Draft revised entry for International Encyclopedia of Statistical Science, second edition, Lovric, Miodrag (Ed.), Springer-Verlag, 2024.

Non-Uniform Random Variate Generation

Pierre L'Ecuyer
Professor, DIRO
Université de Montréal, Montréal, QC, Canada
http://www.iro.umontreal.ca/~lecuyer/

Introduction

As explained in ▶Uniform Random Number Generators the simulation of random variables on a computer operates in two steps: In the first step, uniform random number generators produce imitations of i.i.d. $\mathcal{U}(0,1)$ (uniform over (0,1)) random variables, and in the second step these numbers are transformed in an appropriate way to imitate random variables or vectors from non-uniform distributions, and other types of random objects. Here we discuss the second step only, assuming that infinite sequences of i.i.d. $\mathcal{U}(0,1)$ random variables are available from the first step. This assumption is not realized exactly in software implementations, but good-enough approximations are available (L'Ecuyer, 2006; L'Ecuyer et al., 2021).

For some distributions, simple exact transformations from the uniform to the target distribution are available, often based on the inversion method. But for many types of distributions and processes, in particular those having shape parameters, and multivariate distributions, one relies on approximations that require a compromise between efficiency and approximation error. That is, the sampling is not always done exactly from the target distribution, but the discrepancy between the sampling and target distributions can often be made smaller with more work. This work may include a one-time setup cost to compute constants and tables that depend on the distribution parameters, and a marginal cost for each random variate generated from this distribution. The marginal speed and the quality of the approximation can often be improved by a larger investment in the setup time, sometimes to precompute larger tables. This investment can be worthwhile when a large number of random variates has to be generated from the same distribution, with the same shape parameters. Robustness with respect to shape parameters is another issue: Some methods provide a good approximation only in a certain range of values of these parameters, so one must be careful not to use the generator outside that range.

Inversion should be the preferred method whenever it is feasible and not too inefficient, because of its compatibility with important variance-reduction techniques such as common random numbers, antithetic variates, randomized quasi-Monte Carlo, and sample average approximation for optimization (Law, 2014; L'Ecuyer, 2009, 2018, 2023b). The rejection method is the most common alternative when inversion is inconvenient or

too slow. It is sometimes much faster than inversion, and applies in different settings.

These exact methods are not always sufficient, in particular for complicated multivariate distributions for which the density is often known only up to a multiplicative constant. ▶Markov chain Monte Carlo (MCMC) methods often provide an approximate solution in this case, by constructing an artificial Markov chain (see ▶Markov chains) whose stationary distribution is the target distribution. A random variate that follows approximately the target distribution is obtained by returning the current state after running the Markov chain long enough. See also ▶Monte Carlo methods in statistics.

In the remainder, we briefly summarize the basic techniques for univariate and multivariate distributions, and some types of stochastic processes. More detailed coverages can be found in Devroye (1986, 2006) and Hörmann et al. (2004).

Inversion

A general transformation that provides a univariate random variable X having the cumulative distribution function (cdf) F from a $\mathcal{U}(0,1)$ random variable U is $X = F^{-1}(U)$, where $F^{-1} : [0,1] \to \mathbb{R}$ is the inverse distribution function, defined as

$$F^{-1}(u) \stackrel{\text{def}}{=} \inf\{x \in \mathbb{R} \mid F(x) \geq u\}.$$

This is the *inversion method*.

As an illustration, if X is a binomial random variable with parameters $(n,p) = (2, 0.4)$, then we have $P[X = i] = p_i$ where $p_0 = (0.6)^2 = 0.36$, $p_1 = 2 \times 0.6 \times 0.4 = 0.48$, $p_2 = (0.4)^2 = 0.16$, and $p_i = 0$ elsewhere. Inversion then returns $X = 0$ if $U < 0.36$, $X = 1$ if $0.36 \leq U < 0.84$, and $X = 2$ if $U \geq 0.84$. Another simple way of generating X here is to generate two Bernoulli random variables X_1 and X_2 by inversion from two independent uniforms U_1 and U_2, i.e., $X_1 = \mathbb{I}[U_1 < p]$ and $X_2 = \mathbb{I}[U_2 < p]$, where $\mathbb{I}$ is the indicator function, and return $X = X_1 + X_2$. This method requires two uniforms and is not inversion for X.

For certain distributions, there is a closed-form formula for F^{-1}. For example, if X has a discrete uniform distribution over $\{0, \ldots, k-1\}$, we have $X = F^{-1}(U) = \lfloor kU \rfloor$. If X has a ▶geometric distribution with parameter p, so $P[X = x] = p(1-p)^x$ for $x = 0, 1, \ldots$, we have $X = F^{-1}(U) = \lceil \ln(1-U)/\ln(1-p) \rceil - 1 = \lfloor \ln(1-U)/\ln(1-p) \rfloor$ with probability 1. If X is exponential, with rate λ, then $X = F^{-1}(U) = -\ln(1-U)/\lambda$. To generate X with cdf F but truncated to an interval $(a, b]$, it suffices to generate U uniform over $(F(a), F(b)]$, and to return $F^{-1}(U)$.

For certain distributions there is no closed-form expression for F^{-1} but good numerical approximations are available. For distributions having a location and a scale parameter, we only need a good approximation of F^{-1} for the standardized form of

the distribution, say with location at 0 and scale 1. We generate a variable from the standardized distribution, then multiply by the scale parameter and add the location parameter. This applies in particular to the normal distribution, for which good numerical approximations of the standard inverse cdf Φ^{-1} are available. For example, the `probdist` package of SSJ (L'Ecuyer, 2023a) implements a slight modification of a rational Chebyshev approximation proposed by Blair et al. (1976), which is quite fast and provides essentially machine-precision accuracy when using floating point numbers with 53 bits of precision. For any $u \in (0, 1)$ that can be represented by such a floating-point number (given as input), the approximation procedure returns $\Phi^{-1}(u)$ with relative error smaller than 10^{-15}.

In general, given an approximation $\tilde{F}^{-1}$ of F^{-1}, the absolute error on X for a given U is $|\tilde{F}^{-1}(U) - F^{-1}(U)|$. The corresponding error on U is $|F(\tilde{F}^{-1}(U)) - U|$. This second error can hardly be less than the error in the representation of U, and we are also limited by the machine precision on the representation of X. If one of these two limits is reached for each $U \in [0, 1]$, for practical purposes we have an exact inversion method.

When shape parameters are involved (e.g., for the gamma and beta distributions), things are more complicated because a different approximation of F^{-1} must be constructed for each choice of shape parameters.

When we have an algorithm for computing F but not F^{-1}, and F is continuous, as a last resort we can always approximate $X = F^{-1}(U)$ by a numerical method that finds a root of the equation $U = F(X)$ for a given U. For instance, we can run the robust Brent-Dekker iterative root finding algorithm of (Brent, 1973, Chapter 4) until we have reached the required precision, as done by default in SSJ (L'Ecuyer, 2023a).

Faster inversion algorithms for fixed shape parameters can be constructed if we are ready to invest in setup time. These methods are called *automatic* when the code that approximates F^{-1} is produced automatically by a general one-time setup algorithm (Hörmann et al., 2004). The setup computes tables that contain the interpolation points and coefficients. With these tables in hand, random variate generation is very fast. For example, a general adaptive method that constructs an accurate Hermite interpolation method for F^{-1}, given a function that computes F, is developed by Hörmann and Leydold (2003). Derflinger et al. (2010) propose an algorithm that constructs an approximation of F^{-1} to a given accuracy (specified by the user) for the case where only the density of X is available. This algorithm is an improvement over similar methods by Ahrens and Kohrt (1981). These methods assume that the distribution has bounded support, but they can be applied to most other distributions by truncating the tails far enough for the error to be negligible.

For discrete distributions, say over the values $x_1 < \cdots < x_k$, inversion finds $I = \min\{i \mid F(x_i) \geq U\}$ and return $X = x_I$. To do this repeatedly for several U, one may first

tabulate the pairs $(x_i, F(x_i))$ for $i = 1, \ldots, k$, and then for each U, find I by sequential or binary search in the table (L'Ecuyer, 2023b). However, the fastest implementation when k is large is obtained by using an index (Chen and Asau, 1974; Devroye, 1986). The idea is to partition the interval $(0, 1)$ into c subintervals of equal sizes, $[j/c, (j+1)/c)$ for $j = 0, \ldots, c-1$, and store the smallest and largest possible values of X for each subinterval, namely $L_j = F^{-1}(j/c)$ and $R_j = F^{-1}((j+1)/c)$. Once U is generated, we find the corresponding interval number $J = \lfloor cU \rfloor$ by direct calculation, and search for I only in that interval, with linear or binary search. The fastest average time per call is usually obtained by taking a large c (so that k/c does not exceed a few units), and linear search in the subintervals (to minimize the overhead). The resulting algorithm is as fast (on average) as the alias method (Law, 2014; Walker, 1974), which is often presented as the fastest algorithm but does not preserve inversion. If k is large or even infinite, for example for the ▶Poisson distribution or the ▶binomial distribution with a large n, the pairs $(x_i, F(x_i))$ are precomputed and tabulated only in the areas where the probabilities are not too small, usually around the center of the distribution. Other values are computed dynamically only in the very rare cases where they are needed. Similar indexing techniques can also be used for piecewise-polynomial approximations of F^{-1} for continuous distributions.

Rejection Methods and Thinning

When inversion is too costly, the best alternative is often a rejection method. It works as follows. Suppose we want to generate X from density f. It suffices to know f up to a multiplicative constant, i.e., to know κf, where κ might be unknown. If f is known, we take $\kappa = 1$. We pick a density r such that $\kappa f(x) \le t(x) \stackrel{\text{def}}{=} a\, r(x)$ for all x for some constant a, and such that sampling random variates Y from r is easy. The function t is called a *hat function*. we see that we must have $\kappa \le a$. To generate X with density f, we generate Y from the density r and $U \sim \mathcal{U}(0, 1)$ independent of Y, repeat this until $U\, t(Y) \le \kappa f(Y)$, and return $X = Y$ (Devroye, 1986; von Neumann, 1951). The number of times we have to retry is a geometric random variable with mean $a/\kappa - 1 \ge 0$. We want a/κ to be as small as possible.

If κf is expensive to compute, computations can often be accelerated by using *squeeze functions* q_1 and q_2 that are less costly to evaluate and such that $q_1(x) \le \kappa f(x) \le q_2(x) \le t(x)$ for all x. After generating Y, we first check if $U\, t(Y) \le q_1(Y)$. If so we accept Y immediately. Otherwise if $U\, t(Y) \ge q_2(Y)$, we reject Y immediately. We verify the condition $U\, t(Y) \le \kappa f(Y)$ explicitly only when none of the two previous inequalities is satisfied. We may also use multiple levels of embedded squeezing, with crude squeezing functions that are very quick to evaluate at the first level, then tighter but slightly more

expensive ones at the second level, and so on.

In most practical situations, rejection is combined with a change of variable to transform the original density into a nicer one, for which a more efficient implementation of the rejection method can be constructed. The change of variable can be selected so that the transformed density is concave and a piecewise linear hat function is easy to construct. Typical examples of transformations can be $T(x) = \log x$ and $T(x) = -x^{-1/2}$, for instance (Devroye, 1986; Hörmann et al., 2004). The rejection method also works for discrete distributions; we just replace the densities by the probability mass functions.

One special case of change of variable combined with rejection leads to the *ratio-of-uniforms* method. It is based on the observation that if X has density f over $\mathbb{R}$, κ is a positive constant, and the pair (U, V) has the uniform distribution over the set

$$\mathcal{C} = \left\{(u, v) \in \mathbb{R}^2 \text{ such that } 0 \le u \le \sqrt{\kappa f(v/u)}\right\},$$

then V/U has the same distribution as X (Devroye, 1986; Kinderman and Monahan, 1977). Thus, one can generate X by generating (U, V) uniformly over $\mathcal{C}$, usually by a rejection method, and returning $X = V/U$.

A special form of rejection called *thinning* is frequently used to generate non-homogeneous ▶point processes. For example, suppose we want to generate the jump times of a ▶Poisson process whose time-varying rate is $\{\lambda(t),\, t \ge 0\}$, where $\lambda(t) \le \bar{\lambda}$ at all times t for some constant $\bar{\lambda}$. Then we can generate *pseudo-jumps* at constant rate $\bar{\lambda}$ by generating the times between successive jumps as i.i.d. exponentials with mean $1/\bar{\lambda}$. A pseudo-jump at time t is accepted (becomes a real jump) with probability $\lambda(t)/\bar{\lambda}$ (Law, 2014).

Multivariate Distributions

A d-dimensional *random vector* $\boldsymbol{X} = (X_1, \ldots, X_d)^\mathsf{t}$ has distribution function F if $\mathbb{P}[X_1 \le x_1, \ldots, X_d \le x_d] = F(x_1, \ldots, x_d)$ for all $\boldsymbol{x} = (x_1, \ldots, x_d)^\mathsf{t} \in \mathbb{R}^d$. The distribution function of a random vector does not have an inverse in general, so the inversion method does not apply directly to multivariate distributions. There are situations where one can generate X_1 directly by inversion from its marginal distribution, then generate X_2 by inversion from its marginal distribution conditional on X_1, then generate X_3 by inversion from its marginal distribution conditional on (X_1, X_2), and so on. But this is not always possible or convenient.

There are important classes of multivariate distributions for which simple and elegant methods are available. For example, suppose $\boldsymbol{X}$ has a ▶multinormal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. When $\boldsymbol{\mu} = \boldsymbol{0}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$ (the identity), we have a *standard multinormal* distribution. This one is easy to generate: the coordinates

are independent standard normals, and they can be generated separately by inversion. For the general case, it suffices to decompose $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, generate $\boldsymbol{Z}$ standard multinormal, and return $\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{Z}$. The most popular way to decompose $\boldsymbol{\Sigma}$ is the Cholesky decomposition, for which $\boldsymbol{A}$ is lower triangular, but there are other possibilities, including for example the eigendecomposition as in ▶principal component analysis. The choice of decomposition can have a large impact on the variance reduction in the context of randomized quasi-Monte Carlo integration, by concentrating much of the variance on just a few underlying uniform random numbers (Glasserman, 2004; L'Ecuyer, 2009, 2018).

Multivariate normals are useful for various purposes. For example, to generate a random point $\boldsymbol{X}$ on a sphere of radius 1 centered at zero in d dimensions, generate a standard multinormal vector $\boldsymbol{Z}$, then normalize its length to the desired radius: $\boldsymbol{X} = \boldsymbol{Z}/\|\boldsymbol{Z}\|$. This is equivalent to generating a *random direction*. A more general class of multivariate distributions named *radially symmetric* are defined by putting $\boldsymbol{X} = R\,\boldsymbol{Z}$ where R has an arbitrary distribution over $(0, \infty)$. A further generalization yields an *elliptic multivariate* random variable: $\boldsymbol{X} = \boldsymbol{\mu} + R\,\boldsymbol{A}\boldsymbol{Z}$ where $\boldsymbol{Z}$ is a standard multinormal in k dimensions and $\boldsymbol{A}$ is a $d \times k$ matrix. It is easy to generate $\boldsymbol{X}$ if we know how to generate R. As a special case, if $R^2 = \nu/Y$ where Y is chi-square with ν degrees of freedom, then $\boldsymbol{X}$ is multivariate Student with ν degrees of freedom.

The most general way to define multivariate distributions is via ▶copulas (Hörmann and Derflinger, 2002; Nelsen, 2006). Any multivariate cdf C over $(0,1)^d$ with uniform one-dimensional marginals is a copula. To generate a vector $\boldsymbol{X} = (X_1, \ldots, X_d)^{\mathsf{t}}$ with arbitrary marginal cdf's F_j and a dependence structure specified by the copula C, generate $\boldsymbol{U} = (U_1, \ldots, U_d)$ with cdf C, then put $X_j = F_j^{-1}(U_j)$ for each j.

One way to define a copula is to start with an arbitrary d-dimensional cdf G with continuous marginals G_j, generate $\boldsymbol{Y} = (Y_1, \ldots, Y_d)^{\mathsf{t}}$ from G, and let $\boldsymbol{U} = (U_1, \ldots, U_d) = (G_1(Y_1), \ldots, G_d(Y_d))^{\mathsf{t}}$. At this point, the U_j have the uniform distribution over $(0,1)$, but they are not independent in general. The cdf C of $\boldsymbol{U}$ is the *copula associated with* G. A popular choice for G is the multinormal cdf with standard normal marginals; then $\boldsymbol{Y}$ and $\boldsymbol{U}$ are easy to generate, and one can select the correlation matrix of $\boldsymbol{Y}$ to approximate a target correlation (or rank correlation) matrix for $\boldsymbol{X}$. This is a *normal copula*. It can usually match the correlations pretty well, but to capture the whole dependence structure in general, it is often not sufficient (Blum et al., 2002; Nelsen, 2006).

Archimedean copulas are a popular alternative class. They are defined as follows. One selects a generating function $\varphi : (0,1] \to \mathbb{R}$ which is d times continuously differentiable and satisfies $\varphi(u) \to \infty$ when $u \to 0^+$, $\varphi(1) = 0$, $\varphi'(u) < 0$ for all u, and

$(-1)^k d^k \varphi^{-1}(x)/dx^k > 0$ for all $x \in [0, \infty)$. The Archimedean copula is defined by

$$C(u_1, \ldots, u_d) = \begin{cases} \varphi^{-1}(\varphi(u_1) + \cdots + \varphi(u_d)) & \text{if } \varphi(u_1) + \cdots + \varphi(u_d) \le \varphi(0); \\ 0 & \text{otherwise.} \end{cases}$$

Examples of generating functions that depend on a single parameter are $\varphi(u) = u^{-\lambda} - 1$ for $\lambda > 0$ (Clayton's copula), $\varphi(u) = (-\ln u)^\lambda$ for $\lambda > 1$ (the Gumbel copula), and $\varphi(u) = -\ln[(e^{-\lambda u} - 1)/(e^{-\lambda} - 1)$ for $\lambda \neq 0$ (Frank's copula). Marshall and Olkin (1988) have shown that if Y is a positive continuous random variable whose density is the inverse of the Laplace transform of φ, $\boldsymbol{V} = (V_1, \ldots, V_d) \sim \mathcal{U}(0,1)^d$ a vector of independent uniforms, and $U_j = \varphi((\ln V_j)/Y)$ for $j = 1, \ldots, d$, then $\boldsymbol{U} = (U_1, \ldots, U_d)$ has the distribution of the Archimedean copula C with generating function φ. This provides an easy way of generating random vectors from such a copula. A much larger variety of copulas and details about their sampling can be found in Nelsen (2006); Hofert (2008); Mai and Scherer (2012).

The rejection method extends rather straightforwardly to multivariate distributions. For a known target d-dimensional density f, pick a d-dimensional density r such that $f(\boldsymbol{x}) \le a\, r(\boldsymbol{x})$ for all $\boldsymbol{x}$ and some constant a, and such that sampling random vectors $\boldsymbol{Y}$ from r is easy. To generate $\boldsymbol{X}$ with density f, generate $\boldsymbol{Y}$ from r and $U \sim \mathcal{U}(0,1)$ independent of $\boldsymbol{Y}$, until $U a\, r(\boldsymbol{Y}) \le f(\boldsymbol{Y})$, and return $\boldsymbol{X} = \boldsymbol{Y}$.

A practical limitation of this simple and direct approach is that it is often too hard to find an "easy" r for which the acceptance probability is large enough, especially when the dimension d is large. A popular alternative in this case is MCMC, mentioned earlier. More generally, MCMC is often used to sample (approximately) from a simple distribution, but conditionally on some event B which is often a *rare event* (it has a very small probability). The artificial Markov chain is then defined so its state space is the set B (the set of states where the rare event occurs) and its equilibrium (or steady-state) distribution is the desired conditional distribution given B. The key challenge is to construct a chain that mixes fast enough, i.e., for which the convergence to the steady-state distribution occurs quickly, from any given starting state. This is not always easy. Another method to sample approximately conditionally on a rare event B is *generalized splitting* (Botev et al., 2016; Botev and L'Ecuyer, 2020; L'Ecuyer et al., 2018). It assumes that we can define an *importance function* $S : \mathbb{R}^d \to \mathbb{R}$ such that $B = \{S(\boldsymbol{X}) \ge \ell\}$ for some constant ℓ. We select an integer splitting factor $s \ge 2$ (usually $s = 2$) and a sequence of levels $-\infty = \ell_0 < \ell_1 < \cdots < \ell_\tau = \ell$, usually estimated by pilot runs, such that $\mathbb{P}[S(\boldsymbol{X}) \ge \ell_j \mid S(\boldsymbol{X}) \ge \ell_{j-1}] \approx 1/s$ for $j = 1, \ldots, \tau$. We also define a Markov chain (as in MCMC) whose stationary distribution is the distribution of $\boldsymbol{X}$ conditional on $\{S(\boldsymbol{X}) \ge \ell_j\}$, for each j. At level 0, we start by sampling a single state $\boldsymbol{X}$ from

its original distribution. At each level j, for $0 \le j < \tau$, we have a set of N_j states $\boldsymbol{X}$ for which $S(\boldsymbol{X}) > \ell_j$. We resample all these states independently conditional on $S(\boldsymbol{X}) > \ell_j$ by running the appropriate Markov chain for s steps, to obtain sN_j new states. We retain the new states for which $S(\boldsymbol{X}) > \ell_{j+1}$, discard the others, and go to level $j+1$. We collect the states that we get at level $j = \tau$. In one version, this process is repeated independently until we have collected at least n states at level τ in total. Botev and L'Ecuyer (2020) prove that the total variation distance between the empirical distribution of these states and the desired conditional distribution converges as $\mathcal{O}(n^{-3/2})$.

Stochastic Processes

Various types of ▶stochastic processes can be simulated in a way that becomes obvious from their definition. ▶Lévy processes form an important class; they are continuous-time stochastic processes $\{Y(t),\, t \ge 0\}$ with $Y(0) = 0$ and whose increments over disjoint time intervals are independent, and for which the increment over a time interval of length t has a distribution that depends only on t (the mean and standard deviation must be proportional to t) (Bertoin, 1996; Asmussen and Glynn, 2007; Barndorff-Nielsen et al., 2013). Special instances include the (univariate or multivariate) ▶Brownian motion, the stationary ▶Poisson process, the *gamma process*, and the *inverse Gaussian process*, for example, whose increments have the multinormal, Poisson, gamma, and inverse Gaussian distributions, respectively. A natural way to generate a Lévy process observed at times $0 = t_0 < t_1 < \cdots < t_c$ is to generate the independent increments $Y(t_j) - Y(t_{j-1})$ successively, for $j = 1, \ldots, c$. This is the *random walk* method. For the special instances just mentioned, this is easy to do.

For certain Lévy processes (including those mentioned above), for any $t_1 < s < t_2$, we know how to generate $Y(s)$ from its distribution *conditional* on $\{Y(t_1) = y_1,\, Y(t_2) = y_2\}$ for arbitrary y_1, y_2. Then a sketch (or skeleton) of the trajectory can be generated via the following *Lévy bridge sampling* strategy, where we assume for simplicity that c is a power of 2. We start by generating $Y(t_c)$ from the distribution of the increment over $[0, t_c]$, then we generate $Y(t_{c/2})$ from its distribution conditional on $(Y(t_0), Y(t_c))$, then we apply the same technique recursively to generate $Y(t_{c/4})$ conditional on $(Y(t_0), Y(t_{c/2}))$, $Y(t_{3c/4})$ conditional on $(Y(t_{c/2}), Y(t_c))$, $Y(t_{c/8})$ conditional on $(Y(t_0), Y(t_{c/4}))$, and so on. This method is convenient if one wishes to later refine the approximation of a trajectory. It is also effective for reducing the effective dimension in the context of quasi-Monte Carlo methods (L'Ecuyer, 2009, 2018).

For the ▶Poisson process, one usually wishes to have the individual jump times, and not only the numbers of jumps in predetermined time intervals. For a stationary

Poisson process, the times between successive jumps are easy to generate, because they are independent exponential random variables. For a non-stationary Poisson process, one way is to apply a nonlinear time transformation to turn it into a standard stationary Poisson process of rate 1, generate the jumps times of the standard process, and apply the reverse time transformation to recover the jump times of the target non-stationary Poisson process (L'Ecuyer, 2023b). This idea applies to other continuous-time stochastic processes as well, as we now explain.

Given a process $X = \{X(t), t \geq 0\}$, and another process $T = \{T(t), t \geq 0\}$ with nondecreasing trajectories, called a *subordinator*, we can define a new process $Y = \{Y(t) \stackrel{\text{def}}{=} X(T(t)), t \geq 0\}$, which is the process X to which we have applied a random time change. This can be applied to any process X with index $t \in \mathbb{R}$. If both X and T are Lévy processes, then so is Y.

If X is a stationary Poisson process with rate 1 and we want a nonstationary Poisson process Y with rate function $\{\lambda(t), t \geq 0\}$, then we must take $T(t) = \Lambda(t) \stackrel{\text{def}}{=} \int_0^t \lambda(s)\mathrm{d}s$. To simulate Y, we generate the jump times $Z_1 < Z_2 < \cdots$ of the stationary process X by generating $Z_j - Z_{j-1}$ as independent exponentials with mean 1, and define the jump times of Y as $T_j = \Lambda^{-1}(Z_j)$ for $j \geq 1$.

If X is a one-dimensional ▶Brownian motion, the random time change is equivalent to replacing the constant volatility parameter σ of the Brownian motion by a stochastic (time-varying) volatility process $\{\sigma(t), t \geq 0\}$. Two well-known examples of Lévy processes that can act as subordinators are the gamma process and the inverse Gaussian process, respectively. Their use as subordinators for the Brownian motion yields the variance gamma and the normal inverse Gaussian processes (Barndorff-Nielsen et al., 2013). Brownian motions with a random time change are important because they provide a better fit to various types of financial data (such as the log prices of stocks and commodities, etc.) than standard Brownian motions.

Acknowledgment

This work has been supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant RGPIN-2018-05795 and a Canada Research Chair to the author.

About the author

See the entry ▶Uniform Random Number Generators.

References

Ahrens, J. H. and Kohrt, K. D. (1981). Computer methods for efficient sampling from largely arbitrary statistical distributions. *Computing*, 26:19–31.

Asmussen, S. and Glynn, P. W. (2007). *Stochastic Simulation*. Springer-Verlag, New York.

Barndorff-Nielsen, O. E., Mikosch, T., and Resnick, S. I. (2013). *Lévy Processes: Theory and Applications*. Birkhäuser.

Bertoin, J. (1996). *Lévy Processes*. Cambridge University Press, Cambridge.

Blair, J. M., Edwards, C. A., and Johnson, J. H. (1976). Rational Chebyshev approximations for the inverse of the error function. *Mathematics of Computation*, 30:827–830.

Blum, P., Dias, A., and Embrechts, P. (2002). The ART of dependence modelling: the latest advances in correlation analysis. In Lane, M., editor, *Alternative Risk Strategies*, pages 339–356. Risk Books, London.

Botev, Z. I. and L'Ecuyer, P. (2020). Sampling conditionally on a rare event via generalized splitting. *INFORMS Journal on Computing*, 32(4):986–995.

Botev, Z. I., L'Ecuyer, P., and Tuffin, B. (2016). Static network reliability estimation under the Marshall-Olkin copula. *ACM Transactions on Modeling and Computer Simulation*, 26(2):Article 14, 28 pages.

Brent, R. P. (1973). *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ.

Chen, H. C. and Asau, Y. (1974). On generating random variates from an empirical distribution. *AIEE Transactions*, 6:163–166.

Derflinger, G., Hörmann, W., and Leydold, J. (2010). Random variate generation by numerical inversion when only the density is known. *ACM Transactions on Modeling and Computer Simulation*, 20(4):Article 18.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, NY.

Devroye, L. (2006). Nonuniform random variate generation. In Henderson, S. G. and Nelson, B. L., editors, *Simulation*, Handbooks in Operations Research and Management Science, pages 83–121. Elsevier, Amsterdam, The Netherlands. Chapter 4.

Glasserman, P. (2004). *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York.

Hofert, M. (2008). Sampling Archimedean copulas. *Computational Statistics and Data Analysis*, 52(12):5163–5174.

Hörmann, W. and Derflinger, G. (2002). Fast generation of order statistics. *ACM Transactions on Modeling and Computer Simulation*, 12(2):83–93.

Hörmann, W. and Leydold, J. (2003). Continuous random variate generation by fast numerical inversion. *ACM Transactions on Modeling and Computer Simulation*, 13(4):347–362.

Hörmann, W., Leydold, J., and Derflinger, G. (2004). *Automatic Nonuniform Random Variate Generation*. Springer-Verlag, Berlin.

Kinderman, A. J. and Monahan, J. F. (1977). Computer generation of random variables using the ratio of uniform deviates. *ACM Transactions on Mathematical Software*, 3:257–260.

Law, A. M. (2014). *Simulation Modeling and Analysis*. McGraw-Hill, New York, fifth edition.

L'Ecuyer, P. (2006). Uniform random number generation. In Henderson, S. G. and Nelson, B. L., editors, *Simulation*, Handbooks in Operations Research and Management Science, pages 55–81. Elsevier, Amsterdam, The Netherlands. Chapter 3.

L'Ecuyer, P. (2009). Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics*, 13(3):307–349.

L'Ecuyer, P. (2018). Randomized quasi-Monte Carlo: An introduction for practitioners. In Glynn, P. W. and Owen, A. B., editors, *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2016*, pages 29–52, Berlin. Springer.

L'Ecuyer, P. (2023a). SSJ: Stochastic simulation in Java. `https://github.com/umontreal-simul/ssj`.

L'Ecuyer, P. (2023b). Stochastic simulation and Monte Carlo methods. Draft Textbook, `https://www-labs.iro.umontreal.ca/~lecuyer/ift6561/book.pdf`.

L'Ecuyer, P., Botev, Z. I., and Kroese, D. P. (2018). On a generalized splitting method for sampling from a conditional distribution. In *Proceedings of the 2018 Winter Simulation Conference*, pages 1694–1705. IEEE Press.

L'Ecuyer, P., Nadeau-Chamard, O., Chen, Y.-F., and Lebar, J. (2021). Multiple streams with recurrence-based, counter-based, and splittable random number generators. In *Proceedings of the 2021 Winter Simulation Conference*, pages 1–16. IEEE Press.

Mai, J.-F. and Scherer, M. (2012). *Simulating Copulas*. Imperial College Press.

Marshall, A. W. and Olkin, I. (1988). Families of multivariate distributions. *Journal of the American Statistical Association*, 83(403):834–841.

Nelsen, R. B. (2006). *An Introduction to Copulas*. Springer Series in Statistics. Springer, New York, NY, second edition.

von Neumann, J. (1951). Various techniques used in connection with random digits. In Householder et al., A. S., editor, *The Monte Carlo Method*, volume 12, pages 36–38. National Bureau of Standards, Applied Mathematics Series.

Walker, A. J. (1974). New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronic Letters*, 10:127–128.