S.G. Henderson and B.L. Nelson (Eds.), *Handbook in OR & MS, Vol. 13*
Copyright © 2006 Elsevier B.V. All rights reserved
DOI: 10.1016/S0927-0507(06)13006-6

Chapter 6

Arrival Processes, Random Lifetimes and Random Objects

Lawrence M. Leemis
Department of Mathematics, The College of William & Mary, USA

Abstract

This chapter considers (1) the modeling, estimation, and generation of arrival processes, (2) the generation of random lifetimes, and (3) the generation of random objects. Accurate modeling of arrival processes is critical for the development of valid discrete-event simulation models. Although the generation of random lifetimes is typically applied to reliability and survival analysis in a simulation setting, their use is widespread in other disciplines as well. The generation of "random objects" will be applied here to generating combinatorial objects (e.g., combinations and permutations), matrices, and polynomials. The wider literature on generating random objects is much more diverse, including generating random card shuffles, generating random colors, generating random geometric objects, and generating random spawning trees.

1 Arrival processes

Most discrete-event simulation models have stochastic elements that mimic the probabilistic nature of the system under consideration. The focus in this section is on the modeling and generation of arrival processes. Stochastic models for arrival processes could be used to model the arrival times of customers to a queue, the arrival times of jobs to a job shop, the arrival times of demands in an inventory system, the arrival times of bills to Congress, or the arrival times of failures to a repairable machine. These stochastic models generalize to model events that occur over time or space. A close match between the arrival process model and the true underlying probabilistic mechanism associated with arrivals to the system of interest is required for successful simulation input modeling. The general question considered here is how to model an arrival process in a discrete-event simulation. It is typically the case that a data set of arrival times has been collected on the system of interest. We begin by introducing probabilistic models for an arrival process, which are special cases of what are known as "point processes", where "events" occur at points in time.

A special case of a point process is a "counting process", where event occurrences increment a counter.

1.1 Counting processes

A continuous-time, discrete-state stochastic process is often characterized by the *counting function* $\{N(t), t \geqslant 0\}$ which represents the total number of "events" that occur by time t (Ross, 2003). In a simulation setting, these events may be arrivals to a store, the births of babies, or machine failures. A counting process satisfies the following properties:

(1) $N(t) \geqslant 0$,
(2) $N(t)$ is integer-valued,
(3) if $s < t$, then $N(s) \leqslant N(t)$, and
(4) for $s < t$, $N(t) - N(s)$ is the number of events in $(s, t]$.

Two important properties associated with some counting processes are *independent increments* and *stationarity*. A counting process has independent increments if the number of events that occur in mutually exclusive time intervals are independent. A counting process is stationary if the distribution of the number of events that occur in any time interval depends only on the length of the interval. Thus, the stationarity property should only apply to counting processes with a constant rate of occurrence of events.

Counting processes can be used in modeling events as diverse as earthquake occurrences (Schoenberg, 2003), customer arrival times to an electronics store (White, 1999), and failure times of a repairable system (Nelson, 2003).

We establish some additional notation at this point which will be used in some results and process generation algorithms that follow. Let $X_1, X_2, \ldots$ represent the times between events in a counting process. Let $T_n = X_1 + X_2 + \cdots + X_n$ be the time of the nth event. With these basic definitions in place, we now define the Poisson process, which is the most fundamental of the counting processes.

1.2 Poisson processes

A Poisson process is a special type of counting process that is a fundamental base case for defining many other types of counting processes.

Definition (Ross, 2003). The counting process $\{N(t), t \geqslant 0\}$ is said to be a *Poisson process* with rate λ, $\lambda > 0$, if

(1) $N(0) = 0$,
(2) the process has independent increments, and
(3) the number of events in any interval of length t is Poisson distributed with mean λt.

The single parameter λ controls the rate at which events occur over time. Since λ is a constant, a Poisson process is often referred to as a *homogeneous* Poisson process. The third condition is equivalent to

$$\mathrm{P}\big(N(t+s) - N(s) = n\big) = \frac{(\lambda t)^n \exp(-\lambda t)}{n!}, \quad n = 0, 1, 2, \ldots,$$

and the stationarity property follows from it.

Although there are many results that follow from the definition of a Poisson process, three are detailed in this paragraph that have applications in discrete-event simulation. Proofs are given in any introductory stochastic processes textbook. First, given that n events occur in a given time interval $(s, t]$, the event times have the same distribution as the order statistics associated with n independent observations drawn from a uniform distribution on $(s, t]$. Second, the times between events in a Poisson process are independent and identically distributed exponential random variables with probability density function $f(x) = \lambda \exp(-\lambda x)$, for $x > 0$. Since the mode of the exponential distribution is 0, a realization of a Poisson process typically exhibits significant clustering of events. Since the sum of n independent and identically distributed exponential(λ) random variables is Erlang(λ, n), T_n has a cumulative distribution function that can be expressed as a summation

$$F_{T_n}(t) = 1 - \sum_{k=0}^{n-1} \frac{(\lambda t)^k}{k!} \mathrm{e}^{-\lambda t}, \quad t > 0.$$

Third, analogous to the central limit theorem, which shows that the sum of arbitrarily distributed random variables is asymptotically normal, the superposition of renewal processes converges asymptotically to a Poisson process (Whitt, 2002, p. 318).

The mathematical tractability associated with the Poisson process makes it a popular model. It is the base case for queueing theory (e.g., the $M/M/1$ queue as defined in Gross and Harris, 1985) and reliability theory (e.g., the models for repairable systems described in Meeker and Escobar, 1998). Its rather restrictive assumptions, however, limit its applicability. For this reason, we consider variants of the Poisson process which can be useful for modeling more complex arrival processes.

1.3 Variants of Poisson processes

There are many variants of the Poisson process which would be appropriate for modeling arrivals in a discrete-event simulation model. We outline several of these models in this section. These variants are typically formulated by generalizing an assumption or a property of the Poisson process. Details can be found, for example, in Resnick (1992) or Nelson (2002).

Renewal processes

A renewal process is a generalization of a Poisson process. Recall that in a Poisson process, the interarrival times $X_1, X_2, \ldots,$ are i.i.d. exponential(λ) random variables. In a renewal process, the interarrival times are independent and identically distributed random variables from any distribution with positive support. One useful classification of renewal processes (Cox and Isham, 1980) concerns the coefficient of variation σ/μ of the distribution of the times between failures. This classification divides renewal processes into underdispersed and overdispersed processes. A renewal process is *underdispersed* (*overdispersed*) if the coefficient of variation of the distribution of the times between failures is less than (greater than) 1. An extreme case of an underdispersed process is when the coefficient of variation is 0 (i.e., deterministic interarrival times), which yields a deterministic renewal process. The underdispersed process is much more regular in its event times. In the case of a repairable system with underdispersed failure times, for example, it is easier to determine when it is appropriate to replace an item in order to avoid experiencing a failure. There is extreme clustering of events, on the other hand, in the case of an overdispersed renewal process, and replacement policies are less effective.

Alternating renewal processes

An alternating renewal process is a generalization of a renewal process that is often used to model the failure and repair times of a repairable item. Let $X_1, X_2, \ldots$ be i.i.d. random variables with positive support and cumulative distribution function $F_X(x)$ that represent the times to failure of a repairable item. Let $R_1, R_2, \ldots$ be i.i.d. random variables with positive support and cumulative distribution function $F_R(r)$ that represent the times to repair of a repairable item. Care must be taken to assure that $X_1, X_2, \ldots$ are indeed identically distributed, i.e., the item is neither improving nor deteriorating. Assuming that the alternating renewal process begins at time 0 with the item functioning, then

- X_1 is the time of the first failure,
- $X_1 + R_1$ is the time of the first repair,
- $X_1 + R_1 + X_2$ is the time of the second failure,
- $X_1 + R_1 + X_2 + R_2$ is the time of the second repair, etc.

Thus the times between events for an alternating renewal process alternate between two distributions, each with positive support.

Nonhomogeneous Poisson processes

A nonhomogeneous Poisson process is also a generalization of a Poisson process which allows for an arrival rate $\lambda(t)$ (known as the *intensity* function) that can vary with time.

Definition (Ross, 2003). The counting process $\{N(t), t \geqslant 0\}$ is said to be a *nonhomogeneous Poisson process* (NHPP) with intensity function $\lambda(t)$, $t \geqslant 0$, if

(1) $N(0) = 0$,
(2) the process has independent increments,
(3) $\mathrm{P}(N(t+h) - N(t) \geqslant 2) = \mathrm{o}(h)$, and
(4) $\mathrm{P}(N(t+h) - N(t) = 1) = \lambda(t)h + \mathrm{o}(h)$,

where a function $f(\cdot)$ is said to be $\mathrm{o}(h)$ if $\lim_{h\to 0} f(h)/h = 0$.

An NHPP is often appropriate for modeling a series of events that occur over time in a nonstationary fashion. Two common application areas are the modeling of arrivals to a waiting line (queueing theory) and the failure times of a repairable system (reliability theory) with negligible repair times. The cumulative intensity function

$$\Lambda(t) = \int_0^t \lambda(\tau)\,\mathrm{d}\tau, \qquad t > 0,$$

gives the expected number of events by time t, i.e., $\Lambda(t) = \mathrm{E}[N(t)]$. The probability of exactly n events occurring in the interval $(a, b]$ is given by

$$\frac{[\int_a^b \lambda(t)\,\mathrm{d}t]^n \exp\{-\int_a^b \lambda(t)\,\mathrm{d}t\}}{n!}$$

for $n = 0, 1, \ldots$ (Çinlar, 1975).

Other variants

Other variants of a Poisson process have been proposed. For brevity, we outline three such variants. Details are given in Resnick (1992). *Mixed Poisson processes* can be formulated in terms of an NHPP with cumulative intensity function $\Lambda(t)$ and a random variable L with positive support. The associated counting process $N(L\Lambda(t))$ is a mixed Poisson process. Transforming the time scale with the random variable L results in a process that does not, in general, have independent increments. Ross (2003, p. 328) provides an illustration from the insurance industry where L models the claim rate (which varies from one policyholder to the next) and $\Lambda(t)$ is linear. *Doubly stochastic Poisson processes* generalize the notion of transforming the time scale by embedding a stochastic process within another stochastic process. The random variable L from a mixed Poisson process is replaced with a stochastic process with nondecreasing paths. Markov modulated Poisson processes are also a special case of doubly stochastic processes. *Compound Poisson processes* are formulated with a homogeneous or nonhomogeneous Poisson process and a sequence of i.i.d. random variables $D_1, D_2, \ldots$. The function

$$C(t) = \begin{cases} \sum_{i=1}^{N(t)} D_i, & \text{if } N(t) > 0, \\ 0, & \text{otherwise,} \end{cases}$$

defines a process that increases by $D_1, D_2, \ldots$ at each event time. This would be an appropriate model for an automobile insurance company whose claims occur according to a Poisson process with claim values $D_1, D_2, \ldots$, and $C(t)$ models the total claim amounts that have occurred by time t. Similarly, if $D_1, D_2, \ldots$ are i.i.d. random variables with support on the nonnegative integers, then a compound Poisson process can be used to model batch arrivals.

1.4 Estimation

Poisson, renewal and alternating renewal processes

Poisson, renewal and alternating renewal processes all have independent interarrival times, which simplifies the estimation process. Let $x_1, x_2, \ldots, x_n$ denote independent observations collected on an interarrival time of interest from the "target" arrival process. The term *target* refers to the population arrival process that we want to estimate and simulate. These values have been assumed to be continuous in the models considered thus far. One simple trace-driven input model for variate generation would be to repeatedly select one of the data values with probability $1/n$, just as in the discrete case. This corresponds to an "empirical" c.d.f.

$$\widehat{F}(x) = \frac{M(x)}{n}, \quad x = x_i, i = 1, 2, \ldots, n,$$

where $M(x)$ is the number of data values that are less than or equal to x. One problem with this estimator is that only the data values will be generated in an associated variate generation algorithm. The interpolation problem can be overcome by filling the $n-1$ "gaps" created by the n data values, with a piecewise-linear empirical c.d.f. These $n-1$ gaps can each have a linear c.d.f. that rises $1/(n-1)$ between each of the sorted data values $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$. The piecewise-linear empirical c.d.f. is

$$\widehat{F}(x) = \frac{i-1}{n-1} + \frac{x - x_{(i)}}{(n-1)(x_{(i+1)} - x_{(i)})}, \quad x_{(i)} \leqslant x < x_{(i+1)},$$

for $i = 1, 2, \ldots, n-1$. Details concerning this model are given in Law and Kelton (2000, pp. 326 and 470) and a slight variant is presented in Banks et al. (2005, p. 281).

In the parametric case, the maximum likelihood estimation technique is generally used to estimate parameters due to its desirable statistical properties. If $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_q)'$ is a vector of unknown parameters associated with the interarrival density f, then the maximum likelihood estimator (MLE) maximizes the likelihood function

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i).$$

Statistical properties of MLEs are given in Casella and Berger (2002). Details concerning selecting the appropriate parametric distribution(s) (in the renewal and alternating renewal cases), estimating parameters, and assessing goodness-of-fit are given in Law and Kelton (2000).

Nonhomogeneous Poisson processes

This subsection describes two nonparametric (trace-driven) procedures and two parametric procedures for estimating the intensity function $\lambda(t)$ or, equivalently, the cumulative intensity function $\Lambda(t) = \int_0^t \lambda(\tau)\,\mathrm{d}\tau$ from k realizations sampled from a target NHPP. The first nonparametric procedure can be used on "count" or "interval" data. The second nonparametric procedure is appropriate for "raw" data. In each of the procedures, we find point estimates and interval estimates for the cumulative intensity function.

First consider finding an estimate for the intensity function when only count data are available. The intensity function $\lambda(t)$ or cumulative intensity function $\Lambda(t)$ is to be estimated on $(0, S]$, where S is a known constant. The interval $(0, S]$ may represent the time a system allows arrivals (e.g., 9:00 a.m. to 5:00 p.m. at a bank) or one period of a cycle (e.g., one day at an emergency room). There are k representative realizations collected on the target process on $(0, S]$.

For systems with high arrival rates (e.g., busy call centers or web sites), there is often so much data that *counts* of events that occur during time subintervals are available, rather than the raw event times. Although this is less preferable that having the raw data, it is still possible to construct an estimate of the intensity function and generate variates for a discrete-event simulation model. The time interval $(0, S]$ can be partitioned into m subintervals $(a_0, a_1], (a_1, a_2], \ldots, (a_{m-1}, a_m]$, where $a_0 = 0$ and $a_m = S$. The subintervals do not necessarily have equal widths. Let $n_1, n_2, \ldots, n_m$ be the total number of observed events in the subintervals over the k realizations.

To simplify the estimation process, assume that the target NHPP has an intensity function $\lambda(t)$ that is piecewise constant on each subinterval of the partition $(a_0, a_1], (a_1, a_2], \ldots, (a_{m-1}, a_m]$. Since the average intensity function on the interval $(a_{i-1}, a_i]$ is the rate per unit time of the events that occur on that interval, the maximum likelihood estimator is the average number of events that occurred on the interval, normalized for the length of the interval (Law and Kelton, 2000, pp. 390–393; Leemis, 2004)

$$\hat{\lambda}(t) = \frac{n_i}{k(a_i - a_{i-1})}, \qquad a_{i-1} < t \leqslant a_i,$$

for $i = 1, 2, \ldots, m$. Since the intensity estimator is piecewise constant, the associated cumulative intensity function estimator is a continuous, piecewise-linear function on $(0, S]$

$$\widehat{\Lambda}(t) = \int_0^t \hat{\lambda}(\tau)\,\mathrm{d}\tau = \left(\sum_{j=1}^{i-1} \frac{n_j}{k}\right) + \frac{n_i(t - a_{i-1})}{k(a_i - a_{i-1})}, \qquad a_{i-1} < t \leqslant a_i,$$

for $i = 1, 2, \ldots, m$. (If there are no events observed on interval i, i.e., $n_i = 0$, then the intensity function estimate is zero on interval i and the cumulative intensity function estimate is constant on interval i.) This estimator passes through the points $(a_i, \sum_{j=1}^{i} n_j/k)$ for $i = 1, 2, \ldots, m$. Asymptotic properties of this estimator in the case of equal-width subintervals are considered by Henderson (2003). Variate generation via inversion is straightforward.

It is important to assess the precision of the point estimator, which is developed thus far, which is typically done via confidence intervals. Based on the fact that the number of events by time t has a Poisson($\Lambda(t)$) distribution, an approximate, two-sided $(1-\alpha)100\%$ confidence interval for $\Lambda(t)$ is

$$\widehat{\Lambda}(t) - z_{\alpha/2}\sqrt{\frac{\widehat{\Lambda}(t)}{k}} < \Lambda(t) < \widehat{\Lambda}(t) + z_{\alpha/2}\sqrt{\frac{\widehat{\Lambda}(t)}{k}}$$

for $0 < t \leqslant S$, where $z_{\alpha/2}$ is the $1-\alpha/2$ fractile of the standard normal distribution. The interval is always asymptotically exact at the endpoints, but only asymptotically exact for all t in $(0, S]$ when the target intensity function $\lambda(t)$ is piecewise constant over each subinterval $(a_{i-1}, a_i]$ in the arbitrary partition of $(0, S]$. In most applications, this assumption is *not* satisfied.

Now consider the case where *raw event times* are available for the k realizations collected on $(0, S]$. The meaning of n_i now changes from the count data case. We now let n_i, $i = 1, 2, \ldots, k$, be the number of observations in the ith realization, $n = \sum_{i=1}^{k} n_i$, and let $t_{(1)}, t_{(2)}, \ldots, t_{(n)}$ be the order statistics of the superposition of the event times in the k realizations, $t_{(0)} = 0$ and $t_{(n+1)} = S$.

The standard step-function estimator for the cumulative intensity function $\Lambda(t)$ takes upward steps of height $1/k$ only at the event times in the superposition of the k realizations collected from the target process. This means that a variate generation algorithm will produce realizations that only contain event times that are equal to the ones that were collected in one of the k realizations. This is known as the "interpolation" problem, which can be overcome by using a piecewise-linear cumulative intensity function estimator.

There are $n + 1$ "gaps" on $(0, S]$ created by the superposition $t_{(1)}, t_{(2)}, \ldots, t_{(n)}$. Setting $\widehat{\Lambda}(S) = n/k$ 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 cumulative intensity function estimate rises by $n/[(n+1)k]$ at each event time in the superposition. Thus the piecewise-linear estimator of the cumulative intensity function between the time values in the superposition is

$$\widehat{\Lambda}(t) = \frac{in}{(n+1)k} + \left[\frac{n(t - t_{(i)})}{(n+1)k(t_{(i+1)} - t_{(i)})}\right], \quad t_{(i)} < t \leqslant t_{(i+1)},$$

for $i = 0, 1, 2, \ldots, n$. This estimator passes through the points $(t_{(i)}, in/(n+1)k)$, for $i = 1, 2, \ldots, n+1$. This estimator was developed in Leemis (1991) and extended to nonoverlapping intervals in Arkin and Leemis (2000).

A $(1 - \alpha)100\%$ asymptotically exact (as $k \to \infty$) confidence interval for $\Lambda(t)$ is given by

$$\widehat{\Lambda}(t) - z_{\alpha/2}\sqrt{\frac{\widehat{\Lambda}(t)}{k}} < \Lambda(t) < \widehat{\Lambda}(t) + z_{\alpha/2}\sqrt{\frac{\widehat{\Lambda}(t)}{k}}$$

for $0 < t \leqslant S$, where $z_{\alpha/2}$ is the $1 - \alpha/2$ fractile of the standard normal distribution.

The estimation procedure given here is nonparametric and does not require any arbitrary decisions (e.g., parameter values) from the modeler. The fact that each of the piecewise segments rises the same height can be exploited to create an efficient variate generation algorithm via inversion. As n increases, the amount of memory required increases, but the amount of execution time required to generate a realization depends only on the ratio n/k, the average number of events per realization. Thus, collecting more realizations (resulting in narrower confidence intervals) increases the amount of memory required, but does not impact the expected execution time for generating a realization.

To summarize, this piecewise-linear cumulative intensity function estimator with raw data is in some sense ideal in that

- it uses raw data which avoids the loss of accuracy imposed by breaking $(0, S]$ into arbitrary time subintervals,
- the point and interval estimates of $\Lambda(t)$ are closed form and easily computed,
- the point estimate of $\Lambda(t)$ is consistent, i.e., $\lim_{k\to\infty} \widehat{\Lambda}(t) = \Lambda(t)$ with probability 1, which is proved using the strong law of large numbers,
- the interval estimate of $\Lambda(t)$ is asymptotically exact as $k \to \infty$, which is proved using the central limit theorem and Slutsky's theorem, and
- the variate generation algorithm which can be used to simulate a realization from $\widehat{\Lambda}(t)$ is efficient, monotone, and synchronized (see Chapter 4 for definitions of these terms).

One downside over a parametric approach is that the variate generation algorithm is storage intensive, since all arrival times collected must be stored in memory.

We now shift to the parametric modeling of NHPPs. Maximum likelihood can again be used for estimating the parameters in a parametric NHPP model. The likelihood function for estimating the vector of unknown parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_q)'$ from a single realization of event times $t_1, t_2, \ldots, t_n$ drawn from an NHPP with intensity $\lambda(t)$ on $(0, S]$ is

$$L(\boldsymbol{\theta}) = \left[\prod_{i=1}^{n} \lambda(t_i)\right] \exp\left[-\int_0^S \lambda(t)\,\mathrm{d}t\right].$$

Maximum likelihood estimators can be determined by maximizing $L(\boldsymbol{\theta})$ or its logarithm with respect to the q unknown parameters. Asymptotically exact confidence regions for the unknown parameters can be found via the likelihood ratio statistic.

Because of the additive property of the intensity function for multiple realizations, the likelihood function for the case of k realizations is

$$L(\boldsymbol{\theta}) = \left[\prod_{i=1}^{n} k\lambda(t_i)\right] \exp\left[-\int_0^S k\lambda(t)\,\mathrm{d}t\right].$$

Example. There are many potential parametric models for nonhomogeneous Poisson processes. Consider a *power law process*, where the intensity function is

$$\lambda(t) = b^a a t^{a-1}, \quad t > 0,$$

for shape parameter $a > 0$ and scale parameter $b > 0$. The analysis for other intensity functions is similar. This intensity function can assume monotone increasing and monotone decreasing shapes. Thus, the likelihood function for k realizations is

$$L(b, a) = k^n b^{na} a^n \mathrm{e}^{-k(bS)^a} \prod_{i=1}^{n} t_i^{a-1}.$$

The log-likelihood function is

$$\log L(b, a) = n\log(ka) + na\log b - k(bS)^a + (a-1)\sum_{i=1}^{n} \log t_i.$$

Differentiating with respect to a and b and equating to zero yields

$$\frac{\partial \log L(b, a)}{\partial a} = n\log b + \frac{n}{a} + \sum_{i=1}^{n} \log t_i - k(bS)^a \log(bS) = 0$$

and

$$\frac{\partial \log L(b, a)}{\partial b} = \frac{an}{b} - kS^a a b^{a-1} = 0.$$

These equations can be solved in closed form. The analytic expressions for b and a are:

$$\hat{a} = \frac{n}{n\log S - \sum_{i=1}^{n} \log t_i},$$
$$\hat{b} = \frac{1}{S}\left(\frac{n}{k}\right)^{1/a}.$$

It is oftentimes the case that a standard parametric model such as a power law process is unable to adequately describe the intensity function. When this is the case, the EPTMP (exponential-polynomial-trigonometric function with multiple periodicities) model, originally given by Lee et al. (1991) and generalized by Kuhl et al. (1997) with intensity function

$$\lambda(t) = \exp\left[\sum_{i=0}^{m} \alpha_i t^i + \sum_{j=1}^{p} \gamma_j \sin(\omega_j t + \phi_j)\right], \quad t > 0,$$

can also model a nonmonotonic intensity function. The exp function forces the intensity function to be positive, the first summation models a polynomial trend and the second summation models sinusoidal periodicities in the intensity. The cyclic portion of the model has been used in discrete-event simulation applications to model storm times in the Arctic Sea (Lee et al., 1991) and arrivals of livers for transplantation by donors (Pritsker et al., 1995). Goodness-of-fit tests associated with the fitted models are given in Rigdon and Basu (2000).

1.5 Process generation

The algorithms presented in this section generate a sequence of event times (in our setting they are typically the arrival times in a discrete-event simulation model) on the time interval $(0, S]$, where S is a real, fixed constant. If the next-event approach is taken for placing events onto the calendar in a discrete-event simulation model, then these algorithms should be modified so that they take the current event time as an argument and return the next event time. All processes are assumed to begin at time 0. The event times that are generated by the counting process are denoted by $T_1, T_2, \ldots$, and random numbers (i.e., U(0, 1) random variables) are denoted by U or $U_1, U_2, \ldots$. If just $T_0 = 0$ is returned, then no arrivals were observed on $(0, S]$. Indentation is used in the algorithms to indicate nesting.

Poisson processes

Since the times between arrivals in a Poisson process are i.i.d. exponential(λ) random variables, the following algorithm generates the arrival times of a Poisson process on $(0, S]$.

```
$T_0 \leftarrow 0$
$i \leftarrow 0$
**while** $T_i \leqslant S$
   $i \leftarrow i + 1$
   **generate** $U_i \sim \mathrm{U}(0, 1)$
   $T_i \leftarrow T_{i-1} - \log(1 - U_i)/\lambda$
**return** $T_1, T_2, \ldots, T_{i-1}$
```

Renewal processes

Arrivals in a renewal process are generated in a similar fashion to a Poisson process. Let $F_X(x)$ denote the cumulative distribution function of the interarrival times $X_1, X_2, \ldots$ in a renewal process. The following algorithm generates the arrival times on $(0, S]$.

```
$T_0 \leftarrow 0$
$i \leftarrow 0$
**while** $T_i \leqslant S$
    $i \leftarrow i + 1$
    **generate** $U_i \sim \mathrm{U}(0, 1)$
    $T_i \leftarrow T_{i-1} + F_X^{-1}(U_i)$
**return** $T_1, T_2, \ldots, T_{i-1}$
```

Alternating renewal processes

Arrivals in an alternating renewal process are generated in a similar fashion to a renewal process, but the interarrival time must alternate between $F_X(x)$, the cumulative distribution function of the times to failure $X_1, X_2, \ldots$ and $F_R(r)$, the cumulative distribution function of the times to repair $R_1, R_2, \ldots$. The following algorithm generates the arrival times on $(0, S]$.

```
$T_0 \leftarrow 0$
$i \leftarrow 0$
$j \leftarrow 0$
**while** $T_i \leqslant S$
    $i \leftarrow i + 1$
    **generate** $U_i \sim \mathrm{U}(0, 1)$
    **if** $j = 0$
        $T_i \leftarrow T_{i-1} + F_X^{-1}(U_i)$
        $j \leftarrow j + 1$
    **else**
        $T_i \leftarrow T_{i-1} + F_R^{-1}(U_i)$
        $j \leftarrow j - 1$
**return** $T_1, T_2, \ldots, T_{i-1}$
```

Nonhomogeneous Poisson process

Arrivals can be generated for use in discrete-event simulation as $\Lambda^{-1}(E_1)$, $\Lambda^{-1}(E_2), \ldots$, where $E_1, E_2, \ldots$ are the event times in a unit Poisson process (Çinlar, 1975). This technique is often referred to as "inversion" and is implemented below.

```
$T_0 \leftarrow 0$
$E_0 \leftarrow 0$
$i \leftarrow 0$
**while** $T_i \leqslant S$
    $i \leftarrow i + 1$
    **generate** $U_i \sim \mathrm{U}(0, 1)$
    $E_i \leftarrow E_{i-1} - \log(1 - U_i)$
    $T_i \leftarrow \Lambda^{-1}(E_i)$
**return** $T_1, T_2, \ldots, T_{i-1}$
```

The inversion algorithm is ideal when $\Lambda(t)$ can be inverted analytically, although it also applies when $\Lambda(t)$ needs to be inverted numerically. There may be occasions when the numerical inversion of $\Lambda(t)$ is so onerous that the *thinning* algorithm devised by Lewis and Shedler (1979) and given below may be preferable. This algorithm assumes that the modeler has determined a *majorizing* value λ^* that satisfies $\lambda^* \geqslant \lambda(t)$ for all $t > 0$.

```
$T_0 \leftarrow 0$
$i \leftarrow 0$
**while** $T_i \leqslant S$
    $t \leftarrow T_i$
    **repeat**
        **generate** $U \sim \mathrm{U}(0, 1)$
        $t \leftarrow t - \log(1 - U)/\lambda^*$
        **generate** $U \sim \mathrm{U}(0, \lambda^*)$
    **until** $U \leqslant \lambda(t)$
    $i \leftarrow i + 1$
    $T_i \leftarrow t$
**return** $T_1, T_2, \ldots, T_{i-1}$
```

The majorizing value λ^* can be generalized to a majorizing function $\lambda^*(t)$ to decrease the CPU time by minimizing the probability of "rejection" in the **repeat–until** loop.

All of the arrival process models given in this section are elementary. More complex models are considered in Nelson et al. (1995).

2 Generating random lifetimes

This section concerns algorithms for generating continuous, positive random variables, referred to here as "lifetimes". Although the main two applications areas are reliability (e.g., a machine or product lifetime, see, for example, Meeker and Escobar, 1998) and survival analysis (e.g., patient survival time after an organ transplant, see, for example, Lawless, 2003), their use is widespread in other disciplines (e.g., sociological applications as in Allison, 1984). The discussion here is limited to generating *continuous*, as opposed to discrete or mixed distributions, due to their pervasiveness in the reliability and survival analysis literature.

2.1 Hazard-based methods

There are three hazard-based methods for generating lifetimes which parallel the associated density-based methods that were introduced in Chapter 4: (1) the inverse cumulative hazard function technique, an inversion technique that parallels the inverse-c.d.f. technique, (2) competing risks, a linear combination technique that parallels composition, and (3) thinning, a majorization technique that parallels acceptance/rejection.

Definitions

Consider a positive, continuous random variable T, referred to here as a "lifetime". We briefly introduce four functions, each of which completely describes the distribution of T: the survivor function, the probability density function, the hazard function, and the cumulative hazard function.

The survivor function, also known as the reliability function and complementary c.d.f., is defined by

$$S(t) = \mathrm{P}(T \geqslant t), \quad t > 0,$$

a nonincreasing function of t satisfying $S(0) = 1$ and $\lim_{t\to\infty} S(t) = 0$. The survivor function is important in the study of systems of components since it is the appropriate argument in the structure function to determine system reliability (Meeker and Escobar, 1998). Notice that $S(t)$ 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 $F(t) = \mathrm{P}(T \leqslant t)$ is the c.d.f.

When the survivor function is differentiable,

$$f(t) = -S'(t), \quad t \geqslant 0,$$

is the associated p.d.f. For any interval (a, b), where $a < b$,

$$\mathrm{P}(a \leqslant T \leqslant b) = \int_a^b f(t)\,\mathrm{d}t.$$

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

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

The hazard function is popular in reliability 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 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 subsequently.

The cumulative hazard function can be defined by

$$H(t) = \int_0^t h(\tau)\,\mathrm{d}\tau, \quad t \geqslant 0.$$

Any one of the four functions that describes the distribution of T can be used to determine the others, e.g., $H(t) = -\log S(t)$.

We now introduce the three hazard-based lifetime variate generation algorithms: the inverse cumulative hazard function technique, competing risks, and thinning. Random numbers are denoted by U and the associated random lifetimes are denoted by T.

Inverse cumulative hazard function technique

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, which is an extension of the probability integral transformation, is the basis for the inverse cumulative hazard function technique. Therefore,

```
generate $U \sim \mathrm{U}(0, 1)$
$T \leftarrow H^{-1}(-\log(1 - U))$
return $T$
```

generates a single random lifetime T. This algorithm is easiest to implement when H can be inverted in closed form. This algorithm is monotone and synchronized. Although the sense of the monotonicity is reversed, $1 - U$ can be replaced with U in order to save a subtraction.

Competing risks

Competing risks (David and Moeschberger, 1978) is a linear combination technique that is analogous to the density-based composition method. The composition method is viable when the p.d.f. can be written as the convex combination of k density functions

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

where $\sum_{j=1}^{k} p_j = 1$. The competing risks technique applies when the hazard function can be written as the sum of hazard functions, each corresponding to a "cause" of failure

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

where $h_j(t)$ is the hazard function associated with cause j of failure acting in a population. The minimum of the lifetimes from each of these risks corresponds to the system lifetime. Competing risks is most commonly used to analyze a series system of k components, but it can also be used in actuarial applications with k causes of failure. The competing risks model is also used for modeling competing failure modes for components that have multiple failure modes. The algorithm to generate a lifetime T is

```
for $j$ from 1 to $k$
    generate $T_j \sim h_j(t)$
$T \leftarrow \min\{T_1, T_2, \ldots, T_k\}$
return $T$
```

The $T_1, T_2, \ldots, T_k$ values can be generated by any of the standard random variate generation algorithms.

Thinning

The third class of techniques for generating random lifetimes is majorization techniques. The density-based acceptance/rejection technique uses a *majorizing function* $f^*(t)$ that satisfies $f^*(t) \geqslant f(t)$ for all $t \geqslant 0$. Likewise *thinning*, which was originally suggested by Lewis and Shedler (1979) for generating the event times in a nonhomogeneous Poisson process, can be adapted to produce a single lifetime by returning only the first event time generated. A majorizing hazard function $h^*(t)$ must be found that satisfies $h^*(t) \geqslant h(t)$ for all $t \geqslant 0$. The algorithm is

```
$T \leftarrow 0$
**repeat**
    **generate** $Y$ **from** $h^*(t)$ **given** $Y > T$
    $T \leftarrow T + Y$
    **generate** $S \sim \mathrm{U}(0, h^*(T))$
**until** $S \leqslant h(T)$
**return** $T$
```

Generating Y in the loop can be performed by inversion or any other method. The name *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.

2.2 Survival models involving covariates

The accelerated life and proportional hazards lifetime models can account for the effects of covariates on a random lifetime (Cox and Oakes, 1984). Variate generation for these models is a straightforward extension of the basic methods for generating random lifetimes. Variate generation algorithms for Monte Carlo simulation of nonhomogeneous Poisson processes are a simple extension of the inverse cumulative hazard function technique.

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, these covariates are exogenous variables such as the turning speed of a machine tool or the stress applied to a component that affect the lifetime of an item. We use the generic term *item* here to represent a manufactured product or organism whose survival time is of interest. Two common models to incorporate the effect of the covariates on lifetimes are the *accelerated life* and *Cox proportional hazards* models.

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

Accelerated life model

The cumulative hazard function for T in the accelerated life model is

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

where H_0 is a baseline cumulative hazard function. When $\mathbf{z} = 0$, $H(t) = H_0(t)$. In this model, the covariates accelerate ($\psi(\mathbf{z}) > 1$) or decelerate ($\psi(\mathbf{z}) < 1$) the rate that the item moves through time.

Proportional hazards model

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 hazard function associated with the lifetime of the item by the factor $\psi(\mathbf{z})$ for all values of t. This model is known in medicine as the "Cox model" and is a standard model for evaluating the effect of covariates on survival. We do not explicitly consider the estimation of the regression coefficients $\boldsymbol{\beta}$ here since the focus is on random lifetime generation. Cox and Oakes (1984) and others give the details associated with estimation of $\boldsymbol{\beta}$ and most modern statistical packages estimate these coefficients using built-in numerical methods.

Lifetime generation

All of 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 $-\log(1-U)$, where $U \sim \mathrm{U}(0, 1)$, and solving for t yields the appropriate generation technique (Leemis, 1987).

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 an NHPP.

In an NHPP, the hazard function, $h(t)$, is analogous to the intensity function $\lambda(t)$, which governs the rate at which events occur. To determine the appropriate method for generating variates from an NHPP model which involves covariates, 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

Table 1.
Lifetime generation in regression survival models

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

survival to time a, is

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

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

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

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 baseline case) because of the memoryless property associated with the exponential distribution.

3 Generating random objects

The notion of a random "object" is quite general. We therefore limit our discussion to (1) combinatorial objects, (2) matrices, and (3) polynomials. The wider literature on generating random objects is rather diverse, ranging from a monograph on shuffling playing cards (Morris, 1998), to generating random colors, to generating random geometric objects, to generating random spawning trees (Fishman, 1996, p. 379), to generating random sequences (Knuth, 1998), to generating Markov chains (Gilks et al., 1996; Ross, 2003; Metropolis et al., 1953; Hastings, 1970).

3.1 Combinatorial objects

A combinatorial object is an object which can be placed into a one-to-one correspondence with a finite set of integers. These objects can be used for a variety of purposes (e.g., entries in a matrix, coefficients of a polynomial). The algorithms given in the subsections below are from Nijenhuis and Wilf (1978), Wilf (1989) and Devroye (1986). These authors also include algorithms for generating *all* combinatorial objects in lexicographic order, as opposed to generating *one* random combinatorial object, which is the emphasis here. We consider only random subsets, combinations, and permutations here. Algorithms for generating other combinatorial objects (e.g., a random composition of n into k parts or a random partition of an integer n) are also considered by these authors.

Generating a random subset of $\{1, 2, \ldots, n\}$

Generating a random subset of the first n positive integers is straightforward since the inclusion or exclusion of each element in the subset is a Bernoulli(1/2) random variable. Thus if $a_i = 0$ if i is excluded from the subset and $a_i = 1$ if i is included in the subset, the following $O(n)$ algorithm generates the $a_1, a_2, \ldots, a_n$ values.

```
for $i$ from 1 to $n$
    generate $U \sim U(0, 1)$
    $a_i \leftarrow \lfloor 2U \rfloor$
return $a_1, a_2, \ldots, a_k$
```

In this fashion, each of the 2^n subsets is generated with probability $1/2^n$.

Generating a random combination of k integers from $\{1, 2, \ldots, n\}$

The naive algorithm for generating a random combination of k integers drawn from $\{1, 2, \ldots, n\}$ is to repeatedly generate random integers between 1 and n, discarding those that have been generated previously. This inefficient $O(k^2)$ algorithm which generates the combination in an unsorted order can be improved on by the following $O(k)$ algorithm which uses exactly k memory locations $a_1, a_2, \ldots, a_k$ for storing the combination and produces sorted output.

```
$c_1 \leftarrow k$
$c_2 \leftarrow n$
$k_0 \leftarrow 0$
$i \leftarrow 1$
while $c_1 > 0$
    generate $U \sim U(0, 1)$
    if $U \leqslant c_1/c_2$
        $c_1 \leftarrow c_1 - 1$
        $k_0 \leftarrow k_0 + 1$
        $a_{k_0} \leftarrow i$
    $c_2 \leftarrow c_2 - 1$
```

```
        $i \leftarrow i + 1$
    return $a_1, a_2, \ldots, a_k$
```

This algorithm progressively chooses each element with the appropriate probability (c_1/c_2) based on the number of elements remaining in the combination and the number remaining in the selection pool, terminating when the kth element is selected for inclusion in the combination. In this fashion, each of the $\binom{n}{k}$ combinations is generated with probability $1/\binom{n}{k}$.

Generating a random permutation of $\{1, 2, \ldots, n\}$

There are exactly $n!$ permutations of the integers $\{1, 2, \ldots, n\}$. An algorithm that generates each of these permutations with probability $1/n!$ is referred to as a *random shuffle*. If $n = 52$, for example, a random shuffle algorithm generates one of the permutations of an ordinary deck of playing cards. The intuitive way to generate a random shuffle of $\{1, 2, \ldots, n\}$ is to (1) select an integer from $\{1, 2, \ldots, n\}$, (2) select an integer from $\{1, 2, \ldots, n\}$ less the integer selected in the previous step, etc. The following $\mathrm{O}(n)$ algorithm does just that while leaving the elements of $\{1, 2, \ldots, n\}$ in place. Prior to executing the algorithm, the first n integers should be stored (in any order) in $a_1, a_2, \ldots, a_n$.

```
for $i$ from 1 to $n$
    generate $U \sim \mathrm{U}(0, 1)$
    $j \leftarrow \lfloor (n - i + 1)U \rfloor + i$
    $t \leftarrow a_j$
    $a_j \leftarrow a_i$
    $a_i \leftarrow t$
return $a_1, a_2, \ldots, a_n$
```

The $j \leftarrow \lfloor (n - i + 1)U \rfloor + i$ step is used to generate a random subscript that is equally likely among $i, i + 1, \ldots, n$. The three steps that follow are used to swap a_i and a_j.

This completes the discussion of algorithms for the generation of three fundamental combinatorial objects: subsets, combinations, and permutations. Other sampling schemes (e.g., sampling with replacement) or other combinatorial objects (e.g., a random partition of an integer) are considered by the authors cited at the beginning of this subsection.

3.2 Random matrices

A random matrix is an $n \times m$ array of entries that are random variables. Generating random matrices is a topic without a well-organized theory. Carmeli (1983), Deift (2000) and Mehta (2004), for example, emphasize applications of random matrices in quantum mechanics where, for example, the energy levels of a system are given by the eigenvalues of a Hermitian operator called the Hamiltonian. When the complex entries are random variables, the associated eigenvalues are also random. Rather than attempt to create a hierarchy for random matrices (e.g., the meaning of a random positive definite matrix), we focus on three examples that illustrate the wide variety of questions that can be posed concerning random matrices: (1) the generation of a particular type of matrix known as a "correlation matrix", (2) the distribution of the number of real eigenvalues associated with a matrix of independent standard normal random variables, and (3) the distribution of the determinant of an M-matrix with U(0, 1) entries.

Generating correlation matrices

A correlation matrix is a symmetric, positive semidefinite matrix with 1's on the diagonal. The (i, j)th element of such a matrix gives the correlation between the ith and jth elements of a random vector. Marsaglia and Olkin (1984) consider the generation of random correlation matrices under the following three scenarios: (1) random correlation matrices with a given mean, (2) random correlation matrices of the form TT', and (3) random correlation matrices with given eigenvalues. We consider the first case.

Let C be a given correlation matrix and X be a random symmetric matrix with zeros on the diagonal and random variables with means of zero for the off-diagonal elements. The matrix $C + X$ is a random correlation matrix with expected value C if and only if the eigenvalues of $C + X$ are nonnegative. Thus X is a perturbation of C that can be used to generate the random correlation matrix $C + X$ under the appropriate restrictions on X. This setting provides a variety of methods for generating random correlation matrices. Here is one such algorithm:

Set up: Compute λ, the smallest eigenvalue of the $n \times n$ matrix C.
Marginal: Generate the upper-triangular elements of X from a radially symmetric distribution in (or on) the unit $n(n-1)/2$-sphere. Letting $x_{ij} = x_{ji}$ for $i \neq j$, the matrix $C + \lambda X/\sqrt{2}$ is a random correlation matrix with expected value C.

Example. Consider the generation of a random correlation matrix when $n = 3$. One way to generate the random elements of the perturbation matrix X is to generate values on the surface of the unit sphere. Using the transformation from spherical coordinates to rectangular coordinates:

$$x_{12} = \rho \sin\phi \cos\theta, \qquad x_{13} = \rho \sin\phi \sin\theta, \qquad x_{23} = \rho \cos\phi,$$

where $\theta \sim \mathrm{U}(0, 2\pi)$, $\phi \sim \mathrm{U}(0, \pi)$ and $\rho = 1$. Using the third random number stream, a perturbation matrix X can be generated in S-Plus (S-Plus is a registered trademark of Insightful Corporation) with the statements:

```
set.seed(3)
x <- matrix(0, 3, 3)
theta <- runif(1, 0, 2 * pi)
phi <- runif(1, 0, pi)
x[1, 2] <- sin(phi) * cos(theta)
x[1, 3] <- sin(phi) * sin(theta)
x[2, 3] <- cos(phi)
```

```
    x[2, 1] <- x[1, 2]
    x[3, 1] <- x[1, 3]
    x[3, 2] <- x[2, 3]
```
yielding

$$X = \begin{bmatrix} 0.0000000 & -0.3956487 & -0.8461130 \\ -0.3956487 & 0.0000000 & 0.3571483 \\ -0.8461130 & 0.3571483 & 0.0000000 \end{bmatrix}.$$

The associated random correlation matrix is thus $C + \lambda X/\sqrt{2}$.

Ghosh and Henderson (2003) give another method for generating correlation matrices.

Number of real eigenvalues

Some random matrices have analytic results that may preclude the need for simulation. This paragraph outlines one such case. An $n \times n$ matrix with independent standard normal entries, for example, has an expected number of real eigenvalues E_n given by Edelman et al. (1994):

$$E_n = \begin{cases} 1 + \sqrt{2}\sum_{k=1}^{(n-1)/2} \frac{(4k-3)!!}{(4k-2)!!}, & n \text{ odd}, \\ \sqrt{2}\sum_{k=0}^{n/2-1} \frac{(4k-1)!!}{(4k)!!}, & n \text{ even}, \end{cases}$$

where the double factorial (also known as the semifactorial) notation is defined by

$$n!! = \begin{cases} 1 \times 3 \times 5 \times \cdots \times n, & n \text{ odd}, \\ 2 \times 4 \times 6 \times \cdots \times n, & n \text{ even}. \end{cases}$$

The authors also show that E_n asymptotically converges to $\sqrt{2n/\pi}$ as $n \to \infty$ and that the distribution of the value of a real normalized eigenvalue (the eigenvalue divided by $\sqrt{n}$) of such a matrix converges in distribution to a $U(-1, 1)$ distribution. Other distributions for the entries of the matrix or correlated entries would likely require Monte Carlo simulation to determine the expected number of real eigenvalues or the distribution of a real normalized eigenvalue.

Example. The analytic result indicates that the expected number of real eigenvalues of a 3×3 matrix of independent standard normal random variables is $E_3 = 1+\sqrt{2}/2 \cong 1.71$. The following Monte Carlo simulation S-Plus code verifies the analytic result (it yields an estimated expected number of real eigenvalues as 1.738, 1.698 and 1.676 for random number streams 3, 4 and 5, which corresponds to an approximate 95% confidence interval of $1.63 < E_3 < 1.78$ based on the assumption of normal sampling) and can easily be modified for other independent or dependent matrix entries where no analytic result exists.

```
set.seed(3)
total.real <- 0
for (i in 1:1000) {
  x <- matrix(rnorm(9), 3, 3)
  total.real <- total.real
                + sum(Im(eigen(x)values) == 0)
}
total.real / 1000
```

Determinant of an M-matrix

Finally, we consider the following question: if the elements of a 3×3 real matrix are independent random numbers with positive diagonal elements and negative off-diagonal elements, what is the probability that the matrix has a positive determinant? That is, find the probability that

$$\begin{vmatrix} +u_{11} & -u_{12} & -u_{13} \\ -u_{21} & +u_{22} & -u_{23} \\ -u_{31} & -u_{32} & +u_{33} \end{vmatrix} > 0,$$

where the u_{ij}'s are independent random numbers. This question is rather vexing analytically due to the appearance of some of the random numbers multiple times in the determinant. This question is of interest to matrix theorists as it is an example of a probability that cannot be calculated easily. A positive determinant in this case is equivalent to the matrix being of a special type called an *M-matrix* (Horn and Johnson, 1990). A positive determinant in a matrix with this particular sign pattern (positive diagonal elements and negative off-diagonal elements) corresponds to having all three 2×2 principal minors (determinants of sub-matrices determined by deleting the same numbered row and column) being positive.

The implementation of the Monte Carlo simulation to estimate the probability of a positive determinant is straightforward. Random matrices are generated in a loop, counting the number of these matrices that have a positive determinant and returning the ratio of the count of the matrices that have a positive determinant to the number of replications. In order to estimate the probability with some precision, it is reasonable to make one long run. For the linear congruential generator $g(x) = ax \bmod m$ with $a = 48271$, $m = 2^{31} - 1$, and a seed of $x_0 = 123456789$ (see Chapter 3), and 200,000,000 replications, the program returns an estimated probability of a positive determinant as 0.050203. Two other simulation runs with different random number generators indicate that only three digits are reasonable to report: the estimated probability of a positive determinant is 0.0502.

3.3 Random polynomials

Like random matrices, the diversity of questions that can be asked concerning random polynomials is also very large. Edelman and Kostlan (1995), for example, use elementary geometric arguments to show that the expected number of real zeros E_n of a degree n polynomial with independent standard normal coefficients is given by the expression

$$E_n = \frac{4}{\pi} \int_0^1 \sqrt{\frac{1}{(1-t^2)^2} - \frac{(n+1)^2 t^{2n}}{(1-t^{2n+2})^2}}\,\mathrm{d}t.$$

The authors show that for a seemingly minor variation in this problem (the independent, normally distributed coefficients have means 0 and variance $\binom{n}{i}$ for the ith coefficient), then the random polynomial has $E_n = \sqrt{n}$ expected real zeros. They also extend these results for arbitrary and multivariate normal distributions.

Example. The Monte Carlo simulation S-Plus code below verifies that $E_2 = \sqrt{2} \cong 1.41421$ for quadratic equations, and can be modified to accommodate more general random coefficients of the polynomial. The program below returns 1.41394, 1.40886 and 1.41636 as estimates for the expected number of real roots for random number streams 3, 4 and 5, resulting in an approximate 95% confidence interval of $1.40 < E_2 < 1.42$ based on the assumption of normal sampling, which verifies the theoretical result

```
set.seed(3)
num.roots <- 0
for (i in 1:100000) {
  aa <- rnorm(1, 0, 1)
  bb <- rnorm(1, 0, sqrt(2))
  cc <- rnorm(1, 0, 1)
  if (bb ^ 2 - 4 * aa * cc > 0)
    num.roots <- num.roots + 2
}
num.roots / 100000
```

Acknowledgements

The author gratefully acknowledges the helpful comments from the editors and a referee.

References

Allison, P.D. (1984). *Event History Analysis: Regression for Longitudinal Event Data*. Sage Publications, Beverly Hills, CA.

Arkin, B.L., Leemis, L.M. (2000). Nonparametric estimation of the cumulative intensity function for a nonhomogeneous Poisson process from overlapping realizations. *Management Science* 46 (7), 989–998.

Banks, J., Carson II, J.S., Nelson, B.L., Nicol, D.M. (2005). *Discrete-Event System Simulation*, 4th edition. Prentice-Hall, Englewood Cliffs, NJ.

Carmeli, M. (1983). *Statistical Theory and Random Matrices*. Marcel Dekker, New York.

Casella, G., Berger, R.L. (2002). *Statistical Inference*, 2nd edition. Duxbury, Pacific Grove, CA.

Çinlar, E. (1975). *Introduction to Stochastic Processes*. Prentice-Hall, Englewood Cliffs, NJ.

Cox, D.R., Isham, V. (1980). *Point Processes*. Chapman and Hall, New York.

Cox, D.R., Oakes, D. (1984). *Analysis of Survival Data*. Chapman and Hall, New York.

David, H.A., Moeschberger, M.L. (1978). *The Theory of Competing Risks*. Macmillan, New York.

Deift, P. (2000). *Orthogonal Polynomials and Random Matrices: A Riemann–Hilbert Approach*. American Mathematical Society, Providence, RI.

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

Edelman, A., Kostlan, E. (1995). How many zeros of a random polynomial are real? *Bulletin of the American Mathematical Society* 32 (1), 1–37.

Edelman, A., Kostlan, E., Shub, M. (1994). How many eigenvalues of a random matrix are real? *Journal of the American Mathematical Society* 7, 247–267.

Fishman, G.S. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, New York.

Ghosh, S., Henderson, S.G. (2003). Behavior of the NORTA method for correlated random vector generation as the dimension increases. *ACM Transaction on Modeling and Computer Simulation* 13, 276–294.

Gilks, W.R., Richardson, S., Spiegelhalter, D.J. (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, New York.

Gross, D., Harris, C.M. (1985). *Fundamentals of Queueing Theory*, 2nd edition. Wiley, New York.

Hastings, W.K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57, 97–109.

Henderson, S.G. (2003). Estimation of nonhomogeneous Poisson processes from aggregated data. *Operations Research Letters* 31 (5), 375–382.

Horn, R.A., Johnson, C.R. (1990). *Topics in Matrix Analysis*. Cambridge University Press, Cambridge.

Knuth, D.E. (1998). *The Art of Computer Programming, volume 2: Seminumerical Algorithms*, 3rd edition. Addison–Wesley, Reading, MA.

Kuhl, M.E., Wilson, J.R., Johnson, M.A. (1997). Estimating and simulating Poisson processes having trends and multiple periodicities. *IIE Transactions* 29, 201–211.

Law, A.M., Kelton, W.D. (2000). *Simulation Modeling and Analysis*, 3rd edition. McGraw-Hill, New York.

Lawless, J.F. (2003). *Statistical Models and Methods for Lifetime Data*, 2nd edition. Wiley, Hoboken, NJ.

Lee, S., Wilson, J.R., Crawford, M.M. (1991). Modeling and simulation of a nonhomogeneous Poisson process having cyclic behavior. *Communications in Statistics – Simulation and Computation* 20 (2/3), 777–809.

Leemis, L.M. (1987). Variate generation for the accelerated life and proportional hazards models. *Operations Research* 35 (6), 892–894.

Leemis, L.M. (1991). Nonparametric estimation of the intensity function for a nonhomogeneous Poisson process. *Management Science* 37 (7), 886–900.

Leemis, L.M. (2004). Nonparametric estimation and variate generation for a nonhomogeneous Poisson process from event count data. *IIE Transactions* 36 (12), 1155–1160.

Lewis, P.A.W., Shedler, G.S. (1979). Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly* 26 (3), 403–413.

Marsaglia, G., Olkin, I. (1984). Generating correlation matrices. *SIAM Journal of Scientific and Statistical Computing* 5 (2), 470–475.

Meeker, W.Q., Escobar, L.A. (1998). *Statistical Methods for Reliability Data*. Wiley, New York.

Mehta, M.L. (2004). *Random Matrices*, 3rd edition. Elsevier, London.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E. (1953). Equations of state calculations by fast computing machine. *Journal of Chemical Physics* 21, 1087–1091.

Morris, S.B. (1998). *Magic Tricks, Card Shuffling and Dynamic Computer Memories*. The Mathematical Association of America, Washington, DC.

Nelson, B.L. (2002). *Stochastic Modeling: Analysis and Simulation*. Dover Publications, Mineola, NY.

Nelson, B.L., Ware, P., Cario, M.C., Harris, C.A., Jamison, S.A., Miller, J.O., Steinbugl, J., Yang, J. (1995). Input modeling when simple models fail. In: Alexopoulos, C., Kang, K., Lilegdon, W.R., Goldsman, D. (Eds.), *Proceedings of the 1995 Winter Simulation Conference*. IEEE Press, Piscataway, NJ, pp. 93–100.

Nelson, W.B. (2003). *Recurrent Events Data Analysis for Product Repairs, Disease Recurrences, and Other Applications*. ASA/SIAM, Philadelphia.

Nijenhuis, A., Wilf, H.S. (1978). *Combinatorial Algorithms for Computers and Calculators*, 2nd edition. Academic Press, New York.

Pritsker, A.A.B., Martin, D.L., Reust, J.S., Wagner, M.A., Wilson, J.R., Kuhl, M.E., Allen, M.D., Daily, O.P., Harper, A.M., Edwards, E.B., Bennett, L.E., Roberts, J.P., Burdick, J.F. (1995). Organ transplantation policy evaluation. In: Alexopoulos, C., Kang, K., Lilegdon, W.R., Goldsman, D. (Eds.), *Proceedings of the 1995 Winter Simulation Conference*. IEEE Press, Piscataway, NJ, pp. 1314–1323.

Resnick, S.I. (1992). *Adventures in Stochastic Processes*. Birkhäuser, Boston, MA.

Rigdon, S.E., Basu, A.P. (2000). *Statistical Methods for the Reliability of Repairable Systems*. Wiley, New York.

Ross, S.M. (2003). *Introduction to Probability Models*, 8th edition. Academic Press, New York.

Schoenberg, F.P. (2003). Multidimensional residual analysis of point process models for earthquake occurrences. *Journal of the American Statistical Association* 98 (464), 789–795.

White, K.P. (1999). Simulating a nonstationary Poisson process using bivariate thinning: The case of "typical weekday" arrivals at a consumer electronics store. In: Farrington, P., Nembhard, P.A., Sturrock, H.B., Evans, G.W. (Eds.), *Proceedings of the 1999 Winter Simulation Conference*. IEEE Press, Piscataway, NJ, pp. 458–461.

Whitt, W. (2002). *Stochastic-Process Limits: An Introduction to Stochastic-Process Limits and Their Application to Queues*. Springer-Verlag, New York.

Wilf, H.S. (1989). *Combinatorial Algorithms: An Update*. SIAM, Philadelphia, PA.