Efficiently sampling Archimedean copulas

Marius Hofert[1]

2009-08-16

Abstract

Efficient sampling algorithms for both exchangeable and nested Archimedean copulas are presented. First, efficient sampling algorithms for the nested Archimedean families of Ali-Mikhail-Haq, Frank, and Joe are introduced. Second, a general strategy how to build a nested Archimedean copula from a given Archimedean generator is presented. Sampling this copula involves sampling an exponentially-tilted Stable distribution. For this task, a fast rejection algorithm is developed. It is proven for the more general class of tilted Archimedean generators that this algorithm reduces the complexity of the standard rejection algorithm to logarithmic complexity. As an application it is shown that this algorithm outperforms existing algorithms for sampling nested Clayton copulas. Third, with the additional help of randomization of generator parameters, explicit sampling algorithms for several nested Archimedean copulas based on different Archimedean families are found. As all algorithms work fast for large parameter ranges and do not require numerical inversion of Laplace transforms, they are recommendable for large-scale simulation studies, even in large dimensions. The presented ideas may also apply in the more general context of sampling distributions given by their Laplace-Stieltjes transforms.

Keywords

Exchangeable Archimedean copulas, nested Archimedean copulas, Laplace-Stieltjes transforms, sampling algorithms, exponentially-tilted Stable distributions.

AMS 2 000 subject classifications

Primary 60E10; secondary 60E05, 60E07, 62H99, 65C10.

1 Introduction

Sampling strategies for copulas are interesting from both a theoretical and an empirical perspective. Concerning the former, they often allow for a stochastic representation of the underlying random variables which may help accessing and understanding the properties of the underlying dependency. For practical applications, fast sampling algorithms are crucial for large-scale simulation studies, see e.g. Hofert and Scherer (2008) in the context of pricing financial products or Hering and Hofert (2009) in the context of goodness-of-fit testing.

In contrast to elliptical copulas such as the Gaussian or t copula, Archimedean copulas are given explicitly in terms of a one-place real function, called generator. All relevant

[1]Ulm University, Institute of Number Theory and Probability Theory, Helmholtzstraße 18, 89081 Ulm, Germany; marius.hofert@uni-ulm.de

properties of Archimedean copulas can be expressed in terms of their generators. Further, as they are not restricted to radial symmetry, Archimedean copulas are able to capture different kinds of tail dependence, a desired feature shared by many applications.

Assuming complete monotonicity, generators of Archimedean copulas correspond to distribution functions on the positive real line via Laplace-Stieltjes transforms. Knowing how to sample these distributions for an Archimedean family allows to sample Archimedean copulas in a fast and efficient way; the corresponding algorithms were presented by Marshall and Olkin (1988) for exchangeable and McNeil (2008) for nested Archimedean copulas. Whenever the distributions involved in an Archimedean structure are not explicitly known, one can compensate this shortcoming by using numerical inversion of Laplace transforms, as suggested by Hofert (2008) and Ridout (2008). A drawback of this approach for sampling nested Archimedean copulas is that the algorithm of McNeil (2008) involves sampling a distribution not only depending on the copula parameters, but additionally on a random variate obtained from an earlier step of the algorithm. In order to ensure that a numerical procedure works accurately for any such random variate, careful checks have to be made which are usually time-consuming. Further, if the underlying distribution is absolutely continuous, then the corresponding random variate almost surely changes with every vector of random variates to be generated. Numerical inversion algorithms for Laplace transforms therefore often do not provide reliable results, let alone within an acceptable amount of time. Moreover, direct sampling strategies do not suffer from these drawbacks and are usually faster.

In this paper, we deal with efficient sampling algorithms for exchangeable and nested Archimedean copulas without using numerical inversion of Laplace transforms. Special focus is put on the more general class of nested Archimedean copulas for which other sampling algorithms such as the conditional distribution method, see e.g. Embrechts et al. (2001), are not nearly as practical to apply, due to the derivatives involved. We present different ideas how the inverse Laplace-Stieltjes transforms involved in the algorithms of Marshall and Olkin (1988) and McNeil (2008) can be accessed and sampled. As a first result, we develop efficient sampling algorithms for the nested Archimedean families of Ali-Mikhail-Haq, Frank, and Joe. The ideas behind the presented algorithms may also apply to other Archimedean generators with discrete inverse Laplace-Stieltjes transforms. As a second result, we consider a general transform of Archimedean generators such that nested Archimedean copulas result. The inverse Laplace-Stieltjes transforms of the generators involved are exponentially-tilted Stable distributions. Such distributions are special cases of exponentially-tilted distributions on the positive real line, which correspond to tilted Archimedean generators, see Hofert (2008). We develop a fast version of the rejection algorithm for sampling exponentially-tilted distributions. Further, we investigate this algorithm exemplarily in the case of a nested Clayton copula and compare it with the algorithms of Rosiński (2007), Ridout (2008), and the standard rejection algorithm for sampling exponentially-tilted distributions. Contrary to the first two algorithms, the fast rejection algorithm is exact. We further prove that it is faster than the standard rejection algorithm reducing the complexity of the standard rejection algorithm to logarithmic complexity. Finally, randomization of generator parameters, together with the fast rejection algorithm, leads to explicit sampling algorithms for the

seven nested Archimedean copulas based on generators belonging to different families as presented in Hofert (2008). The presented sampling strategies may also apply to other sampling problems, see e.g. Joe and Hu (1996) or Joe (1997, p. 87) who discuss the construction of multivariate copulas by mixture representations even more general than nested Archimedean copulas.

This paper is organized as follows. In Section 2 we shortly recall the algorithms of Marshall and Olkin (1988) and McNeil (2008) for sampling exchangeable and nested Archimedean copulas. Section 3 contains the main part, starting with the algorithms for sampling nested Ali-Mikhail-Haq, Frank, and Joe copulas in Section 3.1. In Section 3.2 we present and extend a generator transform partly discussed in Hofert (2008). To sample the corresponding inverse Laplace-Stieltjes transform, we develop a fast version of the rejection algorithm in the more general framework of tilted Archimedean generators. In Section 3.3 we additionally apply the idea of randomizing generator parameters to develop explicit sampling algorithms for all nested Archimedean copulas based on generators belonging to different Archimedean families as presented in Hofert (2008). We consider examples in Section 4 and draw a conclusion in Section 5.

2 Sampling algorithms for exchangeable and nested Archimedean copulas

An *Archimedean generator*, shortly *generator*, is a continuous, decreasing function $\psi : [0,\infty] \to [0,1]$ which satisfies $\psi(0) = 1$, $\psi(\infty) := \lim_{t\to\infty} \psi(t) = 0$, and which is strictly decreasing on $[0, \inf\{t : \psi(t) = 0\}]$. McNeil and Nešlehová (2009) show that a generator defines an *exchangeable Archimedean copula*, given by

$$C(\mathbf{u}) = C(u_1,\dots,u_d;\psi) = \psi(\psi^{-1}(u_1) + \cdots + \psi^{-1}(u_d)),\ \mathbf{u} \in [0,1]^d, \tag{1}$$

for the inverse $\psi^{-1} : [0,1] \to [0,\infty]$ of ψ with $\psi^{-1}(0) := \inf\{t : \psi(t) = 0\}$, if and only if ψ is *d-monotone*, i.e. ψ is continuous on $[0,\infty]$, admits derivatives up to the order $d-2$ satisfying $(-1)^k \frac{d^k}{dt^k}\psi(t) \geq 0$ for all $k \in \{0,\dots,d-2\}$, $t \in (0,\infty)$, and $(-1)^{d-2}\frac{d^{d-2}}{dt^{d-2}}\psi(t)$ is decreasing and convex on $(0,\infty)$. Note that some authors refer to $\varphi := \psi^{-1}$ as generator of the Archimedean copula (1), see e.g. Nelsen (2007, p. 112). The reason why we prefer to work with ψ is simply notational convenience, as we are more interested in the function ψ than its inverse. This notation may be found e.g. in Joe (1997, p. 86) and McNeil and Nešlehová (2009).

Throughout this work we assume ψ to be *completely monotone*, i.e. ψ is continuous on $[0,\infty]$ and $(-1)^k \frac{d^k}{dt^k}\psi(t) \geq 0$ for all $k \in \mathbb{N}_0$, $t \in (0,\infty)$, so that ψ is the Laplace-Stieltjes transform of a distribution function F on the positive real line, i.e. $\psi = \mathcal{LS}[F]$, see Bernstein's Theorem in Feller (1971, p. 439). The class of all such generators is denoted by Ψ_∞. If we know how to efficiently sample $F = \mathcal{LS}^{-1}[\psi]$, the following algorithm by Marshall and Olkin (1988) easily generates vectors of random variates following the exchangeable Archimedean copula generated by ψ in all dimensions d.

Algorithm 2.1 (Marshall and Olkin (1988))

(1) Sample $V \sim F = \mathcal{LS}^{-1}[\psi]$.

(2) Sample i.i.d. $X_i \sim U[0,1]$, $i \in \{1,\dots,d\}$.

(3) Return $(U_1,\dots,U_d)$, where $U_i = \psi((-\log X_i)/V)$, $i \in \{1,\dots,d\}$.

Fully-nested Archimedean copulas can be recursively defined via

$$C(u_1,\dots,u_d;\psi_0,\dots,\psi_{d-2}) = \psi_0(\psi_0^{-1}(u_1) + \psi_0^{-1}(C(u_2,\dots,u_d;\psi_1,\dots,\psi_{d-2}))) \quad (2)$$

for all $d \geq 3$, $u_i \in [0,1]$, $i \in \{1,\dots,d\}$. For $d = 3$, this corresponds to

$$C(u_1, C(u_2,u_3;\psi_1);\psi_0) = \psi_0(\psi_0^{-1}(u_1) + \psi_0^{-1}(\psi_1(\psi_1^{-1}(u_2) + \psi_1^{-1}(u_3)))). \quad (3)$$

Partially-nested Archimedean copulas combine structures (1) and (2) and the resulting structures are naturally motivated by applications involving different levels of dependence. Fully- and partially-nested Archimedean copulas are summarized as *nested* (or *hierarchical*) *Archimedean copulas*. Note that the trivariate copula (3) first appeared in Joe (1997, p. 87), although Joe and Hu (1996) already deal with even more general mixture distributions.

According to McNeil (2008), a sufficient condition for a nested Archimedean copula being indeed a proper copula is that all nodes of the form $\psi_i^{-1} \circ \psi_j$ for all i,j appearing in the Archimedean structure have completely monotone derivatives, e.g. for (2) see the following result.

Theorem 2.2 (McNeil (2008))
Let $\psi_i \in \Psi_\infty$ for $i \in \{0,\dots,d-2\}$ such that $\psi_k^{-1} \circ \psi_{k+1}$ have completely monotone derivatives for all $k \in \{0,\dots,d-3\}$, then $C(u_1,\dots,u_d;\psi_0,\dots,\psi_{d-2})$ is a copula.

The algorithms for sampling nested Archimedean copulas as presented by McNeil (2008) are similar to the algorithm of Marshall and Olkin (1988) for sampling exchangeable Archimedean copulas. However, in addition to sampling the distribution corresponding to the outermost generator, they also involve sampling the distributions corresponding to generators of the form $\psi_{i,j}(t;V) = \exp(-V\psi_i^{-1} \circ \psi_j(t))$, where V is a given random variate of the distribution function F from an earlier step, or outer structure, and i and j refer to the node under consideration. This is due to the mixture representation for these copulas, see McNeil (2008) for more details. The recursive algorithm for sampling (2) is given as follows. Note that sampling a partially-nested Archimedean copula works similarly, see McNeil (2008).

Algorithm 2.3 (McNeil (2008))
(1) Sample $V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0]$.

(2) Sample $X_1 \sim U[0,1]$.

(3) Sample $(X_2,\dots,X_d) \sim C(u_2,\dots,u_d;\psi_{0,1}(\cdot;V_0),\dots,\psi_{0,d-2}(\cdot;V_0))$.

(4) Return $(U_1,\dots,U_d)$, where $U_i = \psi_0((-\log X_i)/V_0)$, $i \in \{1,\dots,d\}$.

As the principles underlying our sampling strategies already become clear by considering fully-nested Archimedean copulas of type (3), we exemplarily study this subclass of copulas

in the sequel and particularly develop efficient sampling algorithms for the distributions corresponding to a generic node of the form

$$\psi_{0,1}(t; V_0) = \exp(-V_0 \psi_0^{-1} \circ \psi_1(t)), \ t \in [0, \infty]. \tag{4}$$

However, we specifically emphasize that all presented ideas are applicable to different kinds of nested Archimedean copulas, involving several levels of nesting, even in large dimensions. For an example involving three levels of nesting, see Section 4.5. Note that V_0 almost surely takes values in $(0, \infty)$ since $V_0 = 0$ with positive probability would imply $\psi_0(\infty) > 0$ and $V_0 = \infty$ with positive probability would imply $\psi_0(0) < 1$.

Table 1 lists commonly used generators and conditions on the parameters involved such that nested Archimedean copulas result, including those which generate nested Ali-Mikhail-Haq (A), Clayton (C), Frank (F), Gumbel (G), and Joe (J) copulas, see Nelsen (2007, pp. 116).

Family	ϑ_i	$\psi_i(t)$	$(\psi_0^{-1} \circ \psi_1)'$ c.m.
A	$[0, 1)$	$(1 - \vartheta_i)/(\exp(t) - \vartheta_i)$	$\vartheta_0, \vartheta_1 \in [0, 1) : \vartheta_0 \leq \vartheta_1$
C	$(0, \infty)$	$(1 + t)^{-1/\vartheta_i}$	$\vartheta_0, \vartheta_1 \in (0, \infty) : \vartheta_0 \leq \vartheta_1$
F	$(0, \infty)$	$-(\log(e^{-t}(e^{-\vartheta_i} - 1) + 1))/\vartheta_i$	$\vartheta_0, \vartheta_1 \in [1, \infty) : \vartheta_0 \leq \vartheta_1$
G	$[1, \infty)$	$\exp(-t^{1/\vartheta_i})$	$\vartheta_0, \vartheta_1 \in (0, \infty) : \vartheta_0 \leq \vartheta_1$
J	$[1, \infty)$	$1 - (1 - \exp(-t))^{1/\vartheta_i}$	$\vartheta_0, \vartheta_1 \in [1, \infty) : \vartheta_0 \leq \vartheta_1$
12	$[1, \infty)$	$(1 + t^{1/\vartheta_i})^{-1}$	$\vartheta_0, \vartheta_1 \in [1, \infty) : \vartheta_0 \leq \vartheta_1$
13	$[1, \infty)$	$\exp(1 - (1 + t)^{1/\vartheta_i})$	$\vartheta_0, \vartheta_1 \in [1, \infty) : \vartheta_0 \leq \vartheta_1$
14	$[1, \infty)$	$(1 + t^{1/\vartheta_i})^{-\vartheta_i}$	$\vartheta_0, \vartheta_1 \in [1, \infty) : \vartheta_0, \vartheta_1/\vartheta_0 \in \mathbb{N}$
19	$(0, \infty)$	$\vartheta_i / \log(t + \exp(\vartheta_i))$	$\vartheta_0, \vartheta_1 \in (0, \infty) : \vartheta_0 \leq \vartheta_1$
20	$(0, \infty)$	$(\log(t + e))^{-1/\vartheta_i}$	$\vartheta_0, \vartheta_1 \in (0, \infty) : \vartheta_0 \leq \vartheta_1$

Table 1 Completely monotone Archimedean generators of Nelsen (2007, pp. 116) with corresponding parameter ranges.

Note that the class of distributions of type $F_{0,1} = \mathcal{LS}^{-1}[\psi_{0,1}(\cdot; V_0)]$ precisely coincides with the class of infinitely divisible distributions, see Feller (1971, p. 450). Approximate sampling methods for such distributions when the Lévy measure associated with the Laplace-Stieltjes transform is known were already presented in Bondesson (1982) and Damien et al. (1995). In our case, we aim for fast and exact algorithms. The sampling problems we consider possibly extend to related sampling problems for Archimedean generators not discussed here or for other distributions known by their Laplace-Stieltjes transforms in general.

The main problem in sampling distributions of type $F_{0,1}$ is that they depend on V_0, a random variate following F_0. At first glance, V_0 can be seen as an additional parameter of $F_{0,1}$. However, it usually changes for every vector of random variates to be drawn from the copula. For this reason, efficiently sampling $F_{0,1}$ is usually considerably more complicated than sampling F_0. In the remaining part of the paper, we illustrate and solve this problem by developing sampling algorithms which isolate $F_{0,1}$ from the influence of

V_0. This will be the main key to efficient sampling strategies for nested Archimedean copulas.

3 Efficiently sampling Archimedean copulas

3.1 The nested Archimedean families of Ali-Mikhail-Haq, Frank, and Joe

Assuming F_0 and $F_{0,1}$ to be discrete, e.g. as for the families of Ali-Mikhail-Haq, Frank, and Joe, one might be tempted to precalculate and store values of $F_{0,1}$ for different choices of V_0, see Hofert (2008). However, this procedure can not be recommended for all families involving discrete distributions as it depends on the number of different values V_0 is expected to take. If $\mathbb{E}[V_0] = \infty$, which is the case e.g. for the Archimedean family of Joe, one therefore has to expect different and rather large random variates V_0 in a computational implementation. Further, it may be numerically challenging to evaluate the jump heights of $F_{0,1}$ as they depend on V_0. For example, the jump heights of $F_{0,1}$ for a nested Joe copula are given by sums of numerically challenging products of binomial coefficients, see Hofert (2008). One can show that the first jump of $F_{0,1}$ occurs at V_0 with height $y_{V_0} := \alpha^{V_0}$, $\alpha := \vartheta_0/\vartheta_1$, and numerical experiments further indicate that the computation of the jump heights becomes especially demanding if V_0 is large or α is small. To significantly reduce this problem, consider the following result, see Devroye (1986, p. 487).

Lemma 3.1
Let g, g_0, and g_1 be generating functions such that $g(t) = g_0(g_1(t))$. Then $\sum_{i=1}^N X_i \sim g$, where $N \sim g_0$ and X_i are i.i.d. according to g_1, $i \in \{1, \dots, N\}$.

Lemma 3.1 allows to decompose $F_{0,1}$ into a part which can be precalculated and a sum where each summand is sampled according to the precalculated distribution. The following theorem relies on this idea and presents sampling strategies for the nested Archimedean families of Ali-Mikhail-Haq, Frank, and Joe. It is important to note that a random variate V_0 of F_0 only determines the number of summands to sample; however, the precalculated distribution does not depend on V_0 anymore. For the family of Ali-Mikhail-Haq, this distribution is a Geometric distribution and therefore does not even have to be precalculated and stored for efficient sampling.

Theorem 3.2
(1) For the family of Ali-Mikhail-Haq, F_0 is a $\text{Geo}(1-\vartheta_0)$, i.e. a Geometric, distribution on $\mathbb{N}$. Further, $F_{0,1}$ is also discrete and can be sampled via the following algorithm, where V_0 denotes a random variate drawn from F_0.

(1.1) Sample i.i.d. $V_{0,1,i} \sim \text{Geo}((1-\vartheta_1)/(1-\vartheta_0))$, $i \in \{1, \dots, V_0\}$.

(1.2) Return $V_{0,1} = \sum_{i=1}^{V_0} V_{0,1,i}$.

(2) For the family of Frank, F_0 is a $\text{Log}(1-\exp(-\vartheta_0))$, i.e. a Logarithmic, distribution. Further, $F_{0,1}$ is also discrete and can be sampled via the following algorithm, where V_0 again denotes a random variate drawn from F_0.

(2.1) Sample i.i.d. $V_{0,1,i}$, $i \in \{1, \dots, V_0\}$, with discrete probability density given by $y_k = \binom{\vartheta_0/\vartheta_1}{k}(-1)^{k-1}(1-\exp(-\vartheta_1))^k/(1-\exp(-\vartheta_0))$ at $k \in \mathbb{N}$.

(2.2) Return $V_{0,1} = \sum_{i=1}^{V_0} V_{0,1,i}$.

(3) For the family of Joe, F_0 has discrete probability density given by $y_k = \binom{1/\vartheta_0}{k}(-1)^{k-1}$ at $k \in \mathbb{N}$. Further, $F_{0,1}$ is also discrete and can be sampled via the following algorithm, where V_0 denotes a random variate drawn from F_0 as before.

(3.1) Sample i.i.d. $V_{0,1,i}$, $i \in \{1, \dots, V_0\}$, with discrete probability density given by $y_k = \binom{\vartheta_0/\vartheta_1}{k}(-1)^{k-1}$ at $k \in \mathbb{N}$.

(3.2) Return $V_{0,1} = \sum_{i=1}^{V_0} V_{0,1,i}$.

Proof

First consider Part (1). For the statement about F_0, see Hofert (2008). For the part concerning $F_{0,1}$, we compute the generating function g of $F_{0,1}$ and show that it can be written as a composition of two generating functions g_0 and g_1. The result will then follow from Lemma 3.1. Hofert (2008) showed that $F_{0,1} = \sum_{k=V_0}^{\infty} y_k \mathbb{1}_{[x_k,\infty)}(x)$ with

$$x_k = k,\ y_k = \frac{c_1^{k-V_0}}{c_0^k}\binom{k-1}{k-V_0},\ k \in \mathbb{N}\backslash\{1, \dots, V_0 - 1\},$$

where $c_0 = (1-\vartheta_0)/(1-\vartheta_1)$ and $c_1 = (\vartheta_1 - \vartheta_0)/(1-\vartheta_1)$. Applying the identity $\binom{k+V_0-1}{k} = \binom{-V_0}{k}(-1)^k$ and the Binomial Series Theorem therefore leads to the corresponding generating function g, given by

$$\begin{aligned} g(t) &= \sum_{k=V_0}^{\infty} \frac{c_1^{k-V_0}}{c_0^k}\binom{k-1}{k-V_0}t^k = \left(\frac{t}{c_0}\right)^{V_0} \sum_{k=0}^{\infty}\binom{k+V_0-1}{k}\left(\frac{c_1}{c_0}t\right)^k \\ &= \left(\frac{t}{c_0}\right)^{V_0} \sum_{k=0}^{\infty}\binom{-V_0}{k}\left(-\frac{c_1}{c_0}t\right)^k = \left(\frac{t}{c_0}\right)^{V_0}\left(1-\frac{c_1}{c_0}t\right)^{-V_0} = \left(\frac{t}{c_0 - c_1 t}\right)^{V_0}. \end{aligned}$$

Hence, $g(t) = g_0(g_1(t))$ with $g_0(t) = t^{V_0}$ and $g_1(t) = t/(c_0 - c_1 t)$. As g_1 corresponds to a $\mathrm{Geo}((1-\vartheta_1)/(1-\vartheta_0))$ distribution, the result follows from Lemma 3.1. For the statements about F_0 for Part (2) and (3), see Joe (1997, p. 375). The results about $F_{0,1}$ can be obtained similarly as before by choosing $g_1(t) = (1-(1-c_1 x)^{\alpha})/c_0$, $c_i = 1-\exp(-\vartheta_i)$, $i \in \{1, 2\}$, and $g_1(t) = 1-(1-x)^{\alpha}$ for the families of Frank and Joe, respectively. □

Remark 3.3

(1) Sampling a random variate $X \sim \mathrm{Geo}(p)$, as required for the Archimedean family of Ali-Mikhail-Haq, can be achieved by sampling $X \sim \mathrm{Exp}(-\log(1-p))$, i.e. an Exponential distribution, and returning $\lceil X \rceil$, where $\lceil \cdot \rceil$ denotes the ceil function, see Devroye (1986, p. 499).

(2) For sampling the discrete distributions for the families of Frank and Joe, one can precalculate and store the sums of the jump heights (y_k) of the distributions that do not depend on V_0 anymore up to $1-\varepsilon$ for some sufficiently small $\varepsilon > 0$. For most

parameter choices the corresponding algorithms are very fast even if a large number of summands has to be sampled to obtain a random variate $V_{0,1}$. For finding the U-quantile of a $U \sim U[0,1]$ with $U > 1-\varepsilon$, one may either use truncation, see Hofert (2008), or asymptotics in the tail, see Feller (1971, pp. 442). Note that for a specific parameter choice, the behavior of the precomputed distribution function involved can now be studied independent of V_0.

3.2 Exponential tilting and a general nesting result

Starting with any generator $\psi \in \Psi_\infty$ with $F = \mathcal{LS}^{-1}[\psi]$ and given any $h \in [0,\infty)$, the function $\tilde{\psi}(t) = \psi(t+h)/\psi(h)$ also defines an Archimedean generator in Ψ_∞. The inverse Laplace-Stieltjes transform $\tilde{F}$ of the *tilted Archimedean generator* $\tilde{\psi}$ is addressed in Hofert (2008). If F admits a density f, then $\tilde{F}$ admits the *exponentially-tilted density* $\tilde{f}(x) = \exp(-hx)f(x)/\psi(h)$, $x \in [0,\infty)$. If F is a Stable distribution of type $\mathrm{S}(1/\vartheta, 1, \cos^\vartheta(\pi/(2\vartheta)), 0; 1)$ with Laplace-Stieltjes transform $\psi(t) = \exp(-t^{1/\vartheta})$, see Nolan (2009, p. 8) for the parametrization, we denote the corresponding *exponentially-tilted Stable distribution* by $\tilde{\mathrm{S}}(1/\vartheta, 1, \cos^\vartheta(\pi/(2\vartheta)), 0, h; 1)$, where h is the tilt involved.

Starting with any generator $\psi \in \Psi_\infty$ and building $\psi_i(t) = \psi((c^{\vartheta_i}+t)^{1/\vartheta_i} - c)$, $\vartheta_i \in [1,\infty)$, $i \in \{0,1\}$, and $c \in [0,\infty)$, Hofert (2008) showed that ψ_0 and ψ_1 are generators in Ψ_∞ which, if $\vartheta_0 \le \vartheta_1$, can also be mixed to build nested Archimedean copulas. Further, the corresponding distribution function $F_{0,1}$ is an exponentially-tilted Stable distribution. The following result shows how the corresponding distribution F_0 may be sampled.

Theorem 3.4
Let $\psi \in \Psi_\infty$. If $\tilde{\psi}(t) = \psi((c^\vartheta + t)^{1/\vartheta} - c)$, $t \in [0,\infty]$, with $\vartheta \in [1,\infty)$ and $c \in [0,\infty)$, then

$$\tilde{V} = \tilde{S}V^\vartheta \sim \tilde{F} = \mathcal{LS}^{-1}[\tilde{\psi}],$$

where $V \sim F = \mathcal{LS}^{-1}[\psi]$ and $\tilde{S} \sim \tilde{\mathrm{S}}(1/\vartheta, 1, \cos^\vartheta(\pi/(2\vartheta)), 0, (cV)^\vartheta; 1)$.

Proof
We show that the Laplace-Stieltjes transform of the distribution function $F_{\tilde{S}V^\vartheta}$ of $\tilde{S}V^\vartheta$ equals $\tilde{\psi}$. Let f_S denote the density of a $\mathrm{S}(1/\vartheta, 1, \cos^\vartheta(\pi/(2\vartheta)), 0; 1)$ distribution with Laplace-Stieltjes transform $\psi_S(t) = \exp(-t^{1/\vartheta})$ and $f_{\tilde{S}|V}$ the conditional density of $\tilde{S}$ given $V = v$ which is $f_{\tilde{S}|V}(x|v) = \exp(-hx)f_S(x)/\psi_S(h)$ with $h = (cv)^\vartheta$. Then

$$\begin{aligned}\mathcal{LS}[F_{\tilde{S}V^\vartheta}](t) &= \int_0^\infty \int_0^\infty \exp(-txv^\vartheta) f_{\tilde{S}|V}(x|v)\,dx\,dF(v) \\ &= \int_0^\infty \exp(cv) \int_0^\infty \exp(-txv^\vartheta)\exp(-(cv)^\vartheta x) f_S(x)\,dx\,dF(v).\end{aligned}$$

Using $\exp(-txv^\vartheta)\exp(-(cv)^\vartheta x) = \exp(-xv^\vartheta(c^\vartheta+t))$ and $\int_0^\infty \exp(-xv^\vartheta(c^\vartheta+t))f_S(x)\,dx = \psi_S(v^\vartheta(c^\vartheta+t))$, we obtain

$$\begin{aligned}\mathcal{LS}[F_{\tilde{S}V^\vartheta}](t) &= \int_0^\infty \exp(cv)\exp(-(v^\vartheta(c^\vartheta+t))^{1/\vartheta})\,dF(v) \\ &= \int_0^\infty \exp(-v((c^\vartheta+t)^{1/\vartheta} - c))\,dF(v) = \psi((c^\vartheta+t)^{1/\vartheta} - c) = \tilde{\psi}(t)\end{aligned}$$

for all $t \in [0, \infty]$, hence $\tilde{V} \sim \tilde{F}$. □

Theorem 3.4 and Theorem 8 in Hofert (2008) allow to start from any Archimedean generator $\psi \in \Psi_\infty$ and build a nested Archimedean copula based on the generator $\psi((c^\vartheta + t)^{1/\vartheta} - c)$. Further, if we know how to sample $F = \mathcal{LS}^{-1}[\psi]$, we do so for the resulting nested Archimedean copula. The recursive algorithm for sampling the fully-nested Archimedean copula (2) is given as follows. Note that e.g. all *outer power Archimedean copulas*, see Hofert (2008), easily follow from this result by taking $c = 0$. Moreover, nested Clayton copulas arise by taking $\psi(t) = (1 + t)^{-1}$ and $c = 1$.

Algorithm 3.5
Let $\psi \in \Psi_\infty$, $c \in [0, \infty)$, $\psi_i(t) = \psi((c^{\vartheta_i} + t)^{1/\vartheta_i} - c)$, $i \in \{0, \dots, d-2\}$, $d \in \mathbb{N}\backslash\{1\}$, with $1 \le \vartheta_0 \le \cdots \le \vartheta_{d-2} < \infty$.

(1) Sample $V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0]$ via $V_0 = \tilde{S}V^{\vartheta_0}$, where $V \sim F = \mathcal{LS}^{-1}[\psi]$, $\tilde{S} \sim \tilde{\mathrm{S}}(1/\vartheta_0, 1, \cos^{\vartheta_0}(\pi/(2\vartheta_0)), 0, (cV)^{\vartheta_0}; 1)$.

(2) Sample $X_1 \sim U[0, 1]$.

(3) Sample $(X_2, \dots, X_d) \sim C(u_2, \dots, u_d; \psi_{0,1}(\cdot; V_0), \dots, \psi_{0,d-2}(\cdot; V_0))$, where for $i \in \{1, \dots, d-2\}$, $\psi_{0,i}(t; V_0) = \exp(-V_0((h_i + t)^{\alpha_i} - h_i^{\alpha_i}))$ with $h_i = c^{\vartheta_i}$ and $\alpha_i = \vartheta_0/\vartheta_i$ corresponds to an $\tilde{\mathrm{S}}(\alpha_i, 1, (\cos(\pi\alpha_i/2)V_0)^{1/\alpha_i}, 0, h_i; 1)$ distribution.

(4) Return $(U_1, \dots, U_d)$, where $U_i = \psi_0((-\log X_i)/V_0)$, $i \in \{1, \dots, d\}$.

Sampling an exponentially-tilted density $\tilde{f}(x) = \exp(-hx)f(x)/\psi(h)$ is straightforward in certain cases, e.g. when f is a Gamma density. However, if f is the density of a Stable distribution, it is quite difficult. A general algorithm for sampling the density $\tilde{f}$ might be the following rejection algorithm, see Devroye (1986, pp. 40), with $f(x)/\psi(h)$ as envelope if f is easy to sample.

Algorithm 3.6 (Standard rejection)
Repeatedly sample $\tilde{V} \sim f$ and $U \sim U[0, 1]$ until $U \le \exp(-h\tilde{V})$, then return $\tilde{V}$.

The expected number of iterations in the standard rejection algorithm is $1/\psi(h)$, see Devroye (1986, p. 42), i.e. the algorithm has complexity $\mathcal{O}(1/\psi(h))$ assuming constant complexity for sampling f. This may slow down the algorithm considerably, making it inapplicable for certain parameter ranges.

To reduce the complexity of the rejection algorithm, note that a product of Laplace-Stieltjes transforms corresponds to a convolution of the underlying distributions. We therefore aim at writing $\tilde{\psi}(t) = \psi(t + h)/\psi(h)$ as an m-fold product of its $1/m$-th powers, which can then be sampled as a sum of i.i.d. random variables. By choosing this product in such a way that the complexity of the corresponding sampling algorithm is minimized, we obtain the following result.

Theorem 3.7
Let $\psi \in \Psi_\infty$ such that $\psi^{1/m} \in \Psi_\infty$ for all $m \in \mathbb{N}$ and let $\tilde{\psi}(t) = \psi(t+h)/\psi(h)$, $h \in [0, \infty)$.

(1) $\tilde{V} \sim \tilde{F} = \mathcal{LS}^{-1}[\tilde{\psi}]$ can be sampled as $\tilde{V} = \sum_{i=1}^m \tilde{V}_i$, with i.i.d. $\tilde{V}_i \sim \tilde{F}_m = \mathcal{LS}^{-1}[\tilde{\psi}^{1/m}]$, $i \in \{1, \dots, m\}$, and $m \in \mathbb{N}$.

(2) Let $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the floor and ceil function, respectively, and let $c := 1/\psi(h)$. By choosing

$$m = \begin{cases} 1 & \text{if } \log c \leq 1 \\ \lfloor \log c \rfloor & \text{if } \log c > 1,\ \lfloor \log c \rfloor c^{1/\lfloor \log c \rfloor} \leq \lceil \log c \rceil c^{1/\lceil \log c \rceil} \\ \lceil \log c \rceil & \text{if } \log c > 1,\ \lfloor \log c \rfloor c^{1/\lfloor \log c \rfloor} > \lceil \log c \rceil c^{1/\lceil \log c \rceil} \end{cases} \tag{5}$$

in Part (1) and applying the standard rejection algorithm for sampling each $\tilde{V}_i$, $i \in \{1, \ldots, m\}$, the resulting expected number of iterations for sampling $\tilde{V}$ is less than or equal to $1/\psi(h)$. Further, if $\log c > 1$, this strategy for sampling $\tilde{F}$ has complexity $\mathcal{O}(\log(1/\psi(h)))$.

Proof

Part (1) is clear. For Part (2), note that sampling m-times $\tilde{F}_m$ via the standard rejection algorithm takes expected number of iterations $mc^{1/m}$. For determining the choice of $m \in \mathbb{N}$ that minimizes this number, consider the function $g(x) = xc^{1/x}$ on $(0, \infty)$. Since g has its global minimum at $\log c$ and is convex, the optimal $m \in \mathbb{N}$ is either $m = \lfloor \log c \rfloor$ or $m = \lceil \log c \rceil$, chosen such that the expected number of iterations is minimized, with $m = 1$ if $\log c \leq 1$. If $\log c \leq 1$, the resulting expected number of iterations is simply $c = 1/\psi(h)$, as for the standard rejection algorithm. If $\log c > 1$, i.e. $c \in (e, \infty)$, it is $\min\{\lfloor \log c \rfloor c^{1/\lfloor \log c \rfloor}, \lceil \log c \rceil c^{1/\lceil \log c \rceil}\}$ and therefore less than or equal to $\lfloor \log c \rfloor c^{1/\lfloor \log c \rfloor}$. If $c \in (e, e^2)$, this equals $c = 1/\psi(h)$ and if $c \in [e^2, \infty)$, this is less than or equal to $\sqrt{c} \log c < c = 1/\psi(h)$; hence, the expected number of iterations for sampling $\tilde{V}$ is less than or equal to $1/\psi(h)$. Further, if $c \in (e, \infty)$, the upper bound $\lfloor \log c \rfloor c^{1/\lfloor \log c \rfloor}$ for the expected number of iterations of the fast rejection algorithm is bounded above by $c^{1/\lfloor \log c \rfloor} \log c$. With $c^{1/\lfloor \log c \rfloor} = \exp((\log c)/\lfloor \log c \rfloor) \leq \exp((\lfloor \log c \rfloor + 1)/\lfloor \log c \rfloor) \leq e^2$ it follows that the expected number of iterations of the fast rejection algorithm is bounded above by $e^2 \log c$, hence the complexity $\mathcal{O}(\log(1/\psi(h)))$ is proven. □

Note that $\psi^{1/m}$ being in Ψ_∞ for all $m \in \mathbb{N}$ is equivalent to $F = \mathcal{LS}^{-1}[\psi]$ being infinitely divisible, see Feller (1971, p. 450). This is a rather weak assumption, e.g. all generators listed in Table 1 share this property. The fast rejection algorithm for sampling the distribution $\tilde{F} = \mathcal{LS}^{-1}[\tilde{\psi}]$ as in Theorem 3.7 is given as follows. Note that the complexity $\mathcal{O}(1/\psi(h))$ of the standard rejection algorithm is reduced to $\mathcal{O}(\log(1/\psi(h)))$ for any ψ and h such that $-\log \psi(h) > 1$.

Algorithm 3.8 (Fast rejection)

Let $\psi \in \Psi_\infty$ such that $\psi^{1/m} \in \Psi_\infty$ for all $m \in \mathbb{N}$ and let $\tilde{\psi}(t) = \psi(t+h)/\psi(h)$, $h \in [0, \infty)$. For m as in Equation (5), sample i.i.d. $\tilde{V}_i \sim \tilde{F}_m = \mathcal{LS}^{-1}[\tilde{\psi}^{1/m}]$, $i \in \{1, \ldots, m\}$, via the standard rejection algorithm and return $\tilde{V} = \sum_{i=1}^m \tilde{V}_i$.

In the following, we exemplarily consider generating random variates of an exponentially-tilted Stable distributed random variable $\tilde{S} \sim \tilde{\mathrm{S}}(\alpha, 1, (\cos(\pi\alpha/2)V_0)^{1/\alpha}, 0, h; 1)$ with corresponding Laplace-Stieltjes transform $\tilde{\psi}(t) = \exp(-V_0((h+t)^\alpha - h^\alpha))$. Sampling such random variates is required e.g. in Step (1) and Step (3) of Algorithm 3.5 for sampling a nested Archimedean copula based on the generators addressed in Theorem 3.4.

Exponentially-tilted Stable distributions first appeared in Tweedie (1984) and Hougaard (1986). They are also considered by Brix (1999). For applications, see e.g. McCulloch (2003) and McCulloch and Lee (2007) in the context of option pricing. For a more recent treatment, see Barndorff-Nielson and Shephard (2001), who also deal with sampling algorithms for these distributions. Their proposed method originates from Rosiński (2001). This algorithm is also presented in Schoutens (2003). Rosiński (2007) dedicated a whole paper to exponentially-tilted Stable distributions. Ridout (2008) examines sampling algorithms based on numerical inversion of Laplace transforms and also considers exponentially-tilted Stable distributions. His study includes the algorithm of Rosiński (2007). In what follows, we shortly recall the known approaches for sampling exponentially-tilted Stable distributions. In Section 4 we compare these algorithms with the rejection and the fast rejection algorithm for sampling nested Clayton copulas.

Rosiński (2007) considers series representations of Lévy processes, including those with exponentially-tilted Stable distributed increments, and derives approximations for simulations. For sampling $\tilde{S}' \sim \tilde{\mathrm{S}}(\alpha, 1, (\cos(\pi\alpha/2)V_0)^{1/\alpha}h, 0, 1; 1)$ with corresponding Laplace-Stieltjes transform $\exp(-V_0 h^\alpha((1+t)^\alpha - 1))$, Rosiński (2007) derives the representation

$$\tilde{S}' = \sum_{j=1}^{\infty} \min\Big\{\Big(\frac{V_0}{\Gamma(1-\alpha)\gamma_j}\Big)^{1/\alpha} h, e_j u_j^{1/\alpha}\Big\}, \tag{6}$$

where (u_j) is a sequence of i.i.d. $U[0,1]$ random variables, (e_j), $(\tilde{e}_j)$ are sequences of i.i.d. Exp(1) random variables (all sequences are assumed to be independent), and $\gamma_j = \sum_{i=1}^{j} \tilde{e}_i$, $j \in \mathbb{N}$. The random variable $\tilde{S} = \tilde{S}'/h$ is then $\tilde{\mathrm{S}}(\alpha, 1, (\cos(\pi\alpha/2)V_0)^{1/\alpha}, 0, h; 1)$ distributed. Rosiński (2007) also obtains a representation for a Lévy process $(\tilde{S}'_t)_{t\in[0,T]}$, $T > 0$, with exponentially-tilted Stable distributed increments, given by

$$\tilde{S}'_t = \sum_{j=1}^{\infty} \min\Big\{\Big(\frac{TV_0}{\Gamma(1-\alpha)\gamma_j}\Big)^{1/\alpha} h, e_j u_j^{1/\alpha}\Big\} \mathbb{1}_{(0,t/T]}(\tilde{u}_j), \tag{7}$$

where $(\tilde{u}_j)$ is a sequence of i.i.d. $U[0,1]$ random variables, independent of the other sequences of random variables. Increments of unit length are then i.i.d. according to $\tilde{\mathrm{S}}(\alpha, 1, (\cos(\pi\alpha/2)V_0)^{1/\alpha}h, 0, 1; 1)$. The resulting algorithm for sampling n approximately $\tilde{\mathrm{S}}(\alpha, 1, (\cos(\pi\alpha/2)V_0)^{1/\alpha}, 0, h; 1)$ distributed random variates is given as follows, see also Ridout (2008).

Algorithm 3.9 (Rosiński (2007))

(1) Choose a truncation point $J \in \mathbb{N}$ for the sum in (7) and the number n of random variates to be generated.

(2) Generate independent sequences $(u_j)_{j=1}^J$, $(\tilde{u}_j)_{j=1}^J$, $(e_j)_{j=1}^J$, and $(\tilde{e}_j)_{j=1}^J$, with i.i.d. $u_j, \tilde{u}_j \sim U[0,1]$ and $e_j, \tilde{e}_j \sim \mathrm{Exp}(1)$ for all $j \in \{1, \ldots, J\}$.

(3) For $t \in \{0, \ldots, n\}$, compute $\tilde{S}'_t$ as in (7), where the sum is truncated at J and $\tilde{S}'_0 = 0$.

(4) Return $(S_1, \ldots, S_n)$, where $S_t = (\tilde{S}'_t - \tilde{S}'_{t-1})/h$, $t \in \{1, \ldots, n\}$.

Ridout (2008) considers numerical inversion of Laplace transforms for sampling purposes, including exponentially-tilted Stable distributions. For efficiently finding quantiles, he first sorts the generated uniform random variates. He compares his numerical algorithm, called `rlaptrans`, implemented in the statistical software R, to the algorithm of Rosiński (2007) for the parameters $(\alpha, V_0, h) \in \{(0.5, 1/\sqrt{2}, 0.5), (0.75, 4^{3/4}/3, 0.25)\}$, investigating also different choices of truncation points $J \in \{10\,000, 25\,000, 50\,000, 100\,000\}$. Ridout (2008) notes that the algorithm of Rosiński (2007) turns out to be slower (even for $J = 10\,000$) and biased (even for $J = 100\,000$) in comparison to his algorithm `rlaptrans`. He additionally investigates another numerical sampling algorithm, called `rtweedie`, based on a combination of a series representation and a Fourier inversion technique for the exponentially-tilted Stable density. Moreover, Ridout (2008) considers the rejection algorithm `rdevroye` based on an envelope found by using characteristic functions, see Devroye (1986, p. 696), adapted to the setup of nonnegative random variables. According to run time, Ridout (2008) notes that these methods can not compete with `rlaptrans` for the examined parameter choices.

The problem of sampling an exponentially-tilted Stable distribution is even more demanding in the case of sampling a generator $\psi_{0,1}$ involved in a nested Archimedean copula. The reason for this is that V_0 acts as an additional parameter on the exponentially-tilted Stable distribution which possibly changes with every vector of random variates to be generated. Hence, one has to sample vectors of random variates from possibly different distributions rather than from a distribution with fixed parameters. For example, for a nested Clayton copula, V_0 is a Gamma distributed random variable and hence almost surely changes for every vector of random variates to be generated. Algorithms requiring numerically demanding setup steps for sampling $F_{0,1}$ may therefore be inadequate for applications in large-scale simulation studies.

To summarize, all known algorithms for sampling exponentially-tilted Stable distributions appearing in nested Archimedean copulas are either inefficient or the numerical errors due to approximations are hard to control. However, the fast rejection algorithm solves this problem for a large range of parameters. Being originally of complexity $\mathcal{O}(\exp(V_0 h^\alpha))$, the standard rejection algorithm is outperformed by the fast rejection algorithm with complexity $\mathcal{O}(V_0 h^\alpha)$, i.e. linear in $V_0 h^\alpha$ instead of exponential, as long as $V_0 h^\alpha > 1$. Further, the fast rejection algorithm is exact and does not require time-consuming setup steps. In Section 4.3, we exemplarily compare the performance of the presented algorithms for sampling nested Clayton copulas.

Let us close this section with the remark that a similar result as the fast rejection algorithm can be obtained by writing $\tilde{\psi}(t) = \exp(-V_0((h+t)^\alpha - h^\alpha))$ as $\tilde{\psi}(t) = \psi_2(t)\psi_3^{\lfloor V_0 h^\alpha \rfloor}(t)$ with $\psi_2(t) = \exp^{V_0 h^\alpha - \lfloor V_0 h^\alpha \rfloor}(-((1 + t/h)^\alpha - 1))$ and $\psi_3(t) = \exp(-((1 + t/h)^\alpha - 1))$. One can therefore sample a random variable $\tilde{S} \sim \tilde{\mathrm{S}}(\alpha, 1, (\cos(\pi\alpha/2)V_0)^{1/\alpha}, 0, h; 1)$ as $\tilde{S} = \tilde{S}_2 + \sum_{i=1}^{\lfloor V_0 h^\alpha \rfloor} \tilde{S}_{3,i}$, where $\tilde{S}_2$ is a random variable with distribution corresponding to ψ_2, independent of $\tilde{S}_{3,i}$, $i \in \{1, \ldots, \lfloor V_0 h^\alpha \rfloor\}$, which are i.i.d. random variables following the distribution with Laplace-Stieltjes transform ψ_3. Note that the generators ψ_i, $i \in \{2, 3\}$, are of type $\tilde{\psi}_i(t/h)$. Therefore, the distribution corresponding to ψ_i can be sampled by sampling the one corresponding to $\tilde{\psi}_i(t)$ and dividing the generated random variates by

h. If we apply the standard rejection algorithm to sample the random variates involved, the complexity of this sampling strategy is given by $\exp(V_0h^\alpha - \lfloor V_0h^\alpha \rfloor) + \lfloor V_0h^\alpha \rfloor e$. By writing x for V_0h^α, we may investigate the performance of this algorithm for all $V_0 \in (0,\infty)$ and $h \in [0,\infty)$ by investigating the function $\exp(x - \lfloor x \rfloor) + \lfloor x \rfloor e$ for all $x \in [0,\infty)$. Recall that the complexity of the fast rejection algorithm for sampling the distribution corresponding to $\tilde{\psi}$ in terms of $x = V_0h^\alpha$ is given by $\exp(x)$ if $x \in [0,1]$ and $\min\{\lfloor x \rfloor \exp(x/\lfloor x \rfloor), \lceil x \rceil \exp(x/\lceil x \rceil)\}$ if $x \in (1,\infty)$. Comparing these two complexities as functions of $x \in [0,\infty)$ leads to a slightly worse complexity for the algorithm based on ψ_2 and ψ_3. However, ψ_3 is now independent of V_0. By considering $\tilde{\psi}_3$, it is even independent of h. Therefore, fast algorithms requiring setup steps for efficiently sampling the distribution corresponding to $\tilde{\psi}_3$ might be useful. Further, numerical inversion of Laplace transforms is another option since numerical inversion of $\tilde{\psi}_3$ is considerably less critical than that of $\tilde{\psi}$. This is due to the fact that the former only depends on the parameter α but neither on V_0 nor h. Hence, careful checks can be made to find optimal parameters for the numerical inversion procedure under consideration beforehand.

3.3 Randomizing generator parameters and mixing different Archimedean families

Not every combination of generators $\psi_0, \psi_1 \in \Psi_\infty$ is known to lead to a valid nested Archimedean copula as the sufficient nesting condition in Theorem 2.2 does not always hold, see Hofert (2008). However, by choosing ψ_1 as

$$\psi_1(t) := \psi_0(-\log \psi(t)) \tag{8}$$

for some generator ψ such that $\psi^\alpha \in \Psi_\infty$ for all $\alpha \in (0,\infty)$, ψ_0 and ψ_1 do always fulfill the sufficient nesting condition addressed in Theorem 2.2. To see that $\psi_1 \in \Psi_\infty$ and $\psi_0^{-1} \circ \psi_1$ has completely monotone derivative, it suffices to show that $-\log \psi$ has completely monotone derivative, see Feller (1971, p. 441), which in fact holds if and only if $\psi^\alpha \in \Psi_\infty$ for all $\alpha \in (0,\infty)$, see Joe (1997, p. 374). Note that all generators listed in Table 1 share this property.

The choice (8) for ψ_1 implies that (4) takes the form

$$\psi_{0,1}(t; V_0) = \psi^{V_0}(t),$$

which means that ψ_1 is chosen such that $\psi_{0,1}(t; V_0)$ is simply a power in ψ. Starting with a generator ψ such that $\psi^\alpha \in \Psi_\infty$ for all $\alpha \in (0,\infty)$ and whose powers correspond to distributions that are easy to sample, one may thus build and easily sample the corresponding nested Archimedean copula. Further, by writing $\psi_1(t) = \psi_0(-\log \psi(t))$ as

$$\psi_1(t) = \int_0^\infty \psi^x(t)\, dF_0(x),$$

where $F_0 = \mathcal{LS}^{-1}[\psi_0]$, we see that ψ_1 is simply the generator ψ with randomized parameter x according to the mixing distribution F_0.

The following algorithm uses this construction principle to sample the distribution corresponding to ψ_1.

Algorithm 3.10

(1) Sample $V_0 \sim F_0 = \mathcal{LS}^{-1}[\psi_0]$.

(2) Sample and return $V_1 \sim \mathcal{LS}^{-1}[\psi^{V_0}]$.

By a similar reasoning one also sees that $\psi_1(t) := \psi_0(\tilde{\psi}^{-1}(\psi(t)))$ is a proper Archimedean generator for any two generators $\psi, \tilde{\psi} \in \Psi_\infty$ that can be mixed according to Theorem 2.2. Further note that ψ_1 defined this way also makes sense for all generators $\psi \in \Psi_\infty$ as long as $\psi_0 \circ \tilde{\psi}^{-1}$ is *absolutely monotone* on $[0,1]$, i.e. continuous on $[0,1]$ and differentiable of all orders with all derivatives greater than or equal to zero on $(0,1)$, see Feller (1971, p. 223).

Unfortunately, there is no general strategy known for sampling Step (2) of Algorithm 3.10 when the distribution corresponding to ψ is easy to sample. However, using the specific form of a given generator ψ, one usually finds an algorithm for sampling the distribution corresponding to ψ^{V_0}.

The idea of randomizing parameters will be applied in the sequel to sample nested Archimedean copulas constructed via generators belonging to different Archimedean families. We originally applied a numerical inversion procedure for Laplace transforms to sample a fully-nested (A,C) copula, i.e. an Ali-Mikhail-Haq copula and a Clayton copula for the outer and inner Archimedean copula, respectively, see Hofert (2008). With the help of tilted Archimedean generators and randomized parameters we are able to find explicit sampling algorithms for all family combinations listed in Table 2.

Family combination	ϑ_0	ϑ_1	$(\psi_0^{-1} \circ \psi_1)'(t)$ c.m.
(A,C)	$[0,1)$	$(0,\infty)$	$\vartheta_1 \in [1,\infty)$
(A,19)	$[0,1)$	$(0,\infty)$	any ϑ_0, ϑ_1
(A,20)	$[0,1)$	$(0,\infty)$	$\vartheta_1 \in [1,\infty)$
(C,12)	$(0,\infty)$	$[1,\infty)$	$\vartheta_0 \in (0,1]$
(C,14)	$(0,\infty)$	$[1,\infty)$	$\vartheta_0\vartheta_1 \in (0,1]$
(C,19)	$(0,\infty)$	$(0,\infty)$	$\vartheta_0 \in (0,1]$
(C,20)	$(0,\infty)$	$(0,\infty)$	$\vartheta_0 \le \vartheta_1$

Table 2 Proper family combinations and corresponding parameter ranges for the Archimedean families of Nelsen (2007, pp. 116).

Theorem 3.11

(1) For the family combination (A,C), F_0 is a $\mathrm{Geo}(1-\vartheta_0)$ distribution. Further, $V_{0,1} \sim F_{0,1}$ can be sampled as $V_{0,1} = \tilde{S}V^{\vartheta_1}$, where $V \sim \Gamma(V_0, 1/(1-\vartheta_0))$, i.e. a Gamma distribution with density $1/(1-\vartheta_0)^{V_0} x^{V_0-1} \exp(-x/(1-\vartheta_0))/\Gamma(V_0)$, $x \in [0,\infty)$, and $\tilde{S} \sim \tilde{\mathrm{S}}(1/\vartheta_1, 1, \cos^{\vartheta_1}(\pi/(2\vartheta_1)), 0, V^{\vartheta_1}; 1)$, which can be sampled via the fast rejection algorithm.

(2) For the family combination (A,19), F_0 is a $\mathrm{Geo}(1-\vartheta_0)$ distribution. Further, $V_{0,1} \sim F_{0,1}$ can be sampled via the fast rejection algorithm, since $\psi_{0,1}(t)$ is of the form

$\psi(t+h)/\psi(h)$, where $h = \exp(\vartheta_1) - 1$, $\psi(t) = \psi_2(-\log \psi_3(t))$, $\psi_2(t) = (1+t)^{-V_0}$, and $\psi_3(t) = (1+t)^{-(1-\vartheta_0)/(\vartheta_0\vartheta_1)}$. The distribution corresponding to ψ can be sampled via Algorithm 3.10, where ψ_2 corresponds to a $\Gamma(V_0, 1)$ and $\psi_3^{V_2}$ to a $\Gamma((1-\vartheta_0)V_2/(\vartheta_0\vartheta_1), 1)$ distribution. Similarly for the family combination (A,20), where $h = e - 1$, $\psi_2(t) = (1+t^{1/\vartheta_1})^{-V_0}$, and $\psi_3(t) = (1+t)^{-((1-\vartheta_0)/\vartheta_0)^{\vartheta_1}}$. Note that $V_2 \sim \mathcal{LS}^{-1}[\psi_2]$ can be sampled via $V_2 = SV^{\vartheta_1}$, with $S \sim \mathrm{S}(1/\vartheta_1, 1, \cos^{\vartheta_1}(\pi/(2\vartheta_1)), 0; 1)$ and $V \sim \Gamma(V_0, 1)$, and ψ_3 corresponds to a $\Gamma(((1-\vartheta_0)/\vartheta_0)^{\vartheta_1}, 1)$ distribution.

(3) For the family combination (C,12), F_0 is a $\Gamma(1/\vartheta_0, 1)$ distribution. Further, $V_{0,1} \sim F_{0,1}$ can be sampled as $V_{0,1} = S\tilde{S}^{\vartheta_1}$, where $S \sim \mathrm{S}(1/\vartheta_1, 1, \cos^{\vartheta_1}(\pi/(2\vartheta_1)), 0; 1)$ and $\tilde{S} \sim \tilde{\mathrm{S}}(\vartheta_0, 1, (\cos(\pi\vartheta_0/2)V_0)^{1/\vartheta_0}, 0, 1; 1)$, which can be sampled via the fast rejection algorithm. For the family combination (C,14) use the procedure for (C,12) where for $V_{0,1}$, ϑ_0 is replaced by $\vartheta_0\vartheta_1$.

(4) For the family combination (C,19), F_0 is a $\Gamma(1/\vartheta_0, 1)$ distribution. Further, $V_{0,1} \sim F_{0,1}$ can be sampled via the fast rejection algorithm, since $\psi_{0,1}(t)$ is of the form $\psi(t+h)/\psi(h)$, where $h = \exp(\vartheta_1) - 1$, $\psi(t) = \psi_2(-\log \psi_3(t))$, $\psi_2(t) = \exp(-V_0(t/\vartheta_1)^{\vartheta_0})$, and $\psi_3(t) = (1+t)^{-1}$. The distribution corresponding to ψ can be sampled via Algorithm 3.10, where ψ_2 corresponds to a $\mathrm{S}(\vartheta_0, 1, (\cos(\pi\vartheta_0/2)V_0)^{1/\vartheta_0}/\vartheta_1, 0; 1)$ and $\psi_3^{V_2}$ to a $\Gamma(V_2, 1)$ distribution. For the family combination (C,20) use the procedure for (C,19) with ϑ_0 replaced by ϑ_0/ϑ_1 and ϑ_1 replaced by 1.

Proof

For the statements about F_0, see Joe (1997, p. 375) and Hofert (2008). For Part (1), note that $\psi_{0,1}(t; V_0) = \psi((1+t)^{1/\vartheta_1} - 1)$, where $\psi(t) = (1 + (1-\vartheta_0)t)^{-V_0}$ corresponds to a $\Gamma(V_0, 1/(1-\vartheta_0))$ distribution. Hence, an application of Theorem 3.4 leads to the result as stated. For Part (2) and family combination (A,19), $\psi_{0,1}(t; V_0) = (((1-\vartheta_0)/\vartheta_1)\log(\exp(\vartheta_1) + t) + \vartheta_0)^{-V_0}$, which is of the form as stated. The result therefore follows from the fast rejection algorithm and Algorithm 3.10. Similarly, for the family combination (A,20), $\psi_{0,1}(t) = (\log^{1/\vartheta_1}((e+t)^{(1-\vartheta_0)^{\vartheta_1}}) + \vartheta_0)^{-V_0}$, which is of the form as stated. The result again follows from the fast rejection algorithm and Algorithm 3.10. The statement about V_2 follows from Theorem 3.4. For Part (3) and family combination (C,12), $\psi_{0,1}(t; V_0) = \exp(-V_0((1+t^{1/\vartheta_1})^{\vartheta_0} - 1))$, which is of type $\tilde{\psi}(t^{1/\vartheta_1})$ based on $\tilde{\psi}(t) = \exp(-V_0((1+t)^{\vartheta_0} - 1))$, i.e. an outer power family. The statement therefore follows from Theorem 3.4. The statement for the family combination (C,14) is clear. For Part (4) and family combination (C,19), $\psi_{0,1}(t; V_0) = \exp(-V_0(\vartheta_1^{-\vartheta_0}\log^{\vartheta_0}(\exp(\vartheta_1)+t) - 1))$, which is of the form as stated. The result therefore follows from the fast rejection algorithm and Algorithm 3.10. The statement for the family combination (C,20) is clear. □

4 Examples

In this section we consider several examples. First, we start with investigating two different stategies for finding quantiles of $F = \mathcal{LS}^{-1}[\psi]$ for sampling the bivariate Archimedean families of Frank and Joe. We also present precision and run-time results for the fully-nested Archimedean copulas of type (3) based on these two families, see

Theorem 3.2. Second, we compare the algorithms of Rosiński (2007), Ridout (2008), the rejection algorithm proposed by McNeil (2008), and the fast rejection algorithm for generating vectors of random variates from three-dimensional fully-nested Clayton copulas with different parameters. Third, we present precision and run-time results for various fully-nested Archimedean copulas. The three-dimensional case is again chosen exemplarily. To demonstrate that all algorithms apply to more than two hierarchies and may involve different Archimedean families, we finally sample a four-dimensional fully-nested Archimedean copula with three levels of nesting involving three different Archimedean families.

As an indicator of precision, we use the bivariate measure of dependence Kendall's tau, see Nelsen (2007, p. 158). Its population and sample versions are denoted by $\tau_{i,j}$ and $\hat{\tau}_{i,j}$, respectively, where the indices correspond to the components i and j under consideration. Further, κ is used to denote run time measured in seconds.

4.1 A word concerning the implementation

All algorithms are implemented in C/C++ and compiled using the GCC 3.3.3 (SuSE Linux) with option `-O2` for code optimization. The algorithms are run on a node containing two AMD Opteron 252 processors with 2.6 GHz and 8 GB RAM as part of a Linux cluster. The command `gettimeofday` is used to measure run time as wall-clock time. For generating uniform random variates an implementation of the Mersenne Twister by Wagner (2003) is used.

4.2 The families of Frank and Joe

We first investigate two different approaches for finding quantiles for the discrete distributions involved in sampling two-dimensional Frank and Joe copulas with the algorithm of Marshall and Olkin (1988). The direct approach uses a bisection procedure, the second approach first sorts the uniform random variates and then searches monotonically through the positive integers to locate all quantiles, see Ridout (2008). At a first glance, each approach can be advantageous towards the other depending on the number n of random variates to be drawn from F and the number of precomputed values for the quantiles. Table 3 therefore contains the number of precomputed values, with 500 000 as upper limit, and different choices of n. For all investigated dependencies we choose the copula parameters such that Kendall's tau ranges from 0.1 to 0.6, which we feel is adequate for most applications. Note that for all investigated sample sizes n there is no improvement in speed by using sorted uniform random variates.

Table 4 contains pairwise sample versions of Kendall's tau and run times for generating 100 000 vectors of random variates from fully-nested three-dimensional Frank and Joe copulas with parameters chosen such that pairwise Kendall's taus listed in the first two columns result; $\tau_{1,2;3}$ stands for $\tau_{1,2} = \tau_{1,3}$. For Joe's family, the number of summands to be sampled according to Theorem 3.2 becomes large as the dependence increases, therefore run time increases accordingly. However, for a wide range of dependencies, both algorithms are fast.

			$n = 1\,000$		$n = 10\,000$		$n = 100\,000$	
Family	τ	# prec. val.	direct	sort	direct	sort	direct	sort
F	0.1	30	0.0004	0.0005	0.0044	0.0051	0.0436	0.0531
	0.2	89	0.0004	0.0005	0.0044	0.0051	0.0441	0.0527
	0.3	263	0.0005	0.0005	0.0045	0.0051	0.0453	0.0529
	0.4	908	0.0005	0.0005	0.0046	0.0052	0.0461	0.0530
	0.5	4323	0.0007	0.0007	0.0049	0.0054	0.0470	0.0535
	0.6	37 969	0.0024	0.0024	0.0067	0.0070	0.0496	0.0546
J	0.1	500 000	0.1214	0.1219	0.1288	0.1307	0.1956	0.2015
	0.2	500 000	0.1215	0.1214	0.1278	0.1308	0.1956	0.2053
	0.3	500 000	0.1219	0.1245	0.1278	0.1304	0.1991	0.2044
	0.4	500 000	0.1224	0.1240	0.1284	0.1310	0.1961	0.2038
	0.5	500 000	0.1215	0.1244	0.1284	0.1309	0.1969	0.2039
	0.6	500 000	0.1213	0.1254	0.1280	0.1309	0.1976	0.2036

Table 3 Comparison of run times, measured in seconds, for sampling two-dimensional Frank and Joe copulas with two different algorithms for finding quantiles.

		Frank				Joe			
$\tau_{1,2;3}$	$\tau_{2,3}$	$\hat{\tau}_{1,2}$	$\hat{\tau}_{1,3}$	$\hat{\tau}_{2,3}$	κ	$\hat{\tau}_{1,2}$	$\hat{\tau}_{1,3}$	$\hat{\tau}_{2,3}$	κ
0.1	0.2	0.1011	0.0960	0.1994	0.1808	0.0971	0.0968	0.1993	0.4906
0.1	0.3	0.1013	0.0966	0.2992	0.1820	0.0967	0.0966	0.2991	0.4907
0.1	0.4	0.1013	0.0971	0.3993	0.1835	0.0968	0.0966	0.3990	0.4965
0.1	0.5	0.1012	0.0975	0.4997	0.1858	0.0966	0.0967	0.4987	0.4976
0.1	0.6	0.1011	0.0980	0.6002	0.1954	0.0967	0.0968	0.5978	0.5046
0.2	0.3	0.1998	0.1991	0.3001	0.1976	0.2001	0.2039	0.2997	0.8158
0.2	0.4	0.2001	0.1993	0.4001	0.1997	0.2007	0.2039	0.4002	0.8374
0.2	0.5	0.2003	0.1994	0.4999	0.2029	0.2011	0.2037	0.5004	0.8651
0.2	0.6	0.2003	0.1994	0.5997	0.2127	0.2013	0.2034	0.5994	0.9097
0.3	0.4	0.2979	0.2974	0.4003	0.2206	0.2950	0.2955	0.3959	2.5268
0.3	0.5	0.2975	0.2973	0.4998	0.2235	0.2948	0.2952	0.4961	2.6435
0.3	0.6	0.2970	0.2969	0.5995	0.2388	0.2945	0.2949	0.5961	2.8593
0.4	0.5	0.4018	0.4010	0.5015	0.2719	0.3978	0.3986	0.4971	9.7040
0.4	0.6	0.4011	0.4003	0.6009	0.2924	0.3984	0.3988	0.5972	10.4476
0.5	0.6	0.5002	0.5016	0.6005	0.4940	0.4998	0.5003	0.5988	34.9849

Table 4 Comparison of precision and run times, measured in seconds, for fully-nested three-dimensional Frank and Joe copulas based on 100 000 vectors of random variates.

4.3 Comparison of algorithms for nested Clayton copulas

Sampling exponentially-tilted Stable distributions, involved e.g. in a nested Clayton copula, is not an easy task. In this section we investigate the performance of the algorithms of Rosiński (2007), Ridout (2008), the standard rejection algorithm, and our proposed fast rejection algorithm for sampling three-dimensional nested Clayton copulas. This sampling problem is especially complicated, since we do not have to sample random variates from the same distribution with fixed parameters, but almost surely changing parameters for every random variate to be generated, due to the dependence of $F_{0,1}$ on $V_0 \sim \Gamma(1/\vartheta_0, 1)$. For this reason, an algorithm which works fast for sampling a large amount of exponentially-tilted Stable distributed random variates but which requires complicated setup steps depending on the parameters of the distribution may not be adequate for sampling a nested Clayton copula; this is essentially what we see in Table 5 for 100 000 generated vectors of random variates.

First consider the procedure of Rosiński (2007). Since V_0 almost surely changes for every random variate $V_{0,1}$ to be generated, Algorithm 3.9 can not be efficiently used. Instead, we have to use representation (6) for sampling $V_{0,1}$, which requires generating $3J$ random variates for each random variate $V_{0,1}$ to be drawn. For the computations in Table 5 we choose $J = 10\,000$. Although this choice is known to be rather fast than unbiased, see Ridout (2008), the algorithm of Rosiński (2007) turns out to be slow. Note that this can be different if a large amount of exponentially-tilted Stable distributed random variates with fixed parameters is required.

Similarly, the algorithm of Ridout (2008) requires some setup steps, which change with any V_0 from Step (1) of Algorithm 2.3. This, together with the fact that finding the quantile of a uniform random variate with a modified Newton-Raphson method can be time-consuming, causes the algorithm to be slow, too. Further, the numbers in parentheses in Table 5 show the number of warnings we obtained with the default parameter choices of this algorithm. These warnings indicate that the default accuracy for solving $F_{0,1}(x) = u$ with respect to x for a uniform random variate u could not be achieved after the default number of iterations of the modified Newton-Raphson method; after 1 000 iterations $|F_{0,1}(x) - u|$ is still larger than 10^{-7}. Further, for larger dependencies, measured with Kendall's tau, it was not possible to locate a quantile for at least one of the random variates, see the error message "`Cannot locate upper quantile`" of the implementation of the algorithm of Ridout (2008) for details. These cases are indicated by a dash.

The standard rejection algorithm proposed by McNeil (2008) is an exact algorithm. It neither requires approximations nor numerically complicated steps. Only the fact that speed becomes an issue if there is little dependence is a severe problem, due to the exponential complexity of this algorithm in V_0. For some parameter setups it was not possible to generate 100 000 vectors of random variates in less than 555 hours, indicated by a plus sign.

For the case where $\tau_{1,2;3} = 0.025$ and $\tau_{2,3} = 0.6$, a star indicates a problem we found for the standard and also the fast rejection algorithm. Due to the resulting small choice $\alpha = \vartheta_0/\vartheta_1 = 2/117$, the implementation of the function `gsl_ran_levy_skew` of the GSL

library for generating Stable distributed random variates did not always return proper positive real numbers (`nan` was returned in some cases) and the run time for the standard rejection algorithm in this case is not reliable. For the fast rejection algorithm, this only occurred for three generated vectors of random variates. Note that this is only a technical problem, however, we feel obliged to address such problems as both researchers and practitioners often do not seem to be aware of such.

The principle underlying the fast rejection algorithm to reduce the complexity from exponential to linear in V_0 significantly decreases run time. As Theorem 3.7 implies, for every random variate $V_0 > 1$, the algorithm is uniformly faster than the standard rejection algorithm over all dependencies. Further, the linear complexity in V_0 causes this algorithm to be efficient over the whole range of investigated parameters and complicated setup steps are also not required. Moreover, this algorithm is exact. Therefore, the fast rejection algorithm is highly recommendable, e.g. for sampling nested Clayton copulas.

4.4 Overall precision and run-time comparison

Table 6 contains pairwise sample versions of Kendall's tau and run times for different three-dimensional fully-nested Archimedean copulas based on the families of Ali-Mikhail-Haq, Clayton, Frank, Gumbel, and Joe, as well as an outer power Clayton copula (opC), see Hofert (2008), and the family combinations addressed in Theorem 3.11. As benchmark, we include a Gauss copula (Ga) and a t-copula with four degrees of freedom (t_4), see McNeil et al. (2005). For all examples, reasonable parameters are chosen such that pairwise Kendall's taus as given in columns four and five of Table 6 result. The last four columns indicate that all presented algorithms are accurate and fast enough to be applied in large-scale simulation studies.

4.5 A mixed fully-nested Archimedean copula with three hierarchies

The presented algorithms are also applicable to more than two hierarchies. We exemplarily sample a four-dimensional fully-nested Archimedean copula based on the families of Ali-Mikhail-Haq, Clayton, and the family numbered 20 in Nelsen (2007, pp. 118) on the first, second, and third level, respectively. To fulfill the parameter restrictions listed in Table 2, the corresponding parameters are chosen to match pairwise Kendall's tau of 0.2, 0.4, and 0.8. The plot on the left side of Figure 1 shows a scatter plot matrix of 1 000 generated vectors of random variates from this copula. Note that there is a singular component which should not be present. This is again due to a technical problem we found. For sampling this copula with the algorithm of McNeil (2008), the random variables $V_0 \sim \mathrm{Geo}(1-\vartheta_0)$, $V_{0,1} = \tilde{S}V^{\vartheta_1}$ with $V \sim \Gamma(V_0, 1/(1-\vartheta_0))$ and $\tilde{S} \sim \tilde{\mathrm{S}}(1/\vartheta_1, 1, \cos^{\vartheta_1}(\pi/(2\vartheta_1)), 0, V^{\vartheta_1}; 1)$, and $V_{1,2}$ with corresponding Laplace-Stieltjes transform $\psi_{1,2}(t; V_{0,1}) := \psi(t+h)/\psi(h)$ with $h = e-1$, $\psi(t) = \psi_3(-\log\psi_4(t))$, $\psi_3(t) = \exp(-V_{0,1}t^\alpha)$, $\alpha = \vartheta_1/\vartheta_2$, and $\psi_4(t) = (1+t)^{-1}$ are involved. Due to ψ_4, this requires sampling a $\Gamma(V_3, 1)$ distribution, where V_3 is a random variate following the distribution corresponding to ψ_3, a $\mathrm{S}(\alpha, 1, (\cos(\pi\alpha/2)V_{0,1})^{1/\alpha}, 0; 1)$ distribution, see Algorithm 3.10. For this task, we used the function `gsl_ran_gamma` of the GSL library. As the random

$\tau_{1,2;3}$	$\tau_{2,3}$	Rosiński	Ridout	Standard rejection	Fast rejection
0.025	0.05	657.45	25.95	+	6.03
0.025	0.1	655.95	21.09	+	5.76
0.025	0.2	656.01	15.75 (34)	+	5.70
0.025	0.3	633.28	15.72 (1)	+	5.69
0.025	0.4	635.48	24.73 (38)	+	5.61
0.025	0.5	637.52	58.91 (291)	+	5.81
0.025	0.6	633.87	57.46 (146)	112716.80*	5.63*
0.05	0.1	660.30	21.56	659649.15	3.17
0.05	0.2	659.39	16.50 (13)	611434.81	3.01
0.05	0.3	659.29	16.45 (4)	690703.44	3.19
0.05	0.4	637.69	23.05 (36)	697543.63	2.95
0.05	0.5	634.46	46.42 (163)	727002.83	2.89
0.05	0.6	633.52	59.99 (229)	650025.72	2.91
0.1	0.2	660.02	18.24 (1)	1181.88	1.66
0.1	0.3	655.29	16.82 (15)	1047.89	1.56
0.1	0.4	660.19	20.29 (10)	1286.70	1.58
0.1	0.5	660.05	34.64 (93)	1717.43	1.58
0.1	0.6	633.00	55.03 (194)	1719.67	1.57
0.2	0.3	658.12	21.11 (3)	15.47	0.90
0.2	0.4	654.78	19.52 (4)	11.28	0.87
0.2	0.5	656.11	24.02 (18)	14.66	0.85
0.2	0.6	659.33	38.76 (102)	23.49	0.86
0.3	0.4	662.11	25.25 (1)	2.42	0.65
0.3	0.5	658.06	23.51 (6)	2.59	0.64
0.3	0.6	657.26	–	2.38	0.63
0.4	0.5	661.35	–	2.27	0.55
0.4	0.6	657.59	–	1.72	0.54
0.5	0.6	659.01	–	1.28	0.50

Table 5 Comparison of run times, measured in seconds, for generating 100 000 vectors of random variates from fully-nested three-dimensional Clayton copulas with different parameters.

Copula	ϑ_0	ϑ_1	$\tau_{1,2;3}$	$\tau_{2,3}$	$\hat{\tau}_{1,2}$	$\hat{\tau}_{1,3}$	$\hat{\tau}_{2,3}$	κ
A	0.4015	0.9430	0.1	0.3	0.0956	0.0989	0.2990	0.4113
C	0.5	2	0.2	0.5	0.2008	0.1974	0.4987	1.1998
F	1.8609	5.7363	0.2	0.5	0.2000	0.1977	0.5003	0.5033
G	1.25	2	0.2	0.5	0.2017	0.2012	0.4981	0.7300
J	1.4438	2.8562	0.2	0.5	0.2006	0.2010	0.4992	1.0500
opC	1.1364	1.8182	0.2	0.5	0.1988	0.2026	0.5011	0.8502
(A,C)	0.7135	2	0.2	0.5	0.2030	0.2022	0.4992	0.7957
(A,19)	0.7135	0.4462	0.2	0.5	0.1972	0.1985	0.5005	0.6251
(A,20)	0.7135	1.0243	0.2	0.61	0.2012	0.2005	0.6115	1.2237
(C,12)	0.5	1.3333	0.2	0.5	0.2006	0.2004	0.4994	1.2123
(C,14)	0.5	1.5	0.2	0.5	0.2012	0.2007	0.5012	1.4839
(C,19)	0.5	0.4462	0.2	0.5	0.2011	0.2029	0.5009	1.1116
(C,20)	0.5	0.7250	0.2	0.5	0.2016	0.1980	0.4987	1.4005
Ga	0.3090	0.7071	0.2	0.5	0.1963	0.1981	0.4992	0.3822
t_4	0.3090	0.7071	0.2	0.5	0.1997	0.2038	0.4974	0.6174

Table 6 Precision and run time results, measured in seconds, for different three-dimensional copulas based on 100 000 generated vectors of random variates.

variates V_3 tend to be small for the chosen parameters, the Gamma random variate generator returned 0 instead of some almost surely positive number. The Gamma random variate generator `rgamma` used in the statistical software R also shows this problem. The implementation `g05ffc` of the NAG library does not show this problem, see the plot on the right side of Figure 1, however, this routine increases the run time for generating 100 000 vectors of random v ariates from 1.22s for the GSL implementation to 16.42s for the one of NAG.

5 Conclusion

We presented sampling strategies applicable to all commonly used exchangeable and nested Archimedean copulas. Sampling nested Archimedean copulas with the elegant algorithm of McNeil (2008) is particularly challenging, as the distribution $F_{0,1}$ related to the nodes of the copula depends on the parameter $V_0 \sim F_0$, a random variate obtained from an earlier step of the algorithm. Therefore, it almost surely changes for every vector of random variates to be generated if F_0 is absolutely continuous, or at least possibly quite often if F_0 is discrete. The main idea behind our efficient algorithms is to reduce the influence of V_0 on $F_{0,1}$. For the presented algorithms for sampling nested Frank and Joe copulas this means precomputing a distribution not depending on V_0 and sampling this distribution V_0-times to obtain a random variate $V_{0,1}$. This procedure only requires finding quantiles from a fixed precomputed distribution instead of a distribution which changes for different random variates V_0. For sampling nested Ali-Mikhail-Haq copulas,

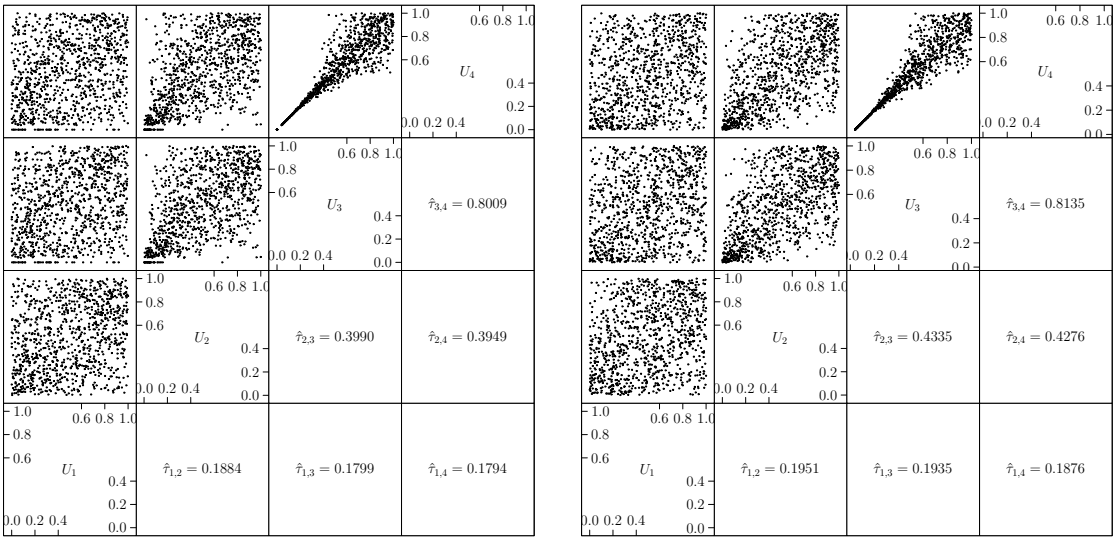

Figure 1 1 000 vectors of random variates from a fully-nested Archimedean copula with three levels; the Gamma distribution involved is sampled via GSL (left) and NAG (right).

the distribution not depending on V_0 is simply a Geometric distribution and therefore does not even have to be precalculated.

Theorem 3.4 addressed a general result how to build a nested Archimedean copula from a given Archimedean generator. For sampling the exponentially-tilted Stable distribution involved, we examined the algorithms of Rosiński (2007), Ridout (2008), and the standard rejection algorithm. As the first two methods are inaccurate and rather slow, and the standard rejection algorithm often turns out to be too slow, we proposed a new algorithm designed for sampling tilted Archimedean generators in general. Our fast rejection algorithm is exact and only of logarithmic complexity in comparison to the standard rejection algorithm. For sampling nested Clayton copulas for example, the complexity of the fast rejection algorithm is only linear in V_0, which again significantly reduced the influence of V_0 on the run time of the algorithm, see Table 5.

By using the fast rejection algorithm in conjunction with a result on randomized generator parameters we were able to find explicit algorithms for sampling all seven nested Archimedean copulas based on generators belonging to different Archimedean families as presented in Hofert (2008).

Our algorithms are promising and the underlying ideas may also serve as strategies for sampling Archimedean copulas not addressed in this paper or even other distributions given by their Laplace-Stieltjes transforms in a more general context. Further, our findings encourage the use of the flexible class of nested Archimedean copulas in large-scale multidimensional simulation studies as an alternative to standard elliptical distributions, which are often appreciated for their simple sampling algorithms, yet are restricted to radial symmetry.

References

Barndorff-Nielson, O. E. and Shephard, N. (2001), "Normal modified Stable processes", http://www.economics.ox.ac.uk/research/WP/PDF/paper072.pdf (2009-08-16).

Bondesson, L. (1982), "On simulation from infinitely divisible distributions", Advances in Applied Probability, 14, 855–869.

Brix, A. (1999), "Generalized gamma measures and shot-noise Cox processes", Advances in Applied Probability, 31, 929–953.

Damien, P., Laud, P. W., and Smith, A. F. M. (1995), "Approximate Random Variate Generation from Infinitely Divisible Distributions with Applications to Bayesian Inference", Journal of the Royal Statistical Society: Series B (Statistical Methodology), 57, 3, 547–563.

Devroye, L. (1986), "Non-Uniform Random Variate Generation", Springer.

Embrechts, P., Lindskog, F., and McNeil, A. J. (2001), "Modelling Dependence with Copulas and Applications to Risk Management", http://www.risklab.ch/ftp/papers/DependenceWithCopulas.pdf (2009-08-16).

Feller, W. (1971), "An Introduction to Probability Theory and Its Applications", 2, Wiley.

GSL, http://www.gnu.org/software/gsl/ (2009-08-16).

Hering, C. and Hofert, M. (2009), "Goodness-of-fit tests for Archimedean copulas in large dimensions", submitted, http://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.zawa/forschung/hering_hofert_2009.pdf (2009-08-16).

Hofert, M. (2008), "Sampling Archimedean copulas", Computational Statistics & Data Analysis, 52, 12, 5163–5174.

Hofert, M. and Scherer, M. (2008), "CDO pricing with nested Archimedean copulas", submitted, http://www.mathematik.uni-ulm.de/numerik/preprints/2008/CDOpricingAC.pdf (2009-08-16).

Hougaard, P. (1986), "Survival models for heterogeneous populations derived from stable distributions", Biometrika, 73, 2, 387–396.

Joe, H. (1997), "Multivariate Models and Dependence Concepts", Chapman & Hall/CRC.

Joe, H. and Hu, T. (1996), "Multivariate Distributions from Mixtures of Max-Infinitely Divisible Distributions", Journal of Multivariate Analysis, 57, 240–265.

Marshall, A. W. and Olkin, I. (1988), "Families of Multivariate Distributions", Journal of the American Statistical Association, 403, 83, 834–841.

McCulloch, J. H. (2003), "The Risk-Neutral Measure and Option Pricing under Log-Stable Uncertainty", http://economics.sbs.ohio-state.edu/pdf/mcculloch/wp03-07.pdf (2009-08-16).

McCulloch, J. H. and Lee, S. H. (2007), "Estimation of the Risk Neutral Measure with the Stable Option Pricing Model", https://editorialexpress.com/cgi-bin/conference/download.cgi?db_name=sce2007&paper_id=305 (2008-11-01).

McNeil, A. J. (2008), "Sampling nested Archimedean copulas", Journal of Statistical Computation and Simulation, 6, 78, 567–581.

McNeil, A. J. and Nešlehová, J. (2009), "Multivariate Archimedean copulas, d-monotone functions and l_1-norm symmetric distributions", The Annals of Statistics, in press.

McNeil, A. J., Frey, R., and Embrechts, P. (2005), "Quantitative Risk Management: Concepts, Techniques, and Tools", Princeton University Press.

NAG, `http://www.nag.co.uk/` (2009-08-16).

Nelsen, R. B. (2007), "An Introduction to Copulas", Springer.

Nolan, J. P. (2009), "Stable Distributions - Models for Heavy Tailed Data", Birkhäuser, `http://academic2.american.edu/~jpnolan/stable/chap1.pdf` (2009-08-16).

R, `http://www.r-project.org/` (2009-08-16).

Ridout, M. (2008), "Generating random numbers from a distribution specified by its Laplace transform", `http://www.kent.ac.uk/IMS/personal/msr/webfiles/rlaptrans/SimRandom3.pdf` (2009-08-16).

Rosiński, J. (2001), "Series representations of Lévy processes from the perspective of point processes", Lévy Processes - Theory and Applications, Barndorff-Nielsen, O. E., Mikosch, T., and Resnick, S. I., Birkhäuser, `http://www.math.utk.edu/~rosinski/Manuscripts/seriesppF.pdf` (2009-08-16).

Rosiński, J. (2007), "Tempering Stable processes", Stochastic Processes and their Applications, 117, 677–707.

Schoutens, W. (2003), "Lévy Processes in Finance: Pricing Financial Derivatives", Wiley.

Tweedie, M. C. K. (1984), "An index which distinguishes between some important exponential families", Statistics: Applications and New Directions: Proceedings Indian Statistical Institute Golden Jubilee International Conference, 579–604.

Wagner, R. (2003), "Mersenne Twister Random number Generator", `http://www-personal.umich.edu/~wagnerr/MersenneTwister.html` (2009-08-16).