Noname manuscript No.
(will be inserted by the editor)

Generating random numbers from a distribution specified by its Laplace transform

M. S. Ridout

Received: date / Accepted: date

Abstract This paper discusses simulation from an absolutely continuous distribution on the positive real line when the Laplace transform of the distribution is known but its density and distribution functions may not be available. We advocate simulation by the inversion method using a modified Newton-Raphson method, with values of the distribution and density functions obtained by numerical transform inversion. We show that this algorithm performs well in a series of increasingly complex examples. Caution is needed in some situations when the numerical Laplace transform inversion becomes unreliable. In particular the algorithm should not be used for distributions with finite range. But otherwise, except for rather pathological distributions, the approach offers a rapid way of generating random samples with minimal user effort. We contrast our approach with an alternative algorithm due to Devroye (1981).

Keywords Archimedean copula · Laplace transform inversion · positive stable distribution · tempered stable distribution

1 Introduction

There are many situations in which a transform of a probability distribution is available, but the cumulative distribution function is not known in a simple closed form and this raises the question of how one might simulate from such a distribution. In this article, we focus on absolutely continuous random variables on the positive real line and assume that the Laplace transform of the distribution is known. We investigate the use of the inversion method to generate random variables from the distribution, using a modified Newton-Raphson algorithm, with values of the distribution and density functions obtained by numerical transform inversion. Details are given in Section 2. Note that we shall refer to the inversion both of the cumulative distribution function (inversion method) and of the Laplace-Stieltjes transform of this function (transform inversion); it should be clear from the context which usage is intended.

In Section 3, we outline an alternative general approach, due to Devroye (1981). Section 4 compares R implementations of these and other methods in terms of speed and accuracy for a series of examples. Whilst the proposed method is not competitive with built-in generators or special purpose algorithms developed for particular distributions, we show that it is fast enough and accurate enough to be practical for many applications.

R functions implementing the main algorithms discussed in the paper are available at
`http://www.kent.ac.uk/ims/personal/msr/rlaptrans.html`

2 Random number generation from the Laplace transform

Let X be an absolutely continuous random variable defined on $(0, \infty)$, with probability density function (p.d.f.) $f(x)$ and cumulative distribution function (c.d.f.) $F(x)$. The Laplace transform of $f(x)$ is defined as

$$f^*(s) = \int_0^\infty e^{-sx} f(x) \mathrm{d}x,$$

provided that the integral converges. The argument s may be complex and the integral converges in the right

M. S. Ridout
Institute of Mathematics, Statistics and Actuarial Science, University of Kent, Canterbury, Kent CT2 7NF, U.K.
E-mail: m.s.ridout@kent.ac.uk

half-plane $\mathrm{Re}(s) > a$ for some $a \leq 0$. The Laplace transform of $F(x)$, denoted by $F^*(s)$, satisfies

$$F^*(s) = \frac{f^*(s)}{s}.$$

The function $f^*(s)$ is equivalently the Laplace-Stieltjes transform of $F(x)$, which exists for $\mathrm{Re}(s) \geq 0$ for all distributions on the positive real line, whether or not they are absolutely continuous.

For future reference, we note that for real s the characteristic function (Fourier-Stieltjes transform) of X is given by

$$\phi(s) = f^*(-is),$$

where $i^2 = -1$.

2.1 The inversion method

Suppose that we wish to generate a random value x from the distribution of X. The inversion method achieves this by generating a random variable u from the uniform distribution $U(0,1)$ and obtaining x as the solution to the equation

$$F(x) = u. \tag{1}$$

For a detailed discussion of the inversion method, see, for example, Chapter 2 of Devroye (1986b).

Generally an iterative approach is required to solve equation (1), even if $F(x)$ is known explicitly. The Newton-Raphson method, which involves the iterative scheme

$$x^{(k+1)} = x^{(k)} - \frac{F\left(x^{(k)}\right) - u}{f\left(x^{(k)}\right)}, \tag{2}$$

provides one possible approach. However, the Newton-Raphson method often fails to converge for this problem because in the upper tail of the distribution the c.d.f. is flat, the p.d.f. is close to zero and the Newton-Raphson updates may be very large and inaccurate. Similar problems are likely to arise in any region in which the c.d.f. is flat, as may occur in the lower tail of the distribution or between modes of a multimodal distribution, for example.

We have therefore used a standard modification of the Newton-Raphson method (Press et al., 1992, Section 9.4), which switches to the bisection method where necessary. This involves keeping track of lower and upper bounds within which the solution to equation (1) is known to lie. These are updated at each iteration and if the proposed Newton-Raphson update lies outside these bounds the updated value is instead set to the mean of the upper and lower bounds.

The following algorithm exploits an additional idea that usually improves efficiency when generating n values from a distribution, with $n > 1$, namely to generate an ordered random sample $x_{(1)}, \ldots, x_{(n)}$ by sorting the initial uniform random numbers. This becomes efficient as n increases because $x_{(i-1)}$ can serve as a starting value for the iterative determination of $x_{(i)}$. Once the ordered sample has been generated, it is permuted randomly to obtain an unordered random sample. For small values of n, the improved efficiency of root finding may be insufficient to offset the overheads of sorting; this is investigated further in Section 4.

Algorithm RLAPTRANS

1. Generate n independent $U(0,1)$ random variables and sort these to give the ordered sample $u_{(1)} < \ldots < u_{(n)}$.
2. Find a value x_{max} such that $F(x_{max}) \geq u_{(n)}$. Set the constants x_{start}, b and j_{max}. Set $x_{max} = x_{start}$ and $j = 0$.
   WHILE ($F(x_{max}) < u_{(n)}$ AND $j < j_{max}$)
   $x_{max} = b x_{max}$
   $j = j + 1$
   END WHILE
   If $j = j_{max}$ the algorithm terminates having failed to find a suitable value of $x_{max} < x_{start} b^{j_{max}}$. Otherwise, set $x_{(0)} = 0$ and proceed to step 3.
3. Set the constant k_{max} and repeat the following modified Newton-Raphson procedure for $i = 1, \ldots, n$.
   Set $x_L = x_{(i-1)}$, $x_U = x_{max}$, $k = 0$ and $t = x_L$.
   WHILE($|F(t) - u_{(i)}| > \tau$ AND $k < k_{max}$)
   $k = k + 1$
   $t = t - (F(t) - u_{(i)})/f(t)$
   IF ($t \notin [x_L, x_U]$) THEN
   $t = (x_L + x_U)/2$
   END IF
   IF ($F(t) \leq u_{(i)}$) THEN
   $x_L = t$
   ELSE
   $x_U = t$
   END IF
   END WHILE
   If $k = k_{max}$ the procedure is terminated, having failed to achieve the desired tolerance. Otherwise set $x_{(i)} = t$.
4. Permute the ordered sample $x_{(1)}, \ldots, x_{(n)}$ randomly, to obtain the unordered sample $x_1, \ldots, x_n$.

Note that in principle it may be impossible to satisfy the convergence criterion

$$\left| F\left(x^{(k)}\right) - u \right| < \tau, \tag{3}$$

in step 3 of the algorithm if the smallest realisable increment in $x^{(k)}$ leads to a change in $F(x)$ that exceeds the tolerance parameter τ (Devroye, 1986b, p. 31). This difficulty is unlikely to arise in most practical applications, however.

Another theoretical difficulty that has been pointed out by a referee is that as n increases, the number of permutations of n will eventually exceed the number of states of any pseudo-random number generator, making it impossible to generate truly random permutations. However, this appears unlikely to lead to any important bias in practice. Random permutation could be avoided altogether by simply reordering the $x_{(i)}$ based on the original ordering of the uniform random numbers, but this required more computational time in our implementation.

The need to solve the equation $F(x) = u$ iteratively means that the algorithm is approximate, even if a perfect source of uniform random numbers is available and if computations with real numbers can be done without numerical error. Furthermore, the algorithm assumes that it is possible to calculate $F(x)$ and $f(x)$ for given x and we propose that these are instead approximated by numerical inversion of the Laplace transform, introducing further error. However, by choosing a suitable inversion algorithm, the numerical errors will generally be small.

2.2 Numerical inversion of Laplace transforms

There are many numerical methods for inverting Laplace transforms (Cohen, 2007). We have used the algorithm described by Abate et al. (2000) for numerical inversion of the Laplace transforms $f^*(s)$ and $F^*(s)$. This is based on approximating the Bromwich inversion integral for the inverse Laplace transform by the trapeziodal rule, using a step size chosen so that the successive terms alternate in sign eventually. The convergence acceleration technique known as Euler summation (Wimp, 1981) is then applied to this 'nearly alternating' sequence of terms.

The algorithm leads to three types of error - discretization error that results from using the trapezoidal rule to approximate the integral by an infinite sum, truncation error that results from replacing that infinite sum by a finite sum and then applying Euler summation and finally rounding error. The algorithm, which is summarised on p. 272 of Abate et al. (2000), has four controlling parameters, A, ℓ, m and n' (n' is simply n in Abate et al. (2000), but here we have used n to denote the sample size). Of these, A and ℓ jointly control the discretisation and rounding error (Abate et al., 2000), whereas ℓ, m and n' determine the truncation error. Detailed error analysis of Euler summation is provided by O'Cinneide (1997) and Sakurai (2004). Broadly speaking, the method becomes increasingly accurate with increasing smoothness of the function whose transform is being inverted. For well-behaved functions, the algorithm typically achieves a relative error of 10^{-7} or less.

Note that the accuracy of the random number generator is determined by the accuracy with which $F(x)$ is calculated, by virtue of equation (3). Inaccuracies in the calculation of $f(x)$ may slow down the rate of convergence of the Newton-Raphson algorithm but will not affect accuracy. The boundedness of the c.d.f. implies that the discretization error that arises from using the trapezoidal rule to approximate the Bromwich integral is bounded by $e^{-A}/(1 - e^{-A})$, which is approximately 10^{-8} for $A = 18.4$. If accuracy in the upper tail of the distribution is a concern it may be preferable to work with the survivor function $1-F(x)$ rather than $F(x)$ itself, since typically this reduces the discretization error further (Abate et al., 2000, Remark 8.2).

An alternative transform inversion algorithm, which also uses the trapezoidal rule but along a contour in the complex plane that is a deformation of the Bromwich contour, is due to Talbot (1979). Variants of this algorithm can offer similar accuracy to the method of Abate et al. (2000) for less computational effort (Trefethen et al., 2006). Such inversion methods might therefore be considered for specific examples. However, Talbot's method is unsuitable as a general algorithm because certain additional conditions on the transform must be met for the contour deformation to be valid and these conditions are not met for several of the examples in Section 4.

Both of these algorithms require the transform to be evaluated for complex arguments. Algorithms such as the Gaver-Stehfest algorithm, which require evaluation only for real arguments, may be very inaccurate unless implemented in a high precision computing environment (Abate and Valko, 2004). This makes them unsuitable for implementation with typical statistical software. The requirement for transform evaluation at complex arguments can lead to problems in packages such as R or Matlab if the transforms involve special functions, since these may not be implemented for complex arguments. This is one limitation of the method described here.

Whichever transform inversion algorithm is used, it may be modified in a straightforward way to allow simultaneous calculation of $F(x)$ and $f(x)$ from a single set of transform values. Thus the combined computa-

tional cost is only slightly greater than that of a single transform inversion.

2.3 Technical issues

The method described above is only appropriate for absolutely continuous distributions (i.e. those with a density). Often it may be clear from the context in which the transform was obtained that the distribution is absolutely continuous. However, more generally, we may wish to know whether a given function $g(s)$ is the Laplace transform of an absolutely continuous distribution on the positive real line.

This Section reviews some of the known results in this area. Since we are interested in automatic procedures for random number generation, we also discuss briefly the extent to which any of these mathematical results might lead to numerical procedures that could indicate automatically, and with high reliability, whether the function $g(s)$ is of the appropriate type.

The first result (Feller, 1971, p. 439), characterises functions that are the Laplace-Stieltjes transform of a probability distribution.

Theorem 1 *A function $g(s)$ defined on $[0, \infty)$ is the Laplace-Stieltjes transform of a probability distribution on $(0, \infty)$ if and only if $g(0) = 1$ and $g(\cdot)$ is completely monotone, that is it possesses derivatives $g^{(n)}(s)$ of all orders with*

$$(-1)^n g^{(n)}(s) \geq 0 \qquad (s > 0).$$

However, it is often difficult to establish mathematically whether a function is completely monotone.

If we are prepared to accept that $g(s)$ is the Laplace-Stieltjes transform of a c.d.f. $F(x)$, there is still the question of whether (a) $F(0) = 0$ and (b) the distribution is absolutely continuous. The first requirement can often be checked using the initial value theorem for Laplace transforms (e.g. Cohen 2007, p. 40), which gives $F(0) = \lim_{s\to\infty} g(s)$.

Conditions for absolute continuity are more readily expressed in terms of the characteristic function $\phi(s) = g(-is)$. We then have the following useful result (Feller, 1971, p. 509),

Theorem 2 *Let ϕ be the characteristic function of a probability distribution on $(0, \infty)$ and suppose that the integral*

$$\int_0^\infty |\phi(s)| \mathrm{d}s$$

is finite. Then the distribution is absolutely continuous, with a bounded continuous density.

A numerical integration routine, such as the R function `integrate`, can often indicate whether the integral is finite with reasonable reliability.

Unfortunately, Theorem 2 can only verify absolute continuity if the density is bounded and continuous, which excludes some common distributions, for example gamma distributions with shape parameter $\alpha \leq 1$, for which the integral in Theorem 2 does not converge. For such distributions, we have the following necessary condition, from Feller (1971, p. 514),

Theorem 3 *If $\phi(s)$ is the characteristic function of an absolutely continuous distribution on $(0, \infty)$, then $\lim_{s\to\infty} |\phi(s)| = 0$.*

Whilst the behaviour of $|\phi(s)|$ for large s can certainly be investigated numerically, it is known that the convergence to zero in Theorem 3 can occur arbitrarily slowly (Ushakov, 1999, pp. 261–262), so that no purely numerical test can be completely reliable. For a concrete example, consider the gamma distribution with shape parameter $\alpha < 1$ and scale parameter $\beta = 1$. For small ϵ , the smallest value of s such that $|\phi(s)| < \epsilon$ is approximately $\epsilon^{-1/\alpha}$. For example, with $\epsilon = 10^{-k}$ and $\alpha = 0.1$, s must exceed approximately 10^{10k} for the inequality to hold.

The converse of Theorem 3 is false; there exist continuous singular distributions for which $\phi(s) \to 0$ as $s \to \infty$ (Lukacs, 1970, p. 20). However, this is unlikely to be of much practical relevance.

3 An alternative algorithm

In a series of papers, Devroye (1981, 1984, 1986a, 1991) has developed some ingenious approaches to the problem of generating random variables from certain classes of characteristic function and from probability generating functions. In particular, Devroye (1981) gives an algorithm based on the characteristic function, $\phi(s)$, which assumes that ϕ is twice differentiable and that ϕ, ϕ' and ϕ'' are absolutely integrable and absolutely continuous. The algorithm may be simplified slightly when it is known that X is positive, leading to a two-fold reduction in the expected number of transform inversions required.

The simplified algorithm has the following form. A preliminary step involves evaluation of the integrals

$$c = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\phi(s)| \mathrm{d}s$$

and

$$k = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\phi''(s)| \mathrm{d}s.$$

Then the following algorithm is used to generate samples from the distribution:

Algorithm DEVROYE

1. Generate independent $U(0,1)$ random variables U, V_1 and V_2. Set $x = \sqrt{(k/c)}V_1/V_2$. Calculate $f(x)$ by transform inversion. If $V_1 \geq V_2$, go to step 3.
2. If $cU < f(x)$, exit with x. Otherwise go to step 1.
3. If $kU < x^2 f(x)$, exit with x. Otherwise go to step 1.

In this algorithm, the expected number of times that step 1 is executed, and hence the expected number of transform inversions required, is $2\sqrt{kc}$. Note that in contrast to algorithm RLAPTRANS, this algorithm is in principle exact, given a perfect source of uniform random numbers and the ability to store real numbers and invert the transform without error.

This algorithm can be automated, so that the user need simply provide the Laplace transform, by approximating ϕ'' numerically using the second-difference formula

$$\phi''(s) \approx \frac{f^*(-is[1+\delta]) + f^*(-is[1-\delta]) - 2f^*(-is)}{\delta^2 s^2} \quad (4)$$

and evaluating the constants c and k by numerical integration.

4 Examples of usage

In this Section we provide some simple examples of usage and discuss timings and limitations, based on R implementations of the algorithms RLAPTRANS and RDEVROYE. For RLAPTRANS, the controlling parameters are the parameters of the algorithm of Abate et al. (2000), which were set to $A = 19$, $\ell = 1$, $m = 11$ and $n' = 38$, which imply that each transform inversion requires the transform to be evaluated 50 times, and the tolerance parameter τ from equation (3), which was set to 10^{-7}. For step 2 of the algorithm, we set $x_{start} = 1$, $b = 2$, $j_{max} = 500$ and $k_{max} = 1000$. For RDEVROYE, the parameter δ in equation (4) was set to $\max(\varepsilon^{1/3}, 10^{-6})$, where ε is the machine precision (Monahan, 2001, p. 185) and the R function `integrate` was used, with default settings, to approximate the constants c and k.

Approximate timings were generated by using the `system.time` function in R and recording the 'elapsed time' output. All computations were done using version 2.7.0 of R on a 3.19 GHz PC with 1GB of RAM running Windows XP.

4.1 Gamma distribution

The Laplace transform of the p.d.f. of the gamma distribution with shape parameter α and scale parameter β is

$f^*(s) = (1 + \beta s)^{-\alpha}$.

Random samples of size n were generated from this distribution using the algorithms RLAPTRANS and, where possible, RDEVROYE. Table 1 shows elapsed times for the two methods, for various values of α and n, and the average number of transform inversions needed to generate each random value.

For $\alpha > 1$, use of RLAPTRANS is consistently faster. For Devroye's method the expected number of transform inversions is fixed whereas for RLAPTRANS the average decreases with the sample size due to the sorting procedure described in Section 2.1; note that even with $n = 10$, it is more efficient to use the sorting procedure than to generate values individually. For $n = 1$ and $n = 10$, Devroye's algorithm requires fewer transform inversions, for $n = 100$ the numbers of inversions required are similar and for $n = 1000$ RLAPTRANS requires fewer inversions. However, when n is small, the preliminary computation of the constants c and k in Devroye's algorithm adds a significant overhead, causing the algorithm to be slower overall. Moreover, the evaluation of these constants, which relies on an adaptive numerical integration method, becomes slower as $\alpha \to 1$ and this is largely responsible for the slower times recorded for this algorithm with $\alpha = 1.25$. In contrast, the CPU times for RLAPTRANS are approximately independent of α. Timings for both algorithms are approximately independent of the scale parameter β.

For $\alpha \leq 1$, the characteristic function ϕ is not absolutely integrable and therefore Devroye's method is not available. In principle, there is no difficulty in using RLAPTRANS for $\alpha \leq 1$ though the average number of transform inversions, and hence the overall computational time, increases sharply as α approaches zero. Table 1 gives results for $\alpha = 0.05$.

To assess the overall accuracy of RLAPTRANS, we modified the algorithm by replacing the uniform random numbers generated at the start of the algorithm with a set of fixed values comprising $0.0001, 0.001, 0.005, 0.01(0.01)0.99, 0.999, 0.9995, 0.9999$. Thus, the algorithm generates approximations to the quantiles of the gamma distribution corresponding to these values and errors in these approximations arise from the combination of inexact convergence of the modified Newton-Raphson method and inexact numerical transform inversion. Table 2 shows the maximum and median relative errors of the approximate quantiles generated in this way. For

$\alpha = 5$ and $\alpha = 2.5$, the maximum relative error occurs for the 0.9999 quantile, where there are problems due to the flatness of the c.d.f. In contrast, for $\alpha = 1.25$ and $\alpha = 0.05$, the maximum relative error occurs for the 0.0001 quantile, where the quantile itself is extremely small (5.84×10^{-81} for $\alpha = 0.05$), making it difficult to achieve a small *relative* error. In both cases, the performance is improved by reducing the tolerance parameter τ, to try to ensure more accurate convergence of the Newton-Raphson procedure. The scope for reducing error in this way is limited ultimately by the inaccuracy of the numerical transform inversion. The median relative errors are small even with the default setting of $\tau = 10^{-7}$, though again there are modest improvements as τ is reduced.

There are of course many efficient algorithms for generating gamma random variables. For example, the built-in R function `rgamma`, which uses the algorithm of Ahrens and Dieter (1982) for $\alpha \geq 1$ and that of Ahrens and Dieter (1974) for $0 < \alpha < 1$, is hundreds of times faster than RLAPTRANS.

4.2 Positive stable distribution

The positive stable distributions (Feller, 1971, pp. 448-449), the subclass of stable distributions that have support on the positive real axis, are heavy-tailed distributions with parameters $0 < \alpha < 1$ and $\gamma > 0$. The Laplace transform of the distribution is

$$f^*(s) = \exp\left(-\xi s^\alpha\right) \qquad (s \geq 0),$$

where $\xi = \gamma^\alpha / \cos(\pi\alpha/2)$, but simple closed forms for the p.d.f. or c.d.f. are not known, except in special cases.

The characteristic function is absolutely integrable, with $c = \Gamma(1 + 1/\alpha)/(\pi\gamma)$, but its second derivative is not absolutely integrable. Therefore Devroye's algorithm is not available.

Table 3 shows elapsed times for generating samples of size 100 and 1000 using RLAPTRANS for various values of α with $\gamma = 1$; elapsed times are approximately independent of γ. The times increase substantially as α becomes small. Even for $\alpha = 0.9$, elapsed times are greater than for the gamma distributions in Table 1. This is primarily because more computing time is needed to calculate the transform.

To assess the accuracy of RLAPTRANS for this example, we modified the algorithm as described in Section 4.1 to generate approximations to specified quantiles of the distribution. Here we let $\alpha = 0.5$, 0.7 or 0.9, with $\gamma = 1$ and generated approximate quantiles x_q for $q = 0.0001, 0.01, 0.5, 0.99, 0.9999$. We compared these with the accurate values calculated by McCulloch and Panton (1997) and available at
`http://www.econ.ohio-state.edu/jhm/fracden`.

The relative errors are shown in Table 4, for two choices of the tolerance parameter τ. These distributions have a very long right tail and the relative errors are largest at the 0.9999 quantile; the relative errors of other quantiles are reasonable. Reducing τ from 10^{-7} to 10^{-10} leads to only modest improvements and occasionally leads to small *increases* in the relative error. Changing the transform inversion parameters A and ℓ to $A = 24$, $\ell = 2$, which increases the computational effort, generally improves the accuracy in the upper tail and often elsewhere. However, there is sometimes a deterioration in accuracy, particularly for the 0.0001, 0.01 and 0.5 quantiles for $\alpha = 0.9$.

Again this example is intended to be illustrative. For practical generation of stable random variables one may use the fast algorithm of Chambers et al. (1976). In R, this is implemented as the function `rstable` in the package `fBasics`.

4.3 Tempered stable distribution

The operation of exponential tilting (Barndorff-Nielsen and Cox, 1989, p. 105) replaces an existing p.d.f. $f(x)$ by a new p.d.f. $g(x)$, given by

$$g(x) = \frac{\exp(-\theta x) f(x)}{f^*(\theta)} \qquad (\theta > 0).$$

This is an exponential family distribution that possesses moments of all orders.

When exponential tilting is applied to the positive stable distribution, the resulting distribution, with parameters α, γ and θ, is termed the tempered stable distribution (Hougaard, 1986); see also Tweedie (1984). The Laplace transform of the p.d.f. of the tempered stable distribution is

$$f^*(s) = \exp\left[-\xi\left\{(\theta + s)^\alpha - \theta^\alpha\right\}\right],$$

where α and ξ are as defined as in Section 4.2 and $\theta > 0$. The mean and variance of the distribution are given by $\mu = \alpha\xi/\theta^{1-\alpha}$ and $\sigma^2 = \alpha(1-\alpha)\xi/\theta^{2-\alpha}$. We denote the distribution by $\mathrm{TS}(\mu, \nu, \alpha)$, where $\nu = \sigma/\mu$ is the coefficient of variation. This notation is used by Palmer et al. (2008), who use the distribution to model cell generation times.

The characteristic function satisfies the conditions necessary for Devroye's algorithm to be applicable.

Several methods have been proposed to generate variables from a tempered stable distribution. First, it is clear from the exponential tilting process that generated the distribution that the following rejection algorithm can be used (Brix, 1999):

Algorithm BRIX

1. Generate a random value x, from the positive stable distribution, with parameters α and γ. Generate a uniform $U(0,1)$ random variable u.
2. If $u > \exp(-\theta x)$ go to step 1. Otherwise return x.

As described in Section 4.2, positive stable random variables can be simulated efficiently using the algorithm of Chambers et al. (1976). However, the expected number of times that step 1 of algorithm BRIX is evaluated, which is given by $\exp(\xi\theta^\alpha)$, may be prohibitively large; for example, for the parameter values $\alpha = 0.5, \theta = 2, \xi = 11.3137$, for which the distribution has mean $\mu = 4$ and standard deviation $\sigma = 1$, the expected number of positive stable random variables that must be generated to obtain a single tempered stable variable by this method is 8.89×10^6.

The tempered stable *process*, a Lévy process with tempered stable increments is of interest in financial modelling (Schoutens, 2003). Rosiński (2007b) presents an algorithm for simulating this process, based on a series representation; for an accessible introduction to this and other approaches to simulating Lévy processes, see Rosiński (2007a). Note that Rosiński uses the term tempered stable to denote a wider class of distributions than that considered here. The algorithm generates a process in continuous time over the time interval $[0, T]$ that converges uniformly from below to a realisation of the true process as the controlling parameter K increases.

Algorithm ROSIŃSKI

1. Set K to an integer value.
2. For $j = 1, \ldots, K$, generate independent variables u_j, $\tilde{u}_j$, v_j and w_j, where the u_j and $\tilde{u}_j$ are independent $U(0,1)$, the v_j are independent Exp(1) and the w_j are the successive arrival times of a Poisson process of rate 1.
3. For $0 < t \leq T$, return $x^{(K)}(t)$, where

$$x^{(K)}(t) = \sum_{j=1}^{K} \min\left\{ \left(\frac{T\xi}{\Gamma(1-\alpha)w_j} \right)^{1/\alpha}, \frac{v_j u_j^{1/\alpha}}{\theta} \right\} I_{(T\tilde{u}_j < t)}.$$

Figure 1 illustrates the behaviour of the algorithm as K increases. The process is simulated over $T = 200$ time units and the increment of the process per unit time follows the TS$(1, 1, 0.75)$ distribution. Increments of the process $x^{(K)}(t)$ over successive unit time steps therefore provide an approximation to a random sample of size 200 from the TS$(1, 1, 0.75)$ distribution. However, Figure 1 suggests that the quality of the approximation will be poor unless K is large.

To investigate this further, we compared the use of RLAPTRANS, RDEVROYE and the algorithm ROSIŃSKI, with $K = 10000, 25000, 50000$ or 100000, to generate (approximate) random samples from the distributions TS$(1, 1, 0.5)$ and TS$(1, 1, 0.75)$; the former distribution is an inverse Gaussian distribution. Five hundred random samples of size 200 were generated by each method and Table 5 shows the means of the 500 sample means from each method. There is little bias for $\alpha = 0.5$, but for $\alpha = 0.75$, the mean from Rosiński's method is biased downwards, even with $K = 100000$. Moreover, RLAPTRANS was faster than Rosiński's method even with $K = 10000$.

Another method of simulating from the tempered stable distribution in R is to use the function `rtweedie`, which is part of the R package Tweedie. This also uses the inversion algorithm and has the form

```
rtweedie(n, power, mu, phi)
```

A random sample of size n from the tempered stable distribution TS(μ, ν, α) may be generated by setting `power` to be $(2-\alpha)/(1-\alpha)$, `mu` to be μ and `phi` to be $(1-\alpha)(\xi\alpha)^{-1/(1-\alpha)}$. However, to generate a random sample of size 200 from the TS$(1, 1, 0.5)$ distribution, this method was over 400 times slower than RLAPTRANS. The difference in speed occurs primarily in calculation of the c.d.f., which, for these parameter values, is based on numerical inversion of the characteristic function (Dunn and Smyth, 2008).

4.4 Nested Archimedean copula models

Copulas are multivariate distribution functions for dependent random variables with standard uniform marginal distributions (Nelsen, 1999). One important class of copula models, the Archimedean copulas, is closely linked to Laplace transforms. Specifically, let $\psi(t)$ be the Laplace-Stieltjes transform of a probability distribution on $[0, \infty)$ that does not have a point mass at zero, but is otherwise unrestricted. Then the function $C : [0,1]^d \to [0,1]$ defined by

$$C(u_1, \ldots, u_d) = \psi\left\{\psi^{-1}(u_1) + \ldots + \psi^{-1}(u_d)\right\} \qquad (5)$$

is a d-dimensional Archimedean copula (Nelsen, 1999, p. 106). For example, the Clayton copula is given by the function $\psi(s) = (1+s)^{-1/\theta}$, the Laplace transform of a gamma distribution.

Marshall and Olkin (1988) gave the following algorithm to sample from an Archimedean copula:

Algorithm ACOP

1. Generate a random variable V from the distribution with Laplace transform ψ;
2. Generate i.i.d. $U(0,1)$ variables $X_1, \ldots, X_d$;
3. Return $U_1, \ldots, U_d$ where $U_j = \psi\{-\log(X_j)/V\}$.

The algorithm RLAPTRANS can be used for step 1 provided that V is absolutely continuous. The possibility of using transform inversion in this context is mentioned also by Whelan (2004) and recent work by Hofert (2008) discusses the simulation of Archimedean copulas using numerical transform inversion in detail, though focusing on transform inversion algorithms different to that used here.

A limitation of Archimedean copulas is that they are exchangeable. For example, for the Clayton copula, all pairs of variables have Kendall's rank correlation $\theta/(\theta+2)$. Greater flexibility is provided by the class of nested Archimedean copulas (Joe, 1997; McNeil, 2008) in which, whilst the bivariate marginal distribution functions are Archimedean copulas, the degree of dependence is not the same for all pairs. Algorithms for simulating from these models are discussed by Hofert (2008) and McNeil (2008).

Rather than treat the general case, which is complicated notationally, we focus on a particular 4-variable example. Suppose that ψ_1, ψ_2 and ψ_3 are Laplace transform generators of Archimedean copulas as described above and that the composite functions $\psi_1^{-1} \circ \psi_2$ and $\psi_1^{-1} \circ \psi_3$ have derivatives that are completely monotone. Then the following algorithm (Algorithm 4 from McNeil, 2008) generates random variables $U_1, \ldots, U_4$ that are not exchangeable. Instead, U_1 and U_2 have one joint distribution, U_3 and U_4 have another, and all other pairwise distributions are of a third type.

Algorithm NACOP

1. Generate a random variable V from the distribution with Laplace transform ψ_1;
2. Generate (X_1, X_2) from the bivariate Archimedean copula with generator $\psi_{2.1}(\cdot\,; V) = \exp\left[-V\psi_1^{-1} \circ \psi_2(\cdot)\right]$ using algorithm ACOP above;
3. Generate (X_3, X_4) from the bivariate Archimedean copula with generator $\psi_{3.1}(\cdot\,; V) = \exp\left[-V\psi_1^{-1} \circ \psi_3(\cdot)\right]$ using algorithm ACOP above;
4. Return $U_1, \ldots, U_4$ where $U_j = \psi_1\{-\log(X_j)/V\}$.

The technical conditions imposed upon ψ_1, ψ_2 and ψ_3 ensure that the generators $\psi_{2.1}$ and $\psi_{3.1}$ are themselves Laplace transforms of probability distributions and hence steps 1–3 of this algorithm all require generation of a random variable with a given Laplace transform. However, whereas step 1 can be done efficiently, because the distribution of V is fixed, steps 2 and 3 are much less efficient because the transform depends on the random value of V and only a single sample is required for each value of V.

For illustration, McNeil (2008) considers $\psi_i(s) = (1+s)^{-1/\theta_i} (i = 1, \ldots, 3)$. In order to satisfy the complete monotonicity conditions we require $\theta_2 > \theta_1$ and $\theta_3 > \theta_1$. Then U_1 and U_2 are distributed as a bivariate Clayton copula with paramater θ_2, U_3 and U_4 are distributed according to a bivariate Clayton copula with parameter θ_3, and all other pairs are distributed as bivariate Clayton copula with parameter θ_1. McNeil (2008) generates 3000 samples with $\theta_1 = 1$, $\theta_2 = 3$ and $\theta_3 = 8$ using the algorithm NACOP. It transpires that steps 2 and 3 of the algorithm require generation of random variables from a tempered stable distribution, as discussed in Section 4.3.

McNeil's algorithm can be implemented so that the user need only supply the functions $\psi_1(s), \psi_2(s)$ and $\psi_3(s)$, by using RLAPTRANS in steps 1–3. This code took approximately 30 seconds to generate 3000 multivariate samples; the simulated variables are plotted in Figure 2.

As a second example, McNeil (2008) generated 3000 samples from a 7-dimensional Gumbel copula, with a different dependence structure that we do not discuss here. These simulations took about one minute using Algorithm 6 in McNeil (2008), in conjunction with RLAPTRANS.

5 Discussion

The inversion method is perhaps the most basic method of random number generation and its implementation via numerical transform inversion constitutes a brute force approach to the problem of generating random numbers from an absolutely continuous positive distribution that is known only through its Laplace transform. Nonetheless, the examples in Section 4 show that this approach can be quite effective. Of course, for large-scale simulations it may still be worthwhile to develop more specialised algorithms for particular problems.

The device of sorting the uniform random numbers that is adopted in RLAPTRANS to improve the efficiency of the modified Newton-Raphson algorithm is not the only possibility. For example, it may sometimes be more efficient to create and store a table of values of $F^{-1}(v_j)$ for a regular grid of values v_j and use these as starting values for the Newton-Raphson algorithm. With a sufficiently fine grid, the Newton-Raphson step could be replaced by linear or spline interpolation.

The algorithm RLAPTRANS is designed to be very easy to use, but this of course carries inherent dangers. One important limitation stems from the fact that the algorithm of Abate et al. (2000) is a particular instance of the class of Fourier series methods for Laplace

transform inversion. Such methods are susceptible to the Gibbs phenomenon near discontinuities and this is a problem particularly when working with distributions defined on a finite range if the p.d.f. does not decline smoothly to zero at the end-points, though discontinuities at the origin are immune from this problem (Sakurai, 2004). In these circumstances, the algorithm performs poorly near the end-points of the distribution and may generate spurious values that are outside the allowable range. It is not sufficient simply to delete such values because the c.d.f. remains poorly approximated within the range of the distribution. Sakurai (2004) describes a modified algorithm (EULER-GPS) that can approximate the c.d.f. accurately over most of the range, but not in the immediate vicinity of a discontinuity. Several other techniques have been developed for trying to resolve the Gibbs phenomenon and Tadmor (2007) provides a recent review. The development of an efficient, reliable and numerically stable algorithm for generating random samples for distributions on a finite range from their Laplace transform remains a challenge for future research.

Aside from this, we believe that the algorithm should be suitable for most distributions. As we have noted with positive stable distributions, exceptions may occur for distributions that are extremely peaked but also have very long tails. If the nature of the distribution is unknown, it would be sensible as a preliminary step to plot the p.d.f. and c.d.f. (obtained by transform inversion) to check for signs of pathological behaviour. The performance of numerical transform inversion in the extreme tails of the distribution can often be improved by employing a scaling device due to Choudhury and Whitt (1997); an example of this improvement for the tempered stable distribution is provided by Palmer et al. (2008). Although this modification of the basic method, which necessitates solving a saddlepoint equation, would slow down the algorithm considerably it should probably be considered essential if the objective of the simulations is to study extreme values in very large samples.

Despite these cautionary comments, the varied examples presented in this paper indicate that the algorithm RLAPTRANS can generally provide a reliable source of random numbers in reasonable computational time and with minimal user input. Though less elegant than the algorithm of Devroye (1981), it is applicable to a wider range of problems and is usually faster, particularly when large samples are required, due to the beneficial effects of the sorting procedure employed. The performance of Devroye's algorithm can of course be improved in particular examples if analytical expressions are available for the constants c and k.

Acknowledgements I am grateful to Marius Hofert for helpful discussion and to the referees and coordinating editor whose comments helped me to improve the presentation.

References

J. Abate, G.L. Choudhury, and W. Whitt. An introduction to numerical transform inversion and its application to probability models. In W. Grassmann, editor, *Computational Probability*, pages 257–323. Kluwer, 2000.

J. Abate and P.P. Valko. Multi-precision Laplace transform inversion. *International Journal for Numerical Methods in Engineering*, 60:979–993, 2004.

J. H. Ahrens and U. Dieter. Computer methods for sampling from gamma, beta, Poisson and binomial distributions. *Computing*, 12:223–246, 1974.

J. H. Ahrens and U. Dieter. Generating gamma variates by a modified rejection technique. *Communications of the ACM*, 25:47–54, 1982.

O.E. Barndorff-Nielsen and D.R. Cox. *Asymptotic Techniques for Use in Statistics*. Chapman and Hall, 1989.

A. Brix. Generalized gamma measures and shot-noise Cox processes. *Advances in Applied Probability*, 31, 1999.

J.M. Chambers, C.L. Mallows, and B.W. Struck. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71:340–344, 1976.

G.L. Choudhury and W. Whitt. Probabilistic scaling for the numerical inversion of nonprobability transforms. *INFORMS Journal on Computing*, 9:175–184, 1997.

A.M. Cohen. *Numerical Methods for Laplace Transform Inversion*. Springer, 2007.

L. Devroye. On the computer generation of random variables with a given characteristic function. *Computers and Mathematics with Applications*, 7:547–552, 1981.

L. Devroye. Methods for generating random variates with Polya characteristic functions. *Statistics and Probability Letters*, 2:257–261, 1984.

L. Devroye. An automatic method for generating random variables with a given characteristic function. *SIAM Journal on Applied Mathematics*, 46:698–719, 1986a.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986b.

L. Devroye. Algorithms for generating discrete random variables with a given generating function or a given moment sequence. *SIAM Journal on Scientific and Statistical Computing*, 12:107–126, 1991.

P.K. Dunn and G.K. Smyth. Evaluation of Tweedie exponential dispersion model densities by Fourier inversion. *Statistics and Computing*, 18:73–86, 2008.

W.G. Feller. *An Introduction to Probability Theory and Its Applications*, volume 2. Wiley, 2nd edition, 1971.

M. Hofert. Sampling Archimedean copulas. *Computational Statistics and Data Analysis*, 52:5163–5174, 2008.

P. Hougaard. Survival models for heterogeneous populations derived from stable distributions. *Biometrika*, 73:387–396, 1986.

H. Joe. *Multivariate Models and Dependence Concepts*. Chapman and Hall, 1997.

E. Lukacs. *Characteristic Functions*. Griffin, 2nd edition, 1970.

A.W. Marshall and I. Olkin. Families of multivariate distributions. *Journal of the American Statistical Association*, 83:834–841, 1988.

J.H. McCulloch and D.B. Panton. Precise tabulation of the maximally-skewed stable distributions and densities. *Computational Statistics and Data Analysis*, 23:307–320, 1997.

A.J. McNeil. Sampling nested Archimedean copulas. *Journal of Statistical Computation and Simulation*, 78:567–581, 2008.

J.F. Monahan. *Numerical Methods of Statistics*. Cambridge University Press, 2001.

R. Nelsen. *An Introduction to Copulas*. Springer, 1999.

C.A. O'Cinneide. Euler summation for Fourier series and Laplace transform inversion. *Communications in Statistics: Stochastic Models*, 13:315–337, 1997.

K.J. Palmer, M.S. Ridout, and B.J.T. Morgan. Modelling cell generation times by using the tempered stable distribution. *Applied Statistics*, 57:379–397, 2008.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in Fortran*. Cambridge University Press, 2nd edition, 1992.

J. Rosiński. Simulation of Lévy processes. In F. Ruggeri, R. Kenett, and F. Faltin, editors, *Encyclopedia of Statistics in Quality and Reliability*. Wiley, 2007a.

J. Rosiński. Tempering stable processes. *Stochastic Processes and their Applications*, 117:677–707, 2007b.

T. Sakurai. Numerical inversion for Laplace transforms of functions with discontinuities. *Advances in Applied Probability*, 36:616–642, 2004.

W. Schoutens. *Lévy Processes in Finance*. Wiley, 2003.

E. Tadmor. Filters, mollifiers and the computation of the Gibbs phenomenon. *Acta Numerica*, 16:305–378, 2007.

A. Talbot. The accurate numerical inversion of Laplace transforms. *IMA Journal of Applied Mathematics*, 23:97–120, 1979.

L.N. Trefethen, J.A.C. Weideman, and T. Schmelzer. Talbot quadratures and rational approximations. *BIT Numerical Mathematics*, 46:653–670, 2006.

M.C.K. Tweedie. An index which distinguishes between some important exponential families. In J. K. Ghosh and J. Roy, editors, *Statistics: Applications and New Directions: Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, pages 579–604. Indian Statistical Institute, 1984.

N.G. Ushakov. *Selected Topics in Characteristic Functions*. VSP, 1999.

N. Whelan. Sampling from Archimedean copulas. *Quantitative Finance*, 4:339–352, 2004.

J. Wimp. *Sequence Transformations and Their Applications*. Academic Press, 1981.

Table 1. *Performance of RDEVROYE and RLAPTRANS when generating a random sample of size n from the gamma distribution with shape parameter α and scale parameter $\beta = 1$. Elapsed times (seconds) are means of 50 runs. The columns headed* Transform inversions *give the average number of transform inversions per random value generated. For* RDEVROYE *this is the expected number; for RLAPTRANS it is the mean value obtained in 50 runs.*

		Mean elapsed time		Transform inversions	
α	n	RDEVROYE	RLAPTRANS	RDEVROYE	RLAPTRANS
5	1	0.002	0.002	2.08	7.58
5	10	0.011	0.008	2.08	3.64
5	100	0.090	0.044	2.08	2.32
5	1000	0.899	0.347	2.08	1.73
2.5	1	0.002	0.001	1.75	6.40
2.5	10	0.009	0.008	1.75	3.62
2.5	100	0.078	0.047	1.75	2.32
2.5	1000	0.760	0.352	1.75	1.73
1.25	1	0.003	0.001	2.22	5.78
1.25	10	0.012	0.007	2.22	3.79
1.25	100	0.098	0.046	2.22	2.32
1.25	1000	0.976	0.354	2.22	1.74
0.05	1	–	0.007	–	31.42
0.05	10	–	0.025	–	13.68
0.05	100	–	0.096	–	5.11
0.05	1000	–	0.485	–	2.45

Table 2. *Maximum relative error (e_{max}) and median relative error (e_{med}) in recovering a specified set of quantiles of the gamma distribution with shape parameter α and scale parameter $\beta = 1$ using the algorithm RLAPTRANS. The tolerance parameter, τ, was varied as indicated.*

α	τ	$\log_{10}(e_{max})$	$\log_{10}(e_{med})$
5.00	10^{-7}	-5.03	-7.91
5.00	10^{-8}	-5.03	-8.06
5.00	10^{-9}	-5.40	-8.10
5.00	10^{-10}	-5.40	-8.09
2.50	10^{-7}	-4.92	-7.78
2.50	10^{-8}	-4.92	-7.93
2.50	10^{-9}	-5.31	-7.96
2.50	10^{-10}	-5.31	-7.95
1.25	10^{-7}	-3.25	-7.63
1.25	10^{-8}	-4.81	-7.78
1.25	10^{-9}	-5.24	-7.82
1.25	10^{-10}	-5.24	-7.85
0.05	10^{-7}	-2.49	-6.65
0.05	10^{-8}	-4.96	-6.80
0.05	10^{-9}	-4.96	-6.93
0.05	10^{-10}	-4.99	-6.93

Table 3. *Performance of* RLAPTRANS *when generating a random sample of size* 100 *or* 1000 *from the positive stable distribution with parameters* α *and* $\gamma = 1$. *Elapsed times (seconds) are means of 50 runs. The columns headed* Transform inversions *give the average numbers of transform inversions per random value generated. They are the mean values obtained in 50 runs.*

	Mean elapsed time		Transform inversions	
α	$n = 100$	$n = 1000$	$n = 100$	$n = 1000$
0.1	0.116	0.589	5.05	2.41
0.2	0.088	0.518	3.79	2.14
0.3	0.080	0.491	3.35	2.03
0.4	0.072	0.479	3.07	1.97
0.5	0.067	0.464	2.91	1.93
0.6	0.062	0.457	2.66	1.88
0.7	0.062	0.456	2.61	1.86
0.8	0.061	0.451	2.59	1.84
0.9	0.062	0.479	2.62	1.84

Table 4. *Relative error of RLAPTRANS in recovering specified quantiles, x_q, of the positive stable distribution with parameters α and $\gamma = 1$. The parameters A, ℓ and τ were varied as indicated.*

			$(A = 19, \ell = 1)$		$(A = 24, \ell = 2)$	
α	q	x_q	$\tau = 10^{-07}$	$\tau = 10^{-10}$	$\tau = 10^{-07}$	$\tau = 10^{-10}$
0.5	0.0001	0.066	-6.27	-6.74	-6.15	-7.92
0.5	0.01	0.151	-6.02	-7.68	-6.01	-9.11
0.5	0.5	2.198	-7.73	-7.73	-9.33	-9.33
0.5	0.99	6365.9	-5.78	-5.95	-6.27	-8.13
0.5	0.9999	63653798	-3.89	-3.95	-4.77	-6.13
0.7	0.0001	0.547	-7.01	-6.12	-6.08	-7.95
0.7	0.01	0.787	-7.54	-7.54	-9.19	-9.19
0.7	0.5	2.816	-6.71	-7.95	-6.74	-9.43
0.7	0.99	472.7	-6.10	-6.10	-7.79	-7.79
0.7	0.9999	334462.9	-3.69	-4.12	-3.91	-5.52
0.9	0.0001	4.390	-6.34	-6.15	-5.02	-5.03
0.9	0.01	4.834	-7.79	-7.79	-6.13	-6.13
0.9	0.5	6.966	-7.73	-8.55	-5.68	-5.69
0.9	0.99	116.6	-6.24	-6.24	-8.35	-8.35
0.9	0.9999	17903.4	-4.17	-4.21	-5.23	-6.39

Table 5. *Mean values of the sample mean from 500 independent random samples of size 200, generated by different simulation algorithms from two different tempered stable distributions. Standard errors are given in brackets. The true mean of the distribution is one for both distributions.*

	Distribution	
Algorithm	TS$(1, 1, 0.5)$	TS$(1, 1, 0.75)$
RLAPTRANS	1.002 (0.0029)	1.002 (0.0031)
RDEVROYE	1.000 (0.0032)	0.995 (0.0031)
ROSIŃSKI ($K = 10000$)	0.995 (0.0032)	0.863 (0.0031)
ROSIŃSKI ($K = 25000$)	0.992 (0.0032)	0.897 (0.0032)
ROSIŃSKI ($K = 50000$)	0.998 (0.0032)	0.921 (0.0031)
ROSIŃSKI ($K = 100000$)	0.996 (0.0031)	0.938 (0.0033)

Fig. 1 Successive approximations to a realisation of a Lévy process with tempered stable increments using algorithm ROSIŃSKI with increasing values of K. The algorithm simulates a jump process, but here the values of the process at intervals of one time step are joined by straight lines to improve clarity. The inset shows the p.d.f. of the increment in a unit time step.

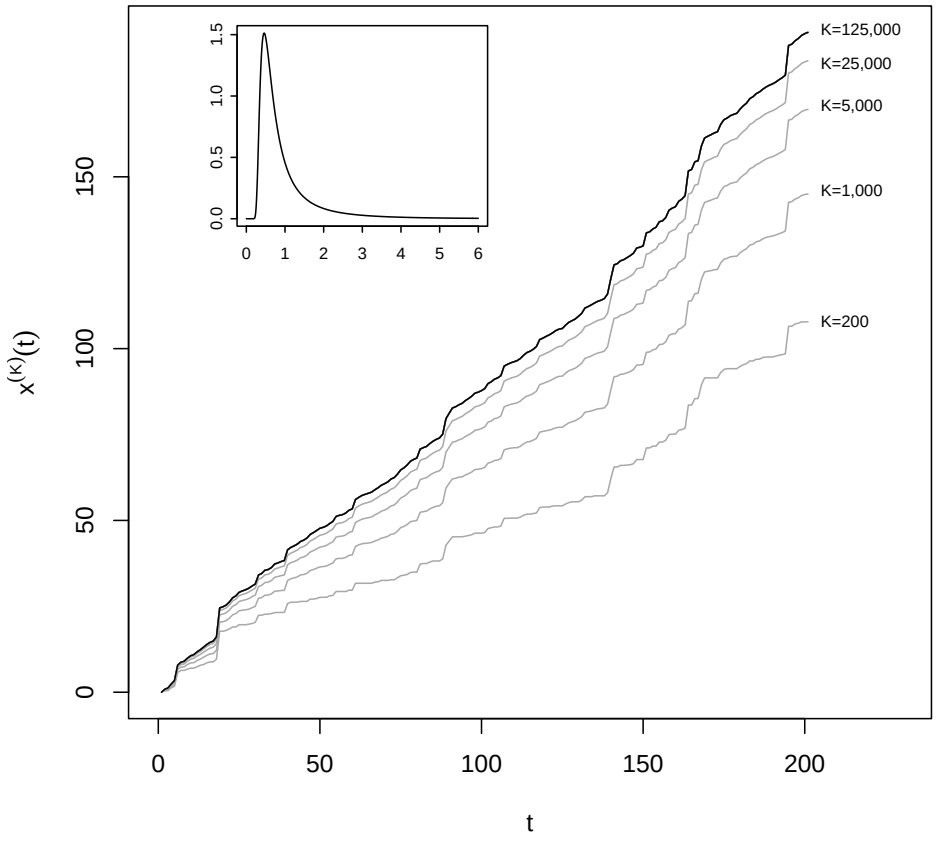

Fig. 2 Scatterplot matrix of variables $U_1, \ldots, U_4$ simulated from a nested Archimedean copula.