

MONTE CARLO SIMULATION IN RELIABILITY

Lawrence M. Leemis
Department of Mathematics
William & Mary
Williamsburg, VA 23187-8795

Simulation is a generic term used loosely in engineering, with application areas ranging from flight simulators used in cockpit design to simulated annealing used in optimization. Simulation is presented here as a mathematical and computational technique used to analyze probabilistic models. Simulation can be divided into *Monte Carlo simulation*, in which *static* models are analyzed, and *discrete-event simulation*, in which *dynamic* models involving the passage of time are analyzed. The early work of Buffon, Gosset, Ulam, and others is described in Goldsman, Nance, and Wilson (1).

Monte Carlo simulation is often used when analytic methods become mathematically intractable, and they typically give a modeler added insight into the structure of a problem. As reliability and lifetime models become more complex and typically less mathematically tractable, Monte Carlo simulations have increasing importance. Monte Carlo simulation techniques mirror the relative frequency approach for determining probabilities. The estimate for the probability of interest converges to the true value as the number of replications increases. This article considers methods for generating random lifetimes and random processes from probabilistic models. The basic methods are inversion (inverse-cdf and inverse-chf), linear combination methods (composition and competing risks), majorizing methods (acceptance/rejection and thinning), and special properties.

The basic methods are followed by a discussion of order statistics. The generation of order statistics is useful for estimating measures of performance associated with series, parallel, and k -out-of- n systems. The accelerated life and proportional hazards lifetime models can account for the effects of covariates on a random lifetime. Variate generation for these models is a straightforward extension of the basic methods when the covariates do not depend on time. Variate generation algorithms for Monte Carlo simulation of nonhomogeneous Poisson processes are a simple extension of the inverse-chf technique. Methods for generating failure times for a repairable system modeled by a nonhomogeneous Poisson process are also reviewed.

1 PROBABILITY MODELS FOR LIFETIMES

In reliability modeling, a continuous positive random variable typically represents the lifetime of a component or system. The generic term *item* is used in this section to apply to either a component or a system. Several functions completely specify the distribution of a random variable. Five of these functions are useful in describing variate generation algorithms: cumulative distribution function (cdf), survivor function, probability density function (pdf), hazard function, and cumulative hazard function (chf). Other functions, not used here, include the mean residual life function (2), moment-generating function (3), and total time on test transform (4).

Although Monte Carlo simulation is applied here to problems in reliability, the techniques are applicable to any problem setting concerning random variables that can assume only positive values. Thus the setting here is actually *survival analysis* and includes fields as diverse as biostatistics, actuarial science, economics, and sociology.

This section considers techniques for generating random variates for Monte Carlo simulation analysis. Two textbooks [i.e., Devroye (5) and Dagpunar (6)] are devoted entirely to this topic. The

purpose here is to review algorithms capable of transforming random numbers to *random variates* possessing known probabilistic properties for use in reliability studies. With the generation of random variates as a basis, several other topics, namely, generating order statistics, generating lifetimes from models with covariates, and generating point processes, are considered.

In the interest of brevity, we assume that a source of randomness is available (i.e., a stream of independent random numbers). These random numbers are uniformly distributed between 0 and 1, and most high-level programming languages now include a random number generator. The random numbers are denoted by U and the random variates (lifetimes) are denoted by T . Algorithms for generating the random numbers and desirable properties associated with random number generators (such as insensitivity to parameter values, speed, memory requirements, relationship to variance reduction techniques) are reviewed by Schmeiser (7), as well as by many of the simulation textbooks that he references. Park and Miller (8) also overview Lehmer random number generators and L'Ecuyer (9) surveys more recent work in random number generation.

The discussion here is limited to generating *continuous*, as opposed to discrete or mixed, distributions. Generating variates from discrete distributions is useful for evaluation of certain types of reliability analysis tools such as fault trees. For simplicity, the examples are confined to the exponential and Weibull distributions, which have been chosen because of their tractability and widespread use. Any continuous lifetime distribution with a closed-form inverse-cdf could have been used. Reliability textbooks that discuss Monte Carlo simulation techniques include Foster, Phillips, and Rogers (10), Goldberg (11), Harr (12), Henley and Kumamoto (13), Leemis (14), Mann, Schafer, and Singpurwalla (15), and Rao (16).

The survivor function, also known as the reliability function and complementary cdf, is defined by

$$S(t) = P(T > t) \quad t \geq 0,$$

which is a nonincreasing function of t satisfying $S(0) = 1$ and $\lim_{t \rightarrow \infty} S(t) = 0$. The survivor function is important in the study of *systems* of components since it is the appropriate argument in the reliability function to determine system reliability (17). The survivor function is the fraction of the population that survives to time t , as well as the probability that a single item survives to time t . For continuous random variables, $S(t) = 1 - F(t)$, where $F(t) = P(T \leq t)$ is the cdf.

When the survivor function is differentiable, the associated pdf is

$$f(t) = -S'(t) \quad t \geq 0.$$

For any interval (a, b) , where $a < b$,

$$P(a \leq T \leq b) = \int_a^b f(t) dt.$$

Finite mixture models for k populations of items may be modeled using the pdf

$$f(t) = \sum_{i=1}^k p_i f_i(t) \quad t \geq 0$$

where $f_i(t)$ is the pdf for population i and p_i is the probability of selecting an item from population i , $i = 1, 2, \dots, k$. Mixture models are used in composition, a density-based variate generation technique.

The hazard function, also known as the rate function, failure rate, and force of mortality, can be defined by

$$h(t) = \frac{f(t)}{S(t)} \quad t \geq 0.$$

The hazard function is popular in reliability work because it has the intuitive interpretation as the amount of *risk* associated with an item that has survived to time t . The hazard function is a special form of the complete intensity function at time t for a point process (18). In other words, the hazard function is mathematically equivalent to the intensity function for a nonhomogeneous Poisson process, and the failure time corresponds to the first event time in the process. Competing risks models are easily formulated in terms of $h(t)$, as shown in the next section.

The cumulative hazard function can be defined by

$$H(t) = \int_0^t h(\tau) d\tau \quad t \geq 0.$$

If T is a random lifetime with cumulative hazard function H , then $H(T)$ is an exponential random variable with a mean of one. This result is the basis for the inverse-chf technique. The cumulative hazard function and the survivor function are related by $H(t) = -\log S(t)$.

2 RANDOM LIFETIME GENERATION

Techniques for generating a single, continuous lifetime from a known parametric probabilistic model can be partitioned into density-based and hazard-based algorithms. Density-based algorithms may be applied to any random variable, whereas hazard-based algorithms can only be applied to non-negative lifetimes. In this section, both types of algorithms are assumed to generate a nonnegative lifetime T .

The three classes of techniques for generating variates reviewed below are *inversion*, *linear combination methods*, and *majorizing methods*. For each class, there is a density-based method and a hazard-based method that are similar in nature. Examples of the use of all these techniques are given in Leemis and Schmeiser (19). Devroye (20) reviewed variate generation techniques requiring just one line of code.

Inversion

The density-based inverse cumulative distribution function technique, or *inverse-cdf* technique, is based on the probability integral transformation which states that $F(T) \sim U(0, 1)$, where F is the cdf for the random lifetime T . Thus,

$$T \leftarrow F^{-1}(U)$$

generates a lifetime T , where \leftarrow denotes assignment. If the cdf has a closed-form inverse, this method typically requires one line of computer code. If the inverse is not closed form, numerical methods must be used to invert the cdf.

Example 1 Consider a Weibull distribution with scale parameter λ and shape parameter κ . The cdf is

$$F(t) = 1 - e^{-(\lambda t)^\kappa} \quad t \geq 0,$$

which has the closed-form inverse

$$F^{-1}(u) = \frac{1}{\lambda}[-\log(1 - u)]^{1/\kappa} \quad 0 < u < 1.$$

Thus, an algorithm for generating a Weibull random variate is

$$T \leftarrow \frac{1}{\lambda}[-\log(1 - U)]^{1/\kappa}$$

where $U \sim U(0, 1)$. Most random number generators currently in use do not return exactly 0 or exactly 1. If U is generated so that 1 is excluded, $1 - U$ can be replaced with U for increased speed without concern over taking the logarithm of 0.

The inverse-cdf technique is based on $H(T)$ being exponentially distributed with a mean of one. So

$$T \leftarrow H^{-1}(-\log(1 - U))$$

generates a single random lifetime T . This algorithm is easiest to implement if H can be inverted in closed form.

Example 2 Consider an arrangement of three identical components with mutually independent and identically distributed Weibull lifetimes with parameters λ and κ as arranged in the block diagram in Figure 1. Find the mean time to system failure.

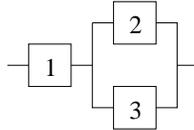


Figure 1: Block diagram for a three-component system.

This problem can be analyzed using both analytic and Monte Carlo simulation techniques. Let T_1 , T_2 , and T_3 be the lifetimes for the three statistically-identical components; let T be the system lifetime; and let $S_i(t) = e^{-(\lambda t)^\kappa}$ be the survivor function for component i for $i = 1, 2, 3$ and $t \geq 0$. The system survivor function is

$$\begin{aligned} S(t) &= S_1(t)[1 - (1 - S_2(t))(1 - S_3(t))] \\ &= e^{-(\lambda t)^\kappa} [1 - (1 - e^{-(\lambda t)^\kappa})(1 - e^{-(\lambda t)^\kappa})] \\ &= 2e^{-2(\lambda t)^\kappa} - e^{-3(\lambda t)^\kappa} \quad t \geq 0. \end{aligned}$$

Thus, the mean time to system failure is

$$E[T] = \int_0^\infty S(\tau) d\tau = \frac{\Gamma(1 + 1/\kappa)}{\lambda} (2^{1-1/\kappa} - 3^{-1/\kappa}).$$

To solve the problem exactly as stated, this analytic solution is ideal. For many applications, however, a Monte Carlo solution can provide additional insight into a problem. Furthermore, a less restricted problem (e.g., with more complicated failure distribution

or dependent component failure times) might not have a mathematically tractable analytic solution. A Monte Carlo estimate for the mean time to system failure requires each component lifetime to be generated, and the inverse-chf technique is used here. The cumulative hazard function for the Weibull distribution is

$$H(t) = (\lambda t)^\kappa \quad t \geq 0,$$

which has the closed-form inverse

$$H^{-1}(y) = \frac{1}{\lambda} y^{1/\kappa} \quad y \geq 0.$$

Thus, an algorithm for generating a Weibull random variate is

$$T \leftarrow \frac{1}{\lambda} [-\log(1 - U)]^{1/\kappa},$$

which is identical to the inverse-cdf technique. In general, the inverse-cdf and inverse-chf techniques are interchangeable in this fashion. An algorithm to estimate the mean time to system failure using N system lifetimes is given below. Indentation is used to indicate nesting.

```

S ← 0
for i from 1 to N
    generate U1, U2, U3 ~ U(0, 1)
    T1 ←  $\frac{1}{\lambda} [-\log(1 - U_1)]^{1/\kappa}$ 
    T2 ←  $\frac{1}{\lambda} [-\log(1 - U_2)]^{1/\kappa}$ 
    T3 ←  $\frac{1}{\lambda} [-\log(1 - U_3)]^{1/\kappa}$ 
    T ← min{T1, max{T2, T3}}
    S ← S + T
A ← S/N

```

The variable S contains a cumulative sum of the system lifetimes, and A contains the average of the system lifetimes generated. The estimate for the average time to system failure A converges to the analytic result as the number of replications $N \rightarrow \infty$. This algorithm can be written more efficiently because the components have identical distributions. To save on the number of logarithms and exponentiations, properties such as $T_2 \geq T_3$ when $U_2 \geq U_3$ can be exploited so that only one Weibull variate needs to be generated for each system lifetime, based on the order of U_1, U_2, U_3 , as shown in the next example. The algorithm given above is inferior to the analytic method presented earlier in the example because it produces a *point estimator* A for the system reliability. It is appropriate to place a *confidence interval* around the point estimator in order to assess its precision (21).

A final example is given to illustrate an alternative way of generating the system lifetime of a coherent system (17) of components.

Example 3 Consider the same system as Example 2, which used three random numbers to generate a single system lifetime T . Thus, the algorithm was not synchronized. A technique for achieving synchronization is illustrated in this example. The first step is to determine $\pi_i = P[T_i = T]$, for $i = 1, 2, 3$, which is the probability that component i is the component that “causes” system failure. Second, the conditional lifetime distribution of all components, given that they are the cause of failure, should be determined, and a lifetime variate generated from the appropriate distribution. For the three-component example, we obtain

$$\pi_2 = P(T_3 < T_2 < T_1)$$

and, by symmetry

$$\pi_3 = P(T_2 < T_3 < T_1) = \pi_2,$$

and

$$\pi_1 = 1 - \pi_2 - \pi_3$$

since $\pi_1 + \pi_2 + \pi_3 = 1$. The synchronized algorithm for generating a system lifetime T from a single $U(0, 1)$ is given below. The U that is used in the last step of the algorithm has been rescaled so that it is conditionally $U(0, 1)$.

Setup

determine π_1, π_2, π_3

find the conditional lifetime distributions for all components

Algorithm

generate $U \sim U(0, 1)$

if $0 < U < \pi_1$

$J \leftarrow 1$

$U \leftarrow U/\pi_1$

if $\pi_1 < U < \pi_1 + \pi_2$

$J \leftarrow 2$

$U \leftarrow (U - \pi_1)/\pi_2$

if $\pi_1 + \pi_2 < U < 1$

$J \leftarrow 3$

$U \leftarrow (U - \pi_1 - \pi_2)/\pi_3$

generate T from conditional lifetime distribution J using U

Inversion techniques use a monotonic transformation to transform a random number U to a random variate T . These techniques exhibit the following important properties.

- They are synchronized (i.e., one random number produces one lifetime).
- They are monotone (i.e., larger random numbers produce larger lifetimes).
- They accommodate truncated distributions.
- They can be modified to generate order statistics (useful for generating the lifetime of a k -out-of- n system, as shown in the next section).

Linear Combination Methods

Linear combination techniques are the density-based composition method and the hazard-based competing risks method. The composition method is viable when the pdf can be written as the convex combination of k density functions

$$f(t) = \sum_{j=1}^k p_j f_j(t) \quad t \geq 0,$$

where $\sum_{j=1}^k p_j = 1$. The algorithm is

choose pdf j with probability p_j , $j = 1, 2, \dots, k$
generate T from pdf j

The first step is typically executed using a discrete inversion algorithm.

The second linear combination technique is called *competing risks*, which can be applied when the hazard function can be written as the sum of hazard functions, each corresponding to a “cause” of failure

$$h(t) = \sum_{j=1}^k h_j(t) \quad t \geq 0,$$

where $h_j(t)$ is the hazard function associated with cause j of failure acting in a population. When the lifetimes T_1, T_2, \dots, T_k are mutually independent, the minimum of the lifetimes from each of these risks corresponds to the system lifetime. Competing risks is most commonly used to analyze series systems; it can also be used in actuarial applications. The competing risks model is also used for modeling competing failure modes for components that have multiple failure modes. The algorithm is

generate T_j from $h_j(t)$, $j = 1, 2, \dots, k$
 $T \leftarrow \min\{T_1, T_2, \dots, T_k\}$

Majorizing Methods

The third class of techniques for generating random lifetimes is the majorizing techniques: the density-based acceptance/rejection technique and a modified version of the hazard-based thinning technique. In order to use acceptance/rejection, there must be a majorizing function $f^*(t)$ that satisfies $f^*(t) \geq f(t)$ for all $t \geq 0$. The pdf corresponding to $f^*(t)$ (which requires normalization) is

$$g(t) = \frac{f^*(t)}{\int_0^\infty f^*(\tau) d\tau}.$$

Assuming that a random variate T is easily generated from the probability distribution with pdf $g(t)$, the acceptance/rejection algorithm for generating T is

repeat
 generate T from $g(t)$
 generate $S \sim U(0, f^*(T))$
until $S \leq f(T)$

Generating T may be done by inversion or any other method. The name *acceptance/rejection* comes from the loop condition; the random variate T is accepted for generation if $S \leq f(T)$ and rejected if $S > f(T)$.

Thinning was originally used by Lewis and Shedler (22) for generating the event times in a nonhomogeneous Poisson process. Thinning can be adapted to produce a single lifetime by ignoring all except the first event time generated. A majorizing hazard function $h^*(t)$ must be found that satisfies $h^*(t) \geq h(t)$ for all $t \geq 0$. The thinning algorithm for generating T is

```

 $T \leftarrow 0$ 
repeat
  generate  $Y$  from  $h^*(t)$  given  $Y > T$ 
   $T \leftarrow T + Y$ 
  generate  $S \sim U(0, h^*(T))$ 
until  $S \leq h(T)$ 

```

Generating Y in the loop can be done by inversion or any other method. The term *thinning* comes from the fact that T can make several steps, each of length Y , that are thinned out before the loop condition is satisfied.

Special Properties

The fourth class of techniques for generating random lifetimes is called *special properties*. These techniques are neither density- nor hazard-based because they depend on relationships between random variables. Examples of special properties include generating an Erlang random variable as the sum of mutually independent and identically distributed exponential random variables, and generating a binomial random variable as the sum of mutually independent and identically distributed Bernoulli random variables. Examples of special properties associated with random variables are given in the encyclopedic work of Johnson, Kotz, and Balakrishnan (23).

The four techniques described in this section are often combined in order to generate a variate from a particular distribution. Devroye (5) and Dagpunar (6) review variate generation techniques for some of the more intractable distributions (e.g., normal and gamma) that are not considered here. Most computer languages have access to subprograms capable of generating variates from a wide range of distributions.

The generation of mutually independent univariate random variates provides the basis for Monte Carlo simulation analysis of reliability models. There are a number of directions that a section of this nature could take at this point. I have opted for surveying the following: generating order statistics, generating lifetimes for models with covariates, and generating nonhomogeneous Poisson processes. Other important topics include generating random vectors [see Rao (16) and Grimlund (24)], civil engineering applications [see Harr (12)], mechanical engineering applications [see Rao (16)], fault tree analysis [see Henley and Kumamoto (13)], or discrete-event simulation [see Law (25)].

3 ORDER STATISTIC GENERATION

In many reliability applications, the efficient generation of order statistics can be useful for generating a random system lifetime. Order statistics play a central role in the analysis of simple

arrangements of systems consisting of n statistically identical components. Let T_1, T_2, \dots, T_n be the n mutually independent failure times of components in a system, and let $T_{(1)}, T_{(2)}, \dots, T_{(n)}$ be the ordered failure times. If T denotes the system failure time, then $T = T_{(1)}$ for a series system, $T = T_{(n)}$ for a parallel system, and $T = T_{(n-k+1)}$ for a k -out-of- n system. The most straightforward approach to generating the system lifetime for these models is to generate the lifetimes of each component, sort the lifetimes, and then choose the appropriate order statistic. This approach is adequate when n is small and the lifetimes are simple to generate. When one or both of these conditions do not hold, the following results from Schucany (26), Ramberg and Tadikamalla (27), and Schmeiser (28, 29) can be used to generate order statistics more efficiently. The algorithms presented in this section are effective ways of decreasing the central processing unit (CPU) time to generate a system lifetime since only one inversion of F is necessary and no sorting is required.

The random variables $\min\{U_1, U_2, \dots, U_n\}$ and $1 - (1 - U)^{1/n}$ have the same probability distribution, where $U_i, i = 1, 2, \dots, n$ and U are independent random numbers. If the function $F^{-1}(u)$ can be evaluated in closed form or numerically, an algorithm to generate the system lifetime of a series system of identical components is

$$T \leftarrow F^{-1}(1 - (1 - U)^{1/n}).$$

Since $\max\{U_1, U_2, \dots, U_n\}$ and $U^{1/n}$ have the same probability distribution, the system lifetime of a parallel system of statistically identical components can be generated by

$$T \leftarrow F^{-1}(U^{1/n}).$$

Example 4 A system of n statistically identical components is arranged in parallel. If each component has an independent Weibull lifetime with scale parameter λ and shape parameter κ , devise a method to generate a system lifetime variate from a single random number U .

The inverse of the Weibull cdf is

$$F^{-1}(u) = \frac{1}{\lambda}[-\log(1 - u)]^{1/\kappa} \quad 0 < u < 1.$$

A system lifetime T , which corresponds to the order statistic $T_{(n)}$, is generated by

$$T \leftarrow \frac{1}{\lambda}[-\log(1 - U^{1/n})]^{1/\kappa}.$$

This is clearly faster than generating n Weibull variates and returning the largest generated.

Generating minimums and maximums of mutually independent random lifetimes are special cases of generating the k th order statistics from a sample of size n . The algorithm below generates order statistics efficiently for a k -out-of- n system when a beta variate generator is available.

generate $X \sim \beta(n - k + 1, k)$

$$T \leftarrow F^{-1}(X)$$

The variate T which is generated corresponds to $T_{(n-k+1)}$. Efficient algorithms for generating beta variates are given by Law (25).

4 ACCELERATED LIFE AND PROPORTIONAL HAZARDS MODELS

The effect of covariates (explanatory variables) on survival often complicates the analysis of a set of lifetime data. In a medical setting, these covariates are usually patient characteristics such as age, gender, or blood pressure. In reliability, covariates (such as the turning speed of a machine tool or the stress applied to a component) affect the lifetime of an item. Two common models to incorporate the effect of the covariates on lifetimes are the *accelerated life* and *Cox proportional hazards* models. This section describes algorithms for the generation of lifetimes that are described by one of these models.

The $q \times 1$ vector \mathbf{z} contains covariates associated with a particular item or individual. The covariates are linked to the lifetime by the function $\psi(\mathbf{z})$, which satisfies $\psi(\mathbf{0}) = 1$ and $\psi(\mathbf{z}) \geq 0$ for all \mathbf{z} . A popular choice is $\psi(\mathbf{z}) = e^{\boldsymbol{\beta}'\mathbf{z}}$, where $\boldsymbol{\beta}$ is a $q \times 1$ vector of regression coefficients.

The cumulative hazard function for T in the *accelerated life* model is (18)

$$H(t) = H_0(t\psi(\mathbf{z})),$$

where H_0 is a baseline cumulative hazard function. When $\mathbf{z} = \mathbf{0}$, then $H_0 \equiv H$. In this model, the covariates accelerate [$\psi(\mathbf{z}) > 1$] or decelerate [$\psi(\mathbf{z}) < 1$] the rate at which the item moves through time. The cumulative hazard function for T in the *proportional hazards* model is

$$H(t) = \psi(\mathbf{z})H_0(t).$$

In this model, the covariates increase [$\psi(\mathbf{z}) > 1$] or decrease [$\psi(\mathbf{z}) < 1$] the failure rate of the item by the factor $\psi(\mathbf{z})$ for all values of t .

All the algorithms for variate generation for these models are based on the fact that $H(T)$ is exponentially distributed with a mean of one. Therefore, equating the cumulative hazard function to $-\log(1 - U)$ and solving for t yields the appropriate generation technique.

In the accelerated life model, since time is being expanded or contracted by a factor $\psi(\mathbf{z})$, variates are generated by

$$T \leftarrow \frac{H_0^{-1}(-\log(1 - U))}{\psi(\mathbf{z})}.$$

In the proportional hazards model, equating $-\log(1 - U)$ to $H(t)$ yields the variate generation formula

$$T \leftarrow H_0^{-1}\left(\frac{-\log(1 - U)}{\psi(\mathbf{z})}\right).$$

In addition to generating individual lifetimes, these variate generation techniques can be applied to point processes. A renewal process, for example, with time between events having a cumulative hazard function $H(t)$ can be simulated by using the appropriate generation formula for the two cases shown above. These variate generation formulas must be modified, however, to generate variates from a nonhomogeneous Poisson process (NHPP).

In an NHPP, the hazard function $h(t)$ is analogous to the intensity function, which governs the rate at which events occur. To determine the appropriate method for generating variates from an NHPP, assume that the last event in a point process has occurred at time a . The cumulative hazard function for the time of the next event, conditioned on survival to time a , is

$$H_{T|T>a}(t) = H(t) - H(a) \quad t \geq a.$$

In the accelerated life model, where $H(t) = H_0(t\psi(\mathbf{z}))$, the time of the next event is generated by

$$T \leftarrow \frac{H_0^{-1}(H_0(a\psi(\mathbf{z})) - \log(1 - U))}{\psi(\mathbf{z})}.$$

Equating the conditional cumulative hazard function to $-\log(1 - U)$, the time of the next event in the proportional hazards case is generated by

$$T \leftarrow H_0^{-1}\left(H_0(a) - \frac{\log(1 - U)}{\psi(\mathbf{z})}\right).$$

An example of the application of these algorithms to a particular parametric distribution is given by Leemis (30), who later extended to the case where the covariates are time dependent are given in Leemis, Shih and Reynertson (31) and Shih and Leemis (32). Table 1 summarizes the variate generation algorithms for the accelerated life and proportional hazards models (the last event occurred at time a). The $1 - U$ has been replaced with U in this table to save a subtraction, although the sense of the monotonicity is reversed. The renewal and NHPP algorithms are equivalent when $a = 0$ (since a renewal process is equivalent to an NHPP restarted at zero after each event), the accelerated life and proportional hazards models are equivalent when $\psi(\mathbf{z}) = 1$, and all four cases are equivalent when $H_0(t) = \lambda t$ (the exponential case) because of its memoryless property.

Table 1. Lifetime generation in regression models

	Renewal	NHPP
Accelerated life	$T \leftarrow a + \frac{H_0^{-1}(-\log(U))}{\psi(\mathbf{z})}$	$T \leftarrow \frac{H_0^{-1}(H_0(a\psi(\mathbf{z})) - \log(U))}{\psi(\mathbf{z})}$
Proportional hazards	$T \leftarrow a + H_0^{-1}\left(\frac{-\log(U)}{\psi(\mathbf{z})}\right)$	$T \leftarrow H_0^{-1}\left(H_0(a) - \frac{\log(U)}{\psi(\mathbf{z})}\right)$

5 GENERATING A NONHOMOGENEOUS POISSON PROCESS

This section describes two techniques for generating event times for NHPPs. Homogeneous Poisson processes and renewal processes are not considered since they are a straightforward generalization of the inversion algorithms. An NHPP is often suggested as an appropriate model for the failure times of repairable systems whose rate of occurrence of failures varies over time, as outlined in Ascher and Feingold (3), Gertsbakh (33), Rigdon and Basu (34), and Ross (35). The repair time must be negligible in order to use an NHPP to approximate the probabilistic mechanism governing the sequence of failures. The two techniques considered here are *inversion*, which relies on a time-scale transformation given by Çinlar (36), and *thinning*, developed by Lewis and Shedler (22).

An NHPP is a generalization of an ordinary homogeneous Poisson process with events occurring randomly over time at the rate of λ . Events occur over time at a rate defined by the *intensity function*, $\lambda(t)$. The *cumulative intensity function* is defined by

$$\Lambda(t) = \int_0^t \lambda(\tau) d\tau \quad t > 0,$$

and is interpreted as the mean number of events by time t .

Inversion

For cases in which $\Lambda(t)$ can be inverted in closed form, or when it can be inverted numerically, Çinlar (36) showed that if E_1, E_2, \dots are the event times in a homogeneous Poisson process with rate 1 (often called a *unit* Poisson process), then $\Lambda^{-1}(E_1), \Lambda^{-1}(E_2), \dots$ are the event times for an NHPP with cumulative intensity function $\Lambda(t)$. This is a generalization of the result that formed the basis for the inverse-chf algorithm. An algorithm for generating the event times T_1, T_2, \dots , for an NHPP with cumulative intensity function $\Lambda(t)$ is given below.

```
 $T_0 \leftarrow 0$ 
 $E_0 \leftarrow 0$ 
 $i \leftarrow 0$ 
repeat
   $i \leftarrow i + 1$ 
  generate  $U \sim U(0, 1)$ 
   $E_i \leftarrow E_{i-1} - \log(1 - U)$ 
   $T_i \leftarrow \Lambda^{-1}(E_i)$ 
until  $T_i \geq S$ 
```

The algorithm returns the event times T_1, T_2, \dots, T_{i-1} , where S is a prescribed termination time of the point process. The algorithm is valid because $-\log(1-U)$ is the appropriate way (via inversion) of generating an exponential variate with a mean of one. As before, replacing $1-U$ with U reduces the CPU time.

Example 5 The cumulative intensity function given by

$$\Lambda(t) = (\lambda t)^\kappa \quad t > 0,$$

is often known as the *power-law process* (34). If $\kappa > 1$, the population of items is deteriorating; if $\kappa < 1$, the population of items is improving; and if $\kappa = 1$, the NHPP simplifies to a homogeneous Poisson process. Since the inverse cumulative intensity function is

$$\Lambda^{-1}(y) = \frac{1}{\lambda} y^{1/\kappa} \quad y > 0,$$

the last statement in the loop becomes $T_i \leftarrow \frac{1}{\lambda} E_i^{1/\kappa}$.

The techniques for estimating the cumulative intensity function for an NHPP from one or more realizations is too broad a topic to be reviewed here. Examples of parametric and nonparametric techniques for estimation and generating realizations for simulation models are given by Lee, Wilson, and Crawford (37), Rigdon and Basu (38), Chen and Schmeiser (39), Nicol and Leemis (40), Chen and Schmeiser (41), and Liu, Kuhl, Liu, and Wilson (42). Also, the technique given by Leemis (43) is illustrated in the following example.

Example 6 This example considers nonparametric estimation of the cumulative intensity function of an NHPP from one or more realizations and the associated algorithm

for generating random variates. This method does not require the modeler to specify any parameters or weighting functions.

The cumulative intensity function is to be estimated from k realizations of the NHPP on $(0, S]$, where S is a known constant. Let $n_i (i = 1, 2, \dots, k)$ be the number of observations in the i^{th} realization, $n = \sum_{i=1}^k n_i$, and let $t_{(1)}, t_{(2)}, \dots, t_{(n)}$ be the order statistics of the superposition of the k realizations, $t_{(0)} = 0$ and $t_{(n+1)} = S$. Setting $\hat{\Lambda}(S) = n/k$ yields a process where the expected number of events by time S is the average number of events in k realizations, since $\Lambda(S)$ is the expected number of events by time S . The piecewise linear estimator of the cumulative intensity function between the time values in the superposition is

$$\hat{\Lambda}(t) = \frac{in}{(n+1)k} + \left[\frac{n(t-t_{(i)})}{(n+1)k(t_{(i+1)}-t_{(i)})} \right] \quad t_{(i)} < t \leq t_{(i+1)}; \quad i = 0, 1, 2, \dots, n.$$

This estimator passes through the points $(t_{(i)}, in/[(n+1)k])$, for $i = 1, 2, \dots, n+1$.

One rationale for using a linear function between the data values is that inversion can be used for generating realizations without having tied events. If the usual step-function estimate of $\Lambda(t)$ is used, only the $t_{(i)}$ values could be generated.

Using inversion, the event times from a unit Poisson process, E_1, E_2, \dots , can be transformed to the event times of an NHPP via $T_i = \Lambda^{-1}(E_i)$. For the NHPP estimate considered here, the events at times T_1, T_2, \dots can be generated for Monte Carlo simulation by the algorithm below, given n, k, S and the superpositioned values.

```

i ← 1
generate  $U_i \sim U(0, 1)$ 
 $E_i \leftarrow -\log(1 - U_i)$ 
while  $E_i < n/k$ 
   $m \leftarrow \left\lfloor \frac{(n+1)kE_i}{n} \right\rfloor$ 
   $T_i \leftarrow t_{(m)} + [t_{(m+1)} - t_{(m)}] \left( \frac{(n+1)kE_i}{n} - m \right)$ 
   $i \leftarrow i + 1$ 
generate  $U_i \sim U(0, 1)$ 
 $E_i \leftarrow E_{i-1} - \log(1 - U_i)$ 

```

Thus, it is a straightforward procedure to obtain a realization of $i - 1$ events on $(0, S]$ from the superpositioned process and $U(0, 1)$ values U_1, U_2, \dots, U_i . Inversion has been used to generate this NHPP, so certain variance reduction techniques, such as antithetic variates or common random numbers, may be applied to the simulation output. Replacing $1 - U_i$ with U_i in generating the exponential variates will save CPU time, although the direction of the monotonicity is reversed. Tied values in the superposition do not pose any problem to this algorithm, although there may be tied values in the realization. As n increases, the amount of memory required increases, but the amount of CPU time required to generate a realization depends only on the ratio n/k , the average number of events per realization.

If the inverse cumulative intensity function is not available, but a majorizing intensity function can be found, then thinning can be used to generate variates.

Thinning

In describing the basic techniques for variate generation, thinning was adapted to generate a single lifetime. Thinning was originally devised to generate the event times for an NHPP (22). Assume that a majorizing intensity function $\lambda^*(t)$ exists such that $\lambda^*(t) \geq \lambda(t)$ for all $t \geq 0$. The thinning algorithm is shown below.

```

 $T_0 \leftarrow 0$ 
 $i \leftarrow 0$ 
repeat
   $i \leftarrow i + 1$ 
   $Y \leftarrow T_{i-1}$ 
  repeat
    generate  $U_1, U_2 \sim U(0, 1)$ 
     $Y \leftarrow Y - \log(1 - U_1)$ 
  until  $U_2 \leq \lambda(Y)/\lambda^*(Y)$ 
   $T_i \leftarrow Y$ 
until  $T_i \geq S$ 

```

If the inside loop condition is not met, then this particular Y value is “thinned” out of the point process and not included as a failure time in the realization. Choosing a majorizing function that is close to the intensity function $\lambda^*(t)$ results in fewer passes through the inside loop, and hence reduces CPU time. Guo and Love (44) have adapted this algorithm for the generation of variates when covariates are included in the model.

The discussion in this article has focused primarily on variate generation for various Monte Carlo simulation applications in reliability. The estimation of unknown parameters from data is an equally important topic that has not been considered here. Banks, Carson, Nelson, and Nicol (45), Bratley, Fox, and Schrage (46), and Seila, Vlatko, and Tadikamalla (47) have sections on a facet of discrete-event simulation known as “input modeling,” where the estimation of parameters is considered. Finally, Fishman (48) presents a comprehensive work on Monte Carlo simulation without taking the reliability modeling perspective that has been taken here.

List of symbols

T	a continuous nonnegative random variable
\sim	“is distributed as”
$f(t), F(t), S(t)$	the pdf, cdf, and survivor function of T
$h(t), H(t)$	the hazard function, cumulative hazard function of T
U	a random number [i.e., a $U(0, 1)$ random variable]
λ	scale parameter for an exponential distribution
λ, κ	scale and shape parameters for a Weibull distribution
\log	natural logarithm
Γ	the gamma function
$T_{(i)}$	order statistic i
\mathbf{z}	a $q \times 1$ vector of covariates
$\boldsymbol{\beta}$	a $q \times 1$ vector of regression coefficients
$\psi(\mathbf{z})$	link function
$\lambda(t)$	intensity function
$\Lambda(t)$	cumulative intensity function
E_1, E_2, \dots	homogeneous Poisson process event times
T_1, T_2, \dots	event times for an NHPP

References

1. D. Goldsman, R.E. Nance, J.R. Wilson, A Brief History of Simulation Revisited, *Proceedings of the 2010 Winter Simulation Conference*, B. Johansson, S. Jain, J. Montoya-Torres, J. Hagan, and E. Yücesan, editors, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 567–574, 2010.
2. G.B. Swartz, The Mean Residual Life Function, *IEEE Transactions on Reliability*, R-22, 108–109, 1973.
3. H. Ascher and H. Feingold, *Repairable Systems Reliability*, New York: Marcel Dekker, 1984.
4. R.E. Barlow, Geometry of the Total Time on Test Transform, *Naval Research Logistics Quarterly*, 26, 393–402, 1979.
5. L. Devroye, *Non-Uniform Random Variate Generation*, New York: Springer-Verlag, 1986.
6. J. Dagpunar, *Principles of Random Variate Generation*, Oxford: Oxford Science Publications, 1988.
7. B. Schmeiser, Simulation Experiments. From *Stochastic Models* (D.P. Heyman and M.J. Sobel, editors), North-Holland, 1990.
8. S.K. Park and K.W. Miller, Random Number Generators: Good Ones Are Hard to Find, *Communications of the ACM*, 31, 10, 1192–1201, 1988.
9. P. L’Ecuyer, History of Uniform Random Number Generation, *Proceedings of the 2017 Winter Simulation Conference*, W. K. V. Chan, A. D’Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, editors, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 202–230, 2017.
10. J.W. Foster, D.T. Phillips, and T.R. Rogers, *Reliability, Availability and Maintainability*, Beaverton, Oregon: M/A Press, 1981.
11. H. Goldberg, *Extending the Limits of Reliability Theory*, New York: John Wiley & Sons, 1981.
12. M.E. Harr, *Reliability-Based Design in Civil Engineering*, New York: McGraw-Hill, 1987.
13. E.J. Henley and H. Kumamoto, *Reliability Engineering and Risk Assessment*, Englewood Cliffs, New Jersey: Prentice-Hall, 1981.
14. L.M. Leemis, *Reliability: Probabilistic Models and Statistical Methods*, Third Edition, La Vergne, Tennessee: Lightning Source, 2025.
15. N.R. Mann, R.E. Schafer, and N.D. Singpurwalla, *Methods for Statistical Analysis of Reliability and Life Data*, New York: John Wiley & Sons, 1974.
16. S.S. Rao, *Reliability Based Design*, New York: McGraw-Hill, 1992.
17. R.E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing*, Silver Springs, Maryland: To Begin With, 1981.

18. D.R. Cox and D. Oakes, *Analysis of Survival Data*, London: Chapman and Hall, 1984.
19. L.M. Leemis and B.W. Schmeiser, Random Variate Generation for Monte Carlo Experiments, *IEEE Transactions on Reliability*, R-34, 1, 81–85, 1985.
20. L. Devroye, Random Variate Generation in One Line of Code, *Proceedings of the 1996 Winter Simulation Conference*, J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, editors, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 265–272, 1996.
21. W.Q. Meeker, G.J. Hahn, L.A. Escobar, *Statistical Intervals: A Guide for Practitioners and Researchers*, Second Edition, John Wiley & Sons, 2017.
22. P.A.W. Lewis and G.S. Shedler, Simulation of Nonhomogeneous Poisson Processes by Thinning, *Naval Research Logistics Quarterly*, 26, 3, 403–413, 1979.
23. N.L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions, Volume 1*, New York: John Wiley & Sons, 1994.
24. R.A. Grimlund, Generating Statistically Dependent Pairs of Random Variables: A Marginal Distribution Approach, *European Journal of Operations Research*, 57, 1, 39–53, 1992.
25. A.M. Law, *Simulation Modeling and Analysis*, Fifth Edition, New York: McGraw–Hill, 2015.
26. W.R. Schucany, Order Statistics in Simulation. *Journal of Statistical Computation and Simulation*, 1, 281–286, 1972.
27. J.S. Ramberg and P.R. Tadikamalla, On the Generation of Subsets of Order Statistics, *Journal of Statistical Computation and Simulation*, 6, 239–241, 1978.
28. B.W. Schmeiser, Generation of the Maximum (Minimum) Value in Digital Computer Simulation, *Journal of Statistical Computation and Simulation*, 8, 103–115, 1978.
29. B.W. Schmeiser, The Generation of Order Statistics in Digital Computer Simulation: A Survey, *Proceedings 1978 Winter Simulation Conference*, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 137–140, 1978.
30. L.M. Leemis, Variate Generation for the Accelerated Life and Proportional Hazards Models, *Operations Research*, 35, 6, 892–894, 1987.
31. L.M. Leemis, L.H. Shih, and K. Reynertson, Variate Generation for the Accelerated Life and Proportional Hazards Models with Time Dependent Covariates, *Statistics and Probability Letters*, 10, 1, 335–339, 1990.
32. L.H. Shih and L.M. Leemis, Variate Generation for a Nonhomogeneous Poisson Process with Time Dependent Covariates, *Journal of Computation and Simulation*, 44, 165–186, 1993.

33. I.B. Gertsbakh, *Reliability Theory With Applications to Preventive Maintenance*, New York: Springer–Verlag, 2000.
34. S.E. Rigdon and A.P. Basu, *Statistical Methods for the Reliability of Repairable Systems*, New York: McGraw–Hill, 2000.
35. S.M. Ross, *Simulation*, Fifth Edition, New York: Academic Press, 2013.
36. E. Çinlar, *Introduction to Stochastic Processes*, Englewood Cliffs, New Jersey: Prentice–Hall, 1975.
37. S. Lee, J.R. Wilson, and M.M. Crawford, Modeling and Simulation of a Nonhomogeneous Poisson Process with Cyclic Features. *Communications in Statistics–Simulation and Computation*, 20, 2&3, 777–809, 1991.
38. S.E. Rigdon and A.P. Basu, The Power Law Process: A Model for the Reliability of Repairable Systems, *Journal of Quality Technology*, 21, 4, 251–260, 1989.
39. H. Chen, B.W. Schmeiser, I-SMOOTH: Iteratively Smoothing Mean-Constrained and Non-negative Piecewise-Constant Functions, *INFORMS Journal on Computing*, 25, 432–445, 2013.
40. D.M. Nicol, L.M. Leemis A Continuous Piecewise-Linear NHPP Intensity Function Estimator, *Proceedings of the 2014 Winter Simulation Conference*, A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, editors, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 498–509, 2014.
41. Chen, H., Schmeiser, B.W. (2017), MNO–PQRS: Max Nonnegativity Ordering—Piecewise-Quadratic Rate Smoothing, *ACM Transactions on Modeling and Computer Simulation*, 27, 3, 1–19, 2017.
42. R. Liu, M.E. Kuhl, Y. Liu, J.R. Wilson, Modeling and Simulation of Nonstationary Non-Poisson Arrival Processes, *INFORMS Journal on Computing*, 31, 2, 347–366, 2018.
43. L.M. Leemis, Nonparametric Estimation of the Intensity Function for a Nonhomogeneous Poisson Process, *Management Science*, 37, 7, 886–900, 1991.
44. R. Guo and C.E. Love, Simulating Nonhomogeneous Poisson Processes with Proportional Intensities, *Naval Research Logistics*, 41, 4, 507–522, 1994.
45. J. Banks, J.S. Carson, B.L. Nelson, and D.M. Nicol, *Discrete-Event System Simulation*, Fifth Edition, Boston: Pearson, 2010.
46. P. Bratley, B.L. Fox, and L.E. Schrage, *A Guide to Simulation*, Second Edition, New York: Springer–Verlag, 1987.
47. A.F. Seila, V.C. Vlatko, P. Tadikamalla, *Applied Simulation Modeling*, Belmont, CA: Thompson/Brooks/Cole, 2003.
48. G. Fishman, *Monte Carlo Simulation: Concepts, Algorithms, and Applications*, New York: Springer–Verlag, 1996.