < 1\}}\right) + 2^\alpha\mathrm{e}^{1+2T_1}\frac{1 + \kappa_{SFP}((1-\alpha)^{-3} + |\log\alpha| + \log N)}{\alpha\log 2} \\
\leqslant{}&\kappa_{TSFP}\mathrm{e}^{2T_1}((1-\alpha)^{-3} + |\log\alpha| + \log N)/\alpha,
\end{aligned}
$$

where κ_{TSFP} is a constant that depends neither on $\alpha \in (0,1)$, $\theta \in (0,\infty)$ nor $q \in (0,\infty)$. $\qquad\square$

Proof of Corollary 2.2. TSFFP-Alg applies TSFP-Alg at most $1 + \lfloor b(0)/R \rfloor$ times. In each application, the expected running time is bounded by

$$
\kappa_{TSFP}\mathrm{e}^{2qR/(2^\alpha-1)}((1-\alpha)^{-3} + |\log\alpha| + \log N)/\alpha.
$$

With our choice of $R = (2^\alpha - 1)/(2q)$, the average running time of TSFFP-Alg is bounded by

$$
\kappa_{TSFP}\mathrm{e}(1 + 2qb(0)/(2^\alpha - 1))((1-\alpha)^{-3} + |\log\alpha| + \log N)/\alpha.
$$

Since $\alpha\log 2 \leqslant 2^\alpha - 1 \leqslant \alpha$, the claim follows. $\qquad\square$

Acknowledgements

References

[1] Sebastian Ament and Michael O'Neil. Accurate and efficient numerical calculation of stable densities via optimized quadrature and asymptotics. *Statistics and Computing*, 28(1):171–185, 2018.

[2] Leif Andersen and Alexander Lipton. Asymptotics for exponential Lévy processes and their volatility smile: survey and new results. *Int. J. Theor. Appl. Finance*, 16(1):1350001, 98, 2013.

[3] Søren Asmussen. *Applied probability and queues*, volume 51 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, second edition, 2003. Stochastic Modelling and Applied Probability.

[4] Søren Asmussen and Peter W. Glynn. *Stochastic simulation: algorithms and analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer, New York, 2007.

[5] Søren Asmussen and Hansjörg Albrecher. *Ruin Probabilities*. WORLD SCIENTIFIC, 2nd edition, 2010.

[6] Jean Bertoin. *Lévy processes*, volume 121 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 1996.

[7] Loïc Chaumont and Thomas Pellas. Creeping of Lévy processes through curves, 2022. Arxiv:2205.06865.

[8] Zhiyi Chi. On exact sampling of the first passage event of a Lévy process with infinite Lévy measure and bounded variation. *Stochastic Processes and their Applications*, 126(4):1124–1144, 2016.

[9] Angelos Dassios, Jia Wei Lim, and Yan Qu. Exact simulation of a truncated Lévy subordinator. *ACM Trans. Model. Comput. Simul.*, 30(3):Art. 17, 26, 2020.

[10] Luc Devroye. Random variate generation for exponentially and polynomially tilted stable distributions. *ACM Trans. Model. Comput. Simul.*, 19(4), Nov 2009.

[11] Luc Devroye. A note on generating random variables with log-concave densities. *Statistics & Probability Letters*, 82(5):1035–1039, 2012.

[12] R. M. Dudley. Distances of probability measures and random variables. *Ann. Math. Statist.*, 39:1563–1572, 1968.

[13] Laurent Fousse. Accurate multiple-precision gauss-legendre quadrature. In *18th IEEE Symposium on Computer Arithmetic (ARITH '07)*, pages 150–160, 2007.

[14] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 2004. Stochastic Modelling and Applied Probability.

[15] Jorge Ignacio González Cázares, Feng Lin, and Aleksandar Mijatović. Github repositories FirstPassageTemperedStable.jl and FirstPassageTemperedStable.py in `github.com/jorgeignaciogc/FirstPassageTemperedStable.jl` and `github.com/jorgeignaciogc/FirstPassageTemperedStable.py`, March 2023.

[16] Jorge Ignacio González Cázares, Feng Lin, and Aleksandar Mijatović. Presentation on "Fast exact simulation of the first passed of a tempered stable subordinator across a non-increasing funciton". YouTube presentation about the algorithm `https://youtu.be/dO-cQeABHdM`, 2023. For Monte Carlo applications see `https://youtu.be/e38c1LmrzfA` on Prob-AM YouTube channel.

[17] Jorge Ignacio González Cázares and Aleksandar Mijatović. Convex minorants and the fluctuation theory of Lévy processes. *ALEA Lat. Am. J. Probab. Math. Stat.*, 19(1):983–999, 2022.

[18] M. E. Hernández-Hernández, Vassili N. Kolokoltsov, and L. Toniazzi. Generalised fractional evolution equations of caputo type. *Chaos, Solitons & Fractals*, 102:184–196, 2017.

[19] Wolfgang Hörmann, Josef Leydold, and Gerhard Derflinger. *Automatic nonuniform random variate generation*. Statistics and Computing. Springer-Verlag, Berlin, 2004.

[20] Kyoung-Kuk Kim and Sojung Kim. Simulation of tempered stable Lévy bridges and its applications. *Oper. Res.*, 64(2):495–509, 2016.

[21] Anatoly N. Kochubei. General fractional calculus, evolution equations, and renewal processes. *Integral Equations Operator Theory*, 71(4):583–600, 2011.

[22] Vassili Kolokoltsov, Feng Lin, and Aleksandar Mijatović. Monte Carlo estimation of the solution of fractional partial differential equations. *Fract. Calc. Appl. Anal.*, 24(1):278–306, 2021.

[23] John P Nolan. Numerical calculation of stable densities and distribution functions. *Communications in statistics. Stochastic models*, 13(4):759–774, 1997.

[24] Yan Qu, Angelos Dassios, and Hongbiao Zhao. Random variate generation for exponential and gamma tilted stable distributions. *ACM Trans. Model. Comput. Simul.*, 31(4), Jul 2021.

[25] Ken-Iti Sato. *Lévy processes and infinitely divisible distributions*. Cambridge university press, 1999.

[26] Vladimir V Uchaikin and Vladimir M Zolotarev. Chance and stability. In *Chance and Stability*. de Gruyter, 2011.

A   Some results for general subordinators

A.1   Joint law of $(\tau_b, S_{\tau_b-}, \Delta_S(\tau_b))$ for general subordinators

Theorem A.1 below and its proof are taken from [8], see also [7]. This appendix is included for completeness, because Theorem A.1 underpins our key SU-Alg.

Let $S = (S_t)_{t \in \mathbb{R}_+}$ be a subordinator with Lévy measure ν on $(0, \infty)$ with infinite mass and a non-negative drift. Assume that the density of S_t exists for $t > 0$ and is continuous as a map $(t, x) \mapsto g_t(x)$ on $(0, \infty) \times \mathbb{R}_+$. Kallenberg's condition, given by the requirement $\int_\mathbb{R} \mathbb{E}[e^{iuS_t}]du < \infty$ for all $t > 0$, is easily seen to imply the continuity of the density of S_t. As above, let $b : \mathbb{R}_+ \to \mathbb{R}_+$ be a non-increasing absolutely continuous function with $b(0) > 0$ and $T_b = \inf\{t > 0 : b(t) = 0\} \in (0, \infty]$ (with convention $\inf \emptyset := \infty$). Note that, without loss of generality, we may assume that the drift $\mu \geq 0$ of S is zero, since otherwise we may consider $\tilde{b}(t) := b(t) - t\mu$ without modifying the first-passage event in an essential way. For any $x \in \mathbb{R}$, let $\delta(\mathrm{d}u - x)$ denote the Dirac delta measure concentrated at x.

Theorem A.1. *For a subordinator S satisfying assumptions in the previous paragraph and $(t, u, v) \in (0, T_b) \times [0, \infty) \times (0, \infty)$ we have:*

$$\mathbb{P}[\tau_b \in \mathrm{d}t, S_{\tau_b-} \in \mathrm{d}u, \Delta_S(\tau_b) \in \mathrm{d}v] = \mathbb{1}\{0 < b(t) - u < v\}g_t(u)\mathrm{d}t\,\mathrm{d}u\,\nu(\mathrm{d}v); \tag{A.1a}$$

$$\mathbb{P}[\tau_b \in \mathrm{d}t, S_{\tau_b-} \in \mathrm{d}u, \Delta_S(\tau_b) = 0] = -b'(t)g_t(b(t))\,\mathrm{d}t\,\delta(\mathrm{d}u - b(t)). \tag{A.1b}$$

The proof of (A.1a) is based on the compensation formula for the Poisson point process of jumps of S. The proof of (A.1b) essentially rests on the identification of the probability of creeping by expressing the density of the crossing time τ_b in two ways. It is this step that requires the continuity of the density $g_t(x)$.

Proof of Theorem A.1. First let us consider (A.1a). Let $f : (0, T_b) \times [0, \infty) \times (0, \infty) \to \mathbb{R}_+$ be a Borel function, such the $f(t, u, v) = 0$ for $v = b(t) - u$. By part (a) of the theorem proved above, we have $\mathbb{P}(\tau_b < T_b) = 1$. Thus, since b is non-increasing and S has increasing paths, we have

$$f(\tau_b, S_{\tau_b-}, \Delta_S(\tau_b)) = \sum_{t \in (0, T_b)} f(t, S_{t-}, \Delta_S(t))\mathbb{1}\{0 \leqslant b(t) - S_{t-} < \Delta_S(t)\}. \tag{A.2}$$

For each $t > 0$, define a σ_α-algebra $\mathcal{F}_t := \sigma_\alpha(S_s : s \in [0,t])$ generated by S during the time interval $[0,t]$ the random function $H_t(\cdot) = f(t, S_{t-}, \cdot)\mathbb{1}\{0 \leqslant b(t) - S_{t-} < \cdot\}$ on $(0, \infty)$. $H = (H_t)$ is a predictable process with respect to the filtration $(\mathcal{F}_t)_{t \in \mathbb{R}_+}$. By (A.2) and the compensation formula [6, Ch. 0.5] for the Poisson point process of jumps $\{\Delta_S(t) : t \in \mathbb{R}_+\}$, we get

$$\mathbb{E}[f(\tau_b, S_{\tau_b-}, \Delta_S(\tau_b))] = \int_0^{T_b} \mathrm{d}t\, \mathbb{E}\Big[\int_{(0,\infty)} f(t, S_{t-}, v)\mathbb{1}\{0 \leqslant b(t) - S_{t-} < v\}\nu(\mathrm{d}v)\Big]$$

$$\overset{(*)}{=} \int_0^{T_b} \mathrm{d}t \int_{[0,\infty)\times(0,\infty)} f(t, u, v)\mathbb{1}\{0 \leqslant b(t) - u < v\}g_t(u)\mathrm{d}u\,\nu(\mathrm{d}v)$$

$$= \int_{(0,T_b)\times[0,\infty)\times(0,\infty)} f(t, u, v)\mathbb{1}\{0 \leqslant b(t) - u < v\}g_t(u)\mathrm{d}t\,\mathrm{d}u\,\nu(\mathrm{d}v),$$

where $(*)$ follows from the Fubini theorem and a.s. equality $S_{t-} = S_t$ for any $t > 0$. Since f is arbitrary, this implies that the vector $(\tau_b, S_{\tau_b-}, \Delta_S(\tau_b))$, on the event $\{\Delta_S(\tau_b) > 0\} \setminus \{S_{\tau_b-} < b(\tau_b), S_{\tau_b} = b(\tau_b)\}$, has the law $\mathbb{1}\{0 \leqslant b(t) - u < v\}g_t(u)\mathrm{d}t\,\mathrm{d}u\,\nu(\mathrm{d}v)$. Since $\mathbb{1}\{0 \leqslant b(t) - u < v\}g_t(u) = \mathbb{1}\{0 < b(t) - u < v\}g_t(u)$ the equality holds $\mathrm{d}t\,\mathrm{d}u\,\nu(\mathrm{d}v)$-a.e., since the set $\{(t, u, v) \in (0, T_b) \times [0, \infty) \times (0, \infty) : b(t) = u\}$ is not charged by the measure $\mathrm{d}t\,\mathrm{d}u\,\nu(\mathrm{d}v)$. The compensation formula applied to the function $f : (0, T_b) \times [0, \infty) \times (0, \infty) \to \mathbb{R}_+$, given by $f(t, u, v) := \mathbb{1}_{\{b(t)-u=v, 0<v\}}$, as in the previous display and the fact that S_t has a density imply that $\mathbb{P}(S_{\tau_b-} < b(\tau_b), S_{\tau_b} = b(\tau_b)) = 0$. Hence we have characterised the law of $(\tau_b, S_{\tau_b-}, \Delta_S(\tau_b))$, on the event $\{\Delta_S(\tau_b) > 0\}$ and (A.1a) follows.

For any $c \in (0, \infty)$ and $t > 0$, define $\varpi(c, t) := \int_0^c \overline{\nu}(c - u)g_t(u)\mathrm{d}u$, where $\overline{\nu}(x) = \nu((x, \infty))$. By (A.1a), we have $\mathbb{P}[\tau_b \in \mathrm{d}t, S_{\tau_b} > b(\tau_b)] = \varpi(b(t), t)\mathrm{d}t$. In particular, when b is constant, i.e. $b(t) = c_0 > 0$ for all $t \in (0, \infty)$, we have $\mathbb{P}[S_{\tau_{c_0}} > c_0] = 1$ by [6, Prop. 3.iv], since S is assumed to have no drift. Thus, $\mathbb{P}[\tau_{c_0} \in \mathrm{d}t] = \mathbb{P}[\tau_{c_0} \in \mathrm{d}t, S_{\tau_{c_0}} > c_0] = \varpi(c_0, t)\mathrm{d}t$ and $t \mapsto \varpi(c_0, t)$ is a density of τ_{c_0}.

Now consider (A.1b). Fix t, set $a := b(t)$ and let $\tau_a := \inf\{s > 0 : S_s > a\}$. Pick $\epsilon > 0$ and note

$$\{t - \epsilon < \tau_b \leq t\} = \{S_{t-\epsilon} < b(t - \epsilon)\} \cap \{S_t \geqslant b(t)\}$$

$$= \{S_{t-\epsilon} < a \leq S_t\} \cup \{b(t) \leq S_{t-\epsilon} < b(t - \epsilon)\}$$

$$= \{t - \epsilon < \tau_a \leq t\} \cup \{b(t) \leq S_{t-\epsilon} < b(t - \epsilon)\}$$

Letting $q(\epsilon) := \mathbb{P}[t - \epsilon < \tau_b \leq t]$, for $\epsilon > 0$, we have $q(\epsilon) = \mathbb{P}[S_{t-\epsilon} < b(t - \epsilon), S_t > b(t)] = q_1(\epsilon) + q_2(\epsilon)$, where $q_1(\epsilon) := \mathbb{P}[S_{t-\epsilon} < a \leq S_t] = \mathbb{P}[t - \epsilon < \tau_a \leq t]$, $q_2(\epsilon) := \mathbb{P}[b(t) \leq S_{t-\epsilon} < b(t - \epsilon)]$. On one hand, $q_1(\epsilon)/\epsilon \to \varpi(a, t) = \varpi(b(t), t)$ as $\epsilon \downarrow 0$. On the other hand, $q_2(\epsilon) = \int_0^{b(t-\epsilon)-b(t)} g_{t-\epsilon}(a + x)\mathrm{d}x$. Recall that the mapping $(t, x) \mapsto g_t(x)$ is continuous on $(0, \infty) \times (0, \infty)$, we have $q_2(\epsilon)/\epsilon \to -b'(t)g_t(b(t))$ as $\epsilon \downarrow 0$. Thus

$$\frac{\mathbb{P}[\tau_b \in \mathrm{d}t]}{\mathrm{d}t} = \lim_{\epsilon \downarrow 0} \frac{\mathbb{P}[\tau_b \leq t] - \mathbb{P}[\tau_b \leq t - \epsilon]}{\epsilon}$$

$$= -b'(t)g_t(b(t)) + \varpi(b(t), t) = -b'(t)g_t(b(t)) + \frac{\mathbb{P}[\tau_b \in \mathrm{d}t, S_{\tau_b} > b(\tau_b)]}{\mathrm{d}t},$$

implying $\mathbb{P}[\tau_b \in \mathrm{d}t, S_{\tau_b} = b(\tau_b)] = -b'(t)g_t(b(t))$ and hence (A.1b). $\qquad\square$

A.2 Moments of a crossing time of a level for infinite activity subordinators

A first-passage time over a level of an infinite activity Lévy process has exponential moments of all order.

43

Lemma A.2. *Given a subordinator X with Lévy measure ν on $(0, \infty)$ satisfying $\nu((0, \infty)) = \infty$. For any $c > 0$, define $\tau_c := \inf\{t > 0 : X_t > c\}$ and $\psi(u) := \int_{(0,\infty)}(1 - e^{-ux})\nu(dx)$ for $u > 0$. Then, for any $p \in (0, \infty)$, there exists $u > 0$ such that $\psi(u) > p$ and*

$$\mathbb{E}[e^{p\tau_c}] \leqslant 1 + \frac{e^{uc}}{\psi(u) - p}.$$

Moreover, $\mathbb{E}[\tau_c] \leqslant e^{uc}/\psi(u)$ for $u > 0$.

Proof of Lemma A.2. Given a differentiable increasing function φ with $\varphi(0) = 0$ and a positive random variable ξ we have

$$\mathbb{E}[\varphi(\xi)] = \int_0^\infty \varphi(x)\mathbb{P}[\xi \in dx] = \int_0^\infty \int_0^x \varphi'(y)dy\mathbb{P}[\xi \in dx] = \int_0^\infty \varphi'(y)\mathbb{P}[\xi > y]dy.$$

First let us set $\varphi(x) = e^{px} - 1$. Then, for $u > 0$, we have

$$\mathbb{P}[\tau_c > t] \leqslant \mathbb{P}[X_t \leqslant c] = \mathbb{P}[e^{-uX_t} \geqslant e^{-uc}] \leqslant \mathbb{E}[e^{-uX_t}]e^{uc} = \exp(uc - t\psi(u)).$$

Therefore, for any $u > 0$ with $\psi(u) > p$ (recall u is a free parameter), we have

$$\int_0^\infty e^{pt}\mathbb{P}[\tau_c > t]dt \leqslant \int_0^\infty e^{pt+uc-t\psi(u)}dt = \frac{e^{uc}}{\psi(u) - p}.$$

Now set $\varphi(x) = x$, we have

$$\mathbb{E}[\tau_c] = \int_0^\infty \mathbb{P}[\tau_c > y]dy \leqslant \int_0^\infty e^{uc-t\psi(u)}dt = \frac{e^{uc}}{\psi(u)}. \qquad \square$$

B Tempering

Fix any $\theta > 0$ and $\alpha \in (0, 1)$. Let S be a driftless stable subordinator under the probability measure $\mathbb{P}$ with Lévy measure $\nu(dx) = w_\alpha^{-1}\theta x^{-\alpha-1}dx$. Denote by $(\mathcal{G}_t)_{t \geqslant 0}$ be the right-continuous filtration generated by S. Given $q \geqslant 0$, let $\mathbb{P}$ be the probability measure with Radon–Nikodym derivative

$$\left.\frac{d\mathbb{P}_q}{d\mathbb{P}}\right|_{\mathcal{G}_t} = M_q(t) := \exp\big(-qS_t + q^\alpha\theta t\big), \quad t > 0.$$

Then, by [Sato, Thm 33.2], S is a tempered stable subordinator under $\mathbb{P}_q$ with Lévy measure $\nu_q(dx) = w_\alpha^{-1}\theta e^{-qx}x^{-\alpha-1}dx$. Denote by $\mathbb{E}$ and $\mathbb{E}_q$ the expectation under the measures $\mathbb{P}$ and $\mathbb{P}_q$, respectively. We have following lemma:

Lemma B.1. *For any $B \in \mathcal{G}_t = \sigma(S_s : s \in [0, t])$, let $E \sim \text{Exp}(1)$ and $E \perp\!\!\!\perp \mathcal{G}_t$. Then*

$$\mathbb{P}_q[B] = \mathbb{P}[B|E \geqslant qS_t]. \tag{B.1}$$

Moreover, for any $C \in \mathcal{G}_t$ satisfying $\mathbb{P}[C] > 0$, we have

$$\mathbb{P}_q[B|C] = \mathbb{P}[B|C, E \geqslant qS_t]. \tag{B.2}$$

Proof. Note that

$$\mathbb{P}[E \geqslant qS_t | S_t] = \exp(-qS_t) \qquad \& \qquad \mathbb{P}[E \geqslant qS_t] = \mathbb{E}[\exp(-qS_t)] = \exp(-q^\alpha \theta t).$$

Therefore

$$\mathbb{P}[B|E \geqslant qS_t] = \frac{\mathbb{E}[\mathbb{1}_B \mathbb{1}_{\{E \geqslant qS_t\}}]}{\mathbb{P}[E \geqslant qS_t]} = \frac{\mathbb{E}[\mathbb{E}[\mathbb{1}_B \mathbb{1}_{\{E \geqslant qS_t\}} | \mathcal{G}_t]]}{\exp(-q^\alpha \theta t)} = \mathbb{E}_q[\mathbb{1}_B \mathbb{E}[\mathbb{1}_{\{E \geqslant qS_t\}} | \mathcal{G}_t] \exp(q^\alpha \theta t)]$$

$$= \mathbb{E}[\mathbb{1}_B \exp(-qS_t + q^\alpha \theta t)] = \mathbb{E}[\mathbb{1}_B M_q(t)] = \mathbb{P}_q[B].$$

Moreover,

$$\mathbb{P}_q[B|C] = \frac{\mathbb{P}_q[B, C]}{\mathbb{P}_q[C]} = \frac{\mathbb{P}[B, C | E \geqslant qS_t]}{\mathbb{P}[C | E \geqslant qS_t]} = \frac{\mathbb{P}[B, C, E \geqslant qS_t]/\mathbb{P}[E \geqslant qS_t]}{\mathbb{P}[C, E \geqslant qS_t]/\mathbb{P}[E \geqslant qS_t]} = \mathbb{P}[B | C, E \geqslant qS_t]. \qquad \square$$

C Useful Algorithms

C.1 Simulation of stable and tempered stable random variables

Let S_t be a positive stable random variable with density g_t given in (4.3). More precisely, $S_t \overset{d}{=} t^{1/\alpha} S_1$ for all $t \in (0, \infty)$, and, by [26, §4.4], for any $x \in (0, \infty)$ we have $g_t(x) = \varphi_\alpha((\theta t)^{-1/\alpha} x)(\theta t)^{-1/\alpha}$ where $\varphi_\alpha(x) = r \int_0^1 \sigma_\alpha(u) x^{-r-1} e^{-\sigma_\alpha(u) x^{-r}} du$ (with $r := \alpha/(1-\alpha) \in (0, \infty)$), $\sigma_\alpha(u) = \rho(u)^{r+1}$ and $\rho(u) = \sin(\alpha \pi u)^\alpha \sin((1-\alpha)\pi u)^{1-\alpha}/\sin(\pi u)$ for $u \in (0, 1)$. The following algorithm samples from the law of S_t.

Algorithm 8 (Chambers–Mallows–Stuck, S-Alg) Simulation from the law of S_t under $\mathbb{P}_0$

Require: Parameters $\alpha \in (0, 1)$ and $\theta, t > 0$

1: Generate $U \sim \mathrm{U}(0, 1)$ and $E \sim \mathrm{Exp}(1)$

2: **return** $(\theta t)^{1/\alpha} (\sigma_\alpha(U)/E)^{(1-\alpha)/\alpha}$

Note that the distribution function of S_t is given by $x \mapsto \int_0^1 \exp(-\sigma_\alpha(u)(x/(\theta t)^{1/\alpha})^{-r}) du$, which clearly equals the distribution function of the output of S-Alg. Moreover, S-Alg clearly has constant computational cost, which does not depend on the parameters θ, t, α.

For the simulation of S_t under $\mathbb{P}_q$ (i.e., a tempered stable marginal), we use the algorithm in [24, Alg. 3.1], which outperforms [10] and is essentially a rejection-sampling algorithm with expected complexity uniformly bounded by 4.2154 for all parameter combinations, see [24, Thm 3.1]. Recall the definition of the function ρ in (4.2). Define the error function $\mathrm{erf}(x) := (2/\sqrt{\pi}) \int_0^x e^{-t^2} dt$ and the functions $\Psi_i : (0, 1) \times [0, \infty) \to \mathbb{R}$, $i \in \{0, 1, 2, 3, 4\}$, given by

$$\Psi_0(a, x) := \frac{\mathrm{erf}\big(\sqrt{a(1-a)x\pi^2/2}\big)}{\sqrt{2\pi a(1-a)x}},$$

$$\Psi_1(a, x) := \frac{\Gamma(ax)e^{ax-1}}{(ax)^x}\Big(\tfrac{a}{1-a} + ax\Big)^{1+(1-a)x}, \quad \Psi_2(a, x) := \frac{\Gamma(1+(1-a)x)e^{(1-a)x}}{((1-a)x)^{(1-a)x}},$$

$$\Psi_3(a, x) := \frac{\Gamma(1+ax)e^{ax-1}(1+1/[(1-a)x])^{1+(1-a)x}}{(ax)^{ax}\sqrt{2\pi a(1-a)x}}, \quad \Psi_4(a, x) := \frac{\Gamma(1+(1-a)x)e^{(1-a)x}}{((1-a)x)^{(1-a)x}\sqrt{2\pi a(1-a)x}}.$$

Denote by $\mathrm{Gamma}(a, \theta)$ the gamma distribution with density $x \mapsto \theta^a x^{a-1} e^{-\theta x}/\Gamma(a)$ and by $\mathrm{N}(0, \theta^2)|_{[0,1]}$ the truncated normal distribution with center 0 and scale θ but restricted to the interval $[0, 1]$, i.e., with

distribution function $x \mapsto \mathrm{erf}\big(x/\sqrt{2\theta^2}\big)/\mathrm{erf}\big(1/\sqrt{2\theta^2}\big)$. Note as well that a sample from $\mathrm{N}(0,\theta^2)|_{[0,1]}$ can be obtained in constant time as there exist algorithms that numerically evaluate the inverse of erf accurately and with logarithmic cost in the number of precision bits N.

Algorithm 9 ([24, Alg. 3.1], TS-Alg) Simulation from the law of S_t under $\mathbb{P}_q$

Require: Parameters $\alpha \in (0,1)$ and $\theta, t > 0$ and $q \geqslant 0$

1: Set the constants $\lambda \leftarrow (\theta t)^{1/\alpha} q$, $\xi = \lambda^\alpha$, $r = \alpha/(1-\alpha)$ and $C_i = \Psi_i(\alpha, \xi)$ for $i \in \{0,1,2,3,4\}$

2: **if** $C_1 = \min\{C_1, C_2, C_3, C_4\}$ **then**

3:     **repeat**

4:         Sample $U, V \sim \mathrm{U}(0,1)$ and $X \sim \mathrm{Gamma}(\alpha\xi, 1)$ and set $s \leftarrow X/\lambda$

5:     **until** $V \leqslant re^\xi \Gamma(\alpha\xi)(\rho(U)\xi)^{r+1}X^{-r-\alpha\xi}\exp\big(-(\rho(U)\xi)^{r+1}X^{-r}\big)/C_1$

6: **else if** $C_2 = \min\{C_1, C_2, C_3, C_4\}$ **then**

7:     **repeat**

8:         Sample $U, V \sim \mathrm{U}(0,1)$ and $X \sim \mathrm{Gamma}(1 + (1-\alpha)\xi, 1)$ and set $s \leftarrow \rho(U)^{1/\alpha}X^{-1/r}$

9:     **until** $V \leqslant e^\xi \Gamma(1 + (1-\alpha)\xi)X^{-(1-\alpha)\xi}\exp(-\lambda s)/C_2$

10: **else if** $C_3 = \min\{C_1, C_2, C_3, C_4\}$ **then**

11:     **repeat**

12:         Sample $U \sim \mathrm{N}(0, 1/[\pi^2\alpha(1-\alpha)\xi])|_{[0,1]}$, $V \sim \mathrm{U}(0,1)$ and $X \sim \mathrm{Gamma}(\alpha\xi, 1)$ and set $s \leftarrow X/\lambda$

13:     **until** $V \leqslant C_0 re^\xi \Gamma(\alpha\xi)(\rho(U)\xi)^{r+1}X^{-r-\alpha\xi}\exp\big(\alpha(1-\alpha)\xi U^2/2 - (\rho(U)\xi)^{r+1}X^{-r}\big)/C_3$

14: **else**

15:     **repeat**

16:         Sample $U \sim \mathrm{N}(0, 1/[\pi^2\alpha(1-\alpha)\xi])|_{[0,1]}$, $V \sim \mathrm{U}(0,1)$ and $X \sim \mathrm{Gamma}(1 + (1-\alpha)\xi, 1)$ and set $s \leftarrow \rho(U)^{1/\alpha}X^{-1/r}$

17:     **until** $V \leqslant C_0 e^\xi \Gamma(1 + (1-\alpha)\xi)X^{-(1-\alpha)\xi}\exp\big(\alpha(1-\alpha)\xi U^2/2 - \lambda s\big)/C_4$

18: **end if**

19: **return** $(\theta t)^{1/\alpha}s$

C.2 Rejection sampling from log-concave density

Devroye's extremely efficient method [11] for the simulation from a log-concave densities is crucial in several steps of our main SFP-Alg. The general algorithm in [11] samples from densities defined on $\mathbb{R}_+$. Here we phrase it for log-concave densities defined on $[0,1]$, as all the densities arising in SFP-Alg are defined on a compact interval contained in $[0,1]$. Recall that $f : [0,1] \to \mathbb{R}_+$ is log-concave by definition if $\log(f)$ is a concave function on $[0,1]$.

    Computational cost of LC-Alg consist of two parts: the computational cost of finding a_1, which may be random if the density f depends on parameters that are themselves random, and the random computational cost of the accept-reject step, which is in expectation bounded above by 5 [11, Section 4.4].

C.3 Numerical inversion

In $\psi^{(2)}$-Alg, we will apply Newton–Raphson method (NR-Alg below) to find the root x_*, satisfying $f(x_*) = 0$, of an increasing function $f : (0,1) \to \mathbb{R}$. The search terminates when the computed approximation to

Algorithm 10 ([11], LC-Alg) Simulation from a log-concave density on $[0, 1]$

Require: Log-concave $f : [0, 1] \to \mathbb{R}_+$ with mode at 0 and $f(0) = 1$, proportional to the target density

1: Find $a_1 \in \{2^{-1}, 2^{-2}, \ldots\}$ such that $f(a_1) \geqslant 1/4 \geqslant f(2a_1)$

2: Define $h(x) = \mathbb{1}_{(0,a_1]}(x) + \mathbb{1}_{(a_1,2a_1]}(x)f(a_1) + \mathbb{1}_{(2a_1,1]}(x)\exp(a_1^{-1}[(2a_1 - x)\log f(a_1) + (x - a_1)\log f(2a_1)])$

3: Set $a_0 \leftarrow a_1 + a_1 f(a_1) + a_1 f(2a_1)/\log(f(a_1)/f(2a_1))$ $\triangleright$ (note $a_0 = \int_0^1 h(x)\mathrm{d}x$)

4: **repeat**

5:      Generate $V_1, V_2, V_3 \sim \mathrm{U}(0, 1)$

6:      **if** $a_0 V_2 \leqslant a_1$ **then**

7:          Set $X \leftarrow a_1 V_1$

8:      **else if** $a_0 V_2 \leqslant a_1 + a_1 f(a_1)$ **then**

9:          Set $X \leftarrow a_1 + a_1 V_1$

10:     **else**

11:         Set $X \leftarrow 2a_1 + a_1 \log(1/V_1)/\log(f(a_1)/f(2a_1))$

12:     **end if**

13: **until** $X \leqslant 1$ and $V_3 \leqslant f(X)/h(X)$

14: **return** X

---

x_* coincides with the actual root x_* in the first N bits (e.g., in double-precision floating-point arithmetic $N = 53$).

In order to control the running time of NR-Alg when applied to a twice differentiable increasing function $f : (0, 1) \to \mathbb{R}$ with root x_*, we need access to an auxiliary function $M : [0, 1] \times \mathbb{N} \to \mathbb{R}_+$ such that

$$M(x_0, k) \geqslant \frac{1}{2}\frac{\sup_{x \in [x_*, x_0]} |f''(x)|}{\inf_{x \in [x_*, x_0]} |f'(x)|} \qquad \text{for all } (x_0, k) \in [0, 1] \times \mathbb{N} \text{ satisfying } |x_* - x_0| \leqslant 2^{1-k}. \qquad \text{(C.1)}$$

In practice, for a specific function f, $M(x_0, k)$ may, but need not, depend on either x_0 or k (see e.g. Propositions 4.5. 4.6 and 4.7). The main purpose of M is to guarantee that the starting point x_0 of the Newton–Raphson method is sufficiently close to the root x_* so that the convergence is quadratic. Constructing M for specific functions f requires *a priori* estimates on f.

NR-Alg starts at the right-end point of $[0, 1]$ and, using bisection (line 2 to line 7, finds a starting point x_0 for the Newton–Raphson method to have quadratic convergence. In fact, the error sequence $(\varepsilon_n)_{n \in \mathbb{N}}$ (where $\varepsilon_n := |x_n - x_*|$ is the error at the n-th iteration) satisfies $\varepsilon_{n+1} \leqslant M\varepsilon_n^2$, implying that $\varepsilon_n \leqslant M^{2^n - 1}\varepsilon_0^{2^n}$. Thus, for the Newton–Raphson methodology to attain quadratic convergence, it suffices to find x_0 such that $M\varepsilon_0 \leqslant 1/2$, as this would imply that $\varepsilon_n \leqslant (1/2)^{2^n - 1}$, making N bits of precision attainable in $\log N$ iterations. The computational cost of NR-Alg has two parts: (I) the cost to finding the value of x_0 and (II) the cost to compute the Newton–Raphson sequence $x_{n+1} = x_n - f(x_n)/f'(x_n)$ until the error reaches desired precision. (I) depends on $2^{1-k}M(x_0, k)$ and is in fact bounded above by N. (II) is bounded by $\log N$ as we explained before.

Householder's method (H-Alg below) of order $k \in \mathbb{N}$ generalises Newton–Raphson (the latter is recovered when $k = 1$). Moreover, its convergence rate is of order $k + 1$: under certain regularity assumptions, the error $\varepsilon_n := |x_n - x|$ of the n-th iteration satisfies $\varepsilon_{n+1} \leqslant M_k \varepsilon_n^{k+1}$ for $n \in \mathbb{N} \cup \{0\}$ and an appropriate $M_k > 0$. This implies that $\varepsilon_n \leqslant M_k^{((k+1)^n - 1)/k}\varepsilon_0^{(k+1)^n}$ for $n \in \mathbb{N}$. The drawback of this method is that it requires an evaluation of $f, f', \ldots, f^{(k)}$ at each step. This drawback usually outweighs the improved

---

Algorithm 11 (NR-Alg) Newton–Raphson root-finding method

Require: Twice differentiable increasing function $f : (0, 1) \to \mathbb{R}$, derivative $f' : (0, 1) \to (0, \infty)$, auxiliary
function $M : [0, 1] \times \mathbb{N} \to \mathbb{R}_+$ in (C.1)

1: Set $n = 0$, $x_0 = k = 1$
2: **repeat**
3:     **if** $f(x_0 - 2^{-k}) > 0$ **then**
4:         Set $x_0 \leftarrow x_0 - 2^{-k}$
5:     **end if**
6:     Set $k \leftarrow k + 1$
7: **until** $2^{1-k} M(x_0, k) < 1/2$
8: **repeat**
9:     $x_{n+1} = x_n - f(x_n)/f'(x_n)$, $n \leftarrow n + 1$
10: **until** $|x_n - x_{n-1}| \leqslant 2^{-N}$
11: **return** x_n

---

convergence, except in the cases where evaluation of f requires a numerical integration and the evaluation of its derivatives is much cheaper by comparison (this is the case in Subsection 2.6 above). We stress that H-Alg is only used to compare the performance of our simulation algorithms $\psi^{(1)}$-Alg and $\psi^{(2)}$-Alg with a simulation using a direct numerical inversion (via H-Alg) of the corresponding distribution functions.

Algorithm 12 (H-Alg) Householder method

Require: Index k, initial guess $x_0 \in (0, 1)$ and $(k + 1)$-differentiable increasing function $f : (0, 1) \to \mathbb{R}$
with a zero on $[0, 1]$ and derivatives $g_m(x) = \frac{\mathrm{d}^m}{\mathrm{d}x^m}(1/f(x))$ for $m \in \{k - 1, k\}$.

1: Set $n = 0$,
2: **repeat**
3:     $x_{n+1} \leftarrow x_n + k g_{k-1}(x_n)/g_k(x_n)$, $n \leftarrow n + 1$
4: **until** $|x_n - x_{n-1}| \leqslant 2^{-N}$
5: **return** x_n

D   Practically exact simulation algorithms

The answer to the question in the title of Subsection 2.8, whether an implemented simulation algorithm can be exact, depends on the definition of exact simulation. The simplest, and in some sense most natural, definition requires the output of the algorithm to have the same law as the variable it is simulating. In this case, the answer is clearly no: working with $N \in \mathbb{N}$ bits of precision makes the output lie on a grid, implying that the total variation (TV) distance between a simulated output and any target law with a density equals one.[3] A way of dealing with this issue might be to understand and minimize a natural distance between the output of an implemented algorithm and a draw from the target law. Four natural ways of measuring the distance between algorithm's output and the actual law are as follows.
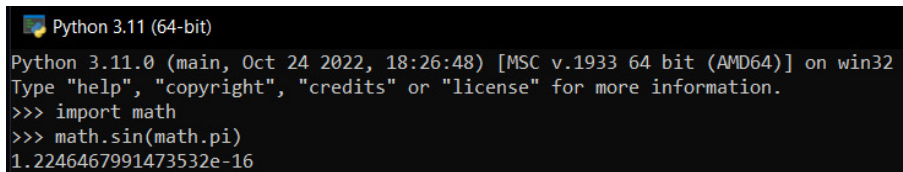
(I) L^∞-Wasserstein distance. Let A (resp. I) denote the output of an implemented algorithm (resp. "ideal"

---
[3]We thank Luc Devroye for pointing this out to us.

sample form the target law). For any coupling λ of the laws of A and I, let $\mathbb{P}_\lambda$ be a probability measure such that (A, I) has law λ under $\mathbb{P}_\lambda$. The $L^\infty(\mathbb{P}_\lambda)$-distance is given by the essential supremum $\mathbb{P}_\lambda\text{-}\mathrm{ess\,sup}\,|A - I| := \inf\{a > 0 : \mathbb{P}_\lambda(|A - I| > a) = 0\}$ with convention $\inf \emptyset = \infty$ and the L^∞-Wasserstein distance equals the infimum of $L^\infty(\mathbb{P}_\lambda)$-distances over all couplings λ. Numerical inversion algorithms ought to control precisely this distance. However, it appears that even a simple accept-reject method, implemented on a computer cannot reduce this distance, regardless of the precision used (i.e. for any $N \in \mathbb{N}$). Indeed, consider an implementation of LC-Alg (in Appendix C.2 below), together with the natural coupling, obtained by running the algorithm on a computer and on an ideal computer (IC), which carries out all evaluations without error. Even if the constants a_0, a_1 and $f(a_1)$ could be represented exactly on the grid, the event-detection occurring in lines 6 and 8 will have a chance of being misdetected by the computer, simply because the variable V_2 is not uniform on $(0, 1)$ but instead lives on the grid. Therefore, there is a (small, but non-zero) chance that the value of I originated from line 7, while the value of A came from line 9. On this event of positive probability, we have $|A - I| = a_1 > 0$, regardless of the precision we are using. The same phenomenon occurs with positive probability with lines 9 and 11, making the L^∞-distances for the natural coupling in fact infinite. Of course, the L^∞-Wasserstein distance may be smaller, but there appears to be no easy way of finding a good bound on it other than through the natural coupling.

(II) Total variation distance on a grid. Another natural way of measuring the distance between the variables A and I would be to project I onto the state space (i.e. grid in the computer) of A and consider the total variation distance (TV-distance) between the two discrete variables. However, rounding errors can make the TV-distance be equal to one for any level $N \in \mathbb{N}$ of precision bits used on a computer: consider, for example, the deterministic random variable 0 and assume the simulation algorithm first samples the deterministic variable π and then computes $\sin(\pi)$. This algorithm on an IC returns zero, which is already on the grid (so the projection does nothing). However, regardless of the number of precision bits N, our computer may, due to rounding errors, return a non-zero value on the grid, see Figure D.1. In particular, in this case A and the projection of I onto the grid are at TV-distance one for all $N \in \mathbb{N}$. Note in passing that the L^∞-distance in this case decreases with increasing precision N, while the TV-distance does not.

Figure D.1: Rounding errors in Python produce singular random variables.

(III) L^∞-Wasserstein distance of cdf mapping. In [19, § 7.2, p. 159], the authors propose controlling the difference $|F_I(A) - F_I(I)|$ a.s., where F_I denotes the distribution function of I. If F_I is continuous, it is then uniformly continuous. If ω_I denotes the modulus of continuity of F_I, then $|F_I(A) - F_I(I)| \leq \omega_I(|A - I|)$, in which case, it suffices to control the L^∞-Wasserstein distance as in (I). However this distance may also be bounded away from zero with tiny (but positive) probability, thus suffering from the same issue as the one described for L^∞-Wasserstein distance.

(IV) Prokhorov distance. The Prokhorov distance metrises weak convergence, making it a prime candidate for what a "natural" distance is. (Recall that, given two probability measures μ_1 and μ_2 on the metric space $(d, \mathbb{S})$, then $d_P(\mu_1, \mu_2) := \inf\{\varepsilon > 0 : \mu_1(F) \leq \mu_2(F^\varepsilon) + \varepsilon$ for all closed $F \subset \mathbb{S}\}$, where $F^\varepsilon := \{x :$

$d(x, F) < \varepsilon\}$, is the Prokhorov distance between them.) By [12, Cor. 1],

$$d_P(\mathcal{L}(I), \mathcal{L}(A)) = \inf\{\varepsilon > 0 : \text{there exists a coupling } \lambda \text{ between } I \text{ and } A \text{ with } \lambda(|I - A| > \varepsilon) \leqslant \varepsilon\}.$$

In words, the Prokhorov distance is the smallest $\varepsilon > 0$ such that, under a suitable coupling and with probability at least $1 - \varepsilon$, the distance between the ideal variable I and the output A of the algorithm is at most ε. In this sense, if an algorithm contains k_1 numerical inversions and k_2 reject sampling steps such that (i) each accept-reject step controls the error of accepting or rejecting a samples by ε_0 and (ii) each numerical inversion step controls the distance between the output and the true inverse by ε_0, then the Prokhorov distance between I and A is bounded by $\max\{k_1, k_2\}\varepsilon_0$.

The discussion in (I)–(IV) above illustrates how one can quantify a deviation from the strong definition of exact simulation. In this paper we use the following definitions: an algorithm implemented on a computer is *exact* if, given exact values of the iid uniforms and assuming all the elementary operations carried out by the algorithm are exact, the returned value has the correct law (i.e. with zero TV-distance from the target law). In this definition, elementary operations include evaluations of elementary functions and their integrals as well as inversions. Moreover, we say that an algorithm is *practically exact*, with respect to a chosen distance d and a given tolerance ε, if $d(\mathcal{L}(I), \mathcal{L}(A)) \leqslant \varepsilon$.

Our main aim in this paper is to implement a fast (practically) exact algorithm (see Appendix D.1 below) and give theoretical guarantees for its running time, so that the algorithm can be used within a Monte Carlo estimation procedure. In exact simulation algorithms, function inversion is often implicitly assumed to produce exact values and take a bounded number of steps, uniformly in the input (see e.g. [8, Table 1, line 2]). In the case of our main simulation method (TSFFP-Alg above), we instead carefully analyse the cost of each Newton–Raphson call needed to perform an inversions within the error margin, smaller than what can be represented on the computer. All other elementary operations, including the evaluation of the definite integrals in SU-Alg (Step 1) and $\psi^{(2)}$-Alg (Step 3) are assumed to have constant complexity (the integrands are well behaved and close to those arising in the representation of a stable density, making the numerical procedures converge fast, see remarks following the algorithms).

Finally, we note that the error between simulated variables on a grid and their idealised counterparts (e.g. for the increment of Brownian motion) are typically not taken into account when analysing the convergence properties of Monte Carlo estimates. This further motivates our definition of (practically) exact simulation for an implemented algorithm.

D.1 TSFFP-Alg is practically exact

Using the definition above, TSFFP-Alg is practically exact with respect to the Prokhorov distance and with tolerance equal to $C2^{-N}$, where C is a constant that may be computed in terms of $(\theta, \alpha, q, b(0))$ and N is the number of precision bits (typically, $N = 53$). Indeed, TSFFP-Alg consists of a loop of at most $1 + \lfloor b(0)/R \rfloor$ repeated calls to TSFP-Alg. In each call to TSFP-Alg, we have (in lines 8-11) 3 accept-reject steps (accounting for all the calls to other algorithms) and at most 2 numerical inversions, each with an error bounded by 2^{-N}, and (in lines 2–7) a loop where each step calls TS-Alg, containing one numerical inversion and one accept-reject step (again, each with an error bounded by 2^{-N}). The number of iterations of this loop in TSFP-Alg is equal to $1 + \lfloor \tau_b/t_* \rfloor$, where, by Markov's inequality and Lemma A.2, we have

$$\mathbb{P}(1 + \lfloor \tau_b/t_* \rfloor \geqslant n) \leqslant e^{-pt_*(n-1)}\left(1 + \frac{e^{uR}}{\theta((u+q)^\alpha - q^\alpha) - p}\right),$$

for any $n \in \mathbb{N}$, $u > 0$ and $p < \theta((u+q)^\alpha - q^\alpha)$. Pick $u = (3^{1/\alpha} - 1)q$ and $p = \theta q^\alpha$, then we have

$$\mathbb{P}(1 + \lfloor \tau_b/t_* \rfloor \geqslant n) \leqslant e^{-(\theta q^\alpha - 1)t_*(n-1)}\left(1 + e^{(3^{1/\alpha}-1)qR}/(\theta q^\alpha)\right) \leqslant n_0 2^{-N},$$

for all $n \geqslant n_0$ and some minimal $n_0 \in \mathbb{N}$ that can be written explicitly in terms of all other parameters. Thus, the accumulation of errors in Porkhorov distance of these repeated calls to TS-Alg is bounded by $n_0 2^{-N}$. Hence, TSFFP-Alg is practically accurate with tolerance $(n_0 + 3)(1 + b(0)/R)2^{-N}$.