Chapter 2
Sampling from Complex Probability Distributions: A Monte Carlo Primer for Engineers

Louis J. M. Aslett

Abstract Models which are constructed to represent the uncertainty arising in engineered systems can often be quite complex to ensure they provide a reasonably faithful reflection of the real-world system. As a result, even computation of simple expectations, event probabilities, variances, or integration over utilities for a decision problem can be analytically intractable. Indeed, such models are often sufficiently high dimensional that even traditional numerical methods perform poorly. However, access to random samples drawn from the probability model under study typically simplifies such problems substantially. The methodologies to generate and use such samples fall under the stable of techniques usually referred to as 'Monte Carlo methods'. This chapter provides a motivation, simple primer introduction to the basics, and sign-posts to further reading and literature on Monte Carlo methods, in a manner that should be accessible to those with an engineering mathematics background. There is deliberately informal mathematical presentation which avoids measure-theoretic formalism. The accompanying lecture can be viewed at https://www.louisaslett.com/Courses/UTOPIAE/.

2.1 Motivation

There is a natural tension when constructing a probabilistic model with the aim of encapsulating the uncertainty in an engineered system: on the one hand, there is a desire to capture every nuance of the system to fully reflect all knowledge about its behaviour; on the other, there is a drive towards parsimony for reasons of interpretability, robustness, and computability. Interpretability and robustness are important goals and should indeed guide a reduction in model complexity, but reducing model complexity purely to enable computability would seem a hinderance, especially if that parsimony impedes answering the research questions at hand since, put simply, 'reality can be complicated' [7]. As such, the methodology of this chapter

L. J. M. Aslett (✉)
Department of Mathematical Sciences, Durham University, Durham, United Kingdom
e-mail: louis.aslett@durham.ac.uk

L. Aslett et al. (eds.), *Uncertainty in Engineering*,
SpringerBriefs in Statistics,
https://doi.org/10.1007/978-3-030-83640-5_2

should not be employed simply to enable an inappropriately complex model, but rather serves to facilitate the use of models which are complex enough when judged by purely subject matter and statistical concerns.

Monte Carlo methods have played a crucial role in a vast array of applications of statistical methodology, from the prediction of future marine species discoveries [29] through to reconstruction of the ancient climate on Earth [16]; from criminal justice offending risk [17] to inferring networks of corporate governance through the financial crash [12]; and from estimating bounds on engineering system survival functions [11] to the assessment of offshore oil production availability [30]. The utility of Monte Carlo in these applications varies substantially, from estimation of confidence intervals and event probabilities, through optimisation methods to full evaluation of Bayesian posterior distributions for parameter inference.

With this breadth of application in mind, we may assume hereinafter that we have a probabilistic model for some engineered system of interest which—after considering all subject matter and statistical concerns—is too complex to be able to compute relevant quantities of interest (be they event probabilities, confidence intervals, posterior distributions, etc.). As a concrete example, if one were to construct a Bayesian model of reliability using ideas introduced in Chap. 1, then our model would comprise some prior distribution over the vector of model parameters, $\pi(\theta)$, together with a generative model for the failure time depending on those parameters, $\pi(t \mid \theta)$. After collecting some lifetime data $\underline{t} = \{t_1, \dots, t_n\}$, the most simple research question of interest may be the posterior expected value of the parameters:

$$\mathbb{E}_\pi[\theta] = \int_\Omega \theta\, \pi(\theta \mid \underline{t})\, d\theta = \frac{1}{c} \int_\Omega \theta\, \pi(\theta) \prod_{i=1}^{n} \pi(t_i \mid \theta)\, d\theta \tag{2.1}$$

where Ω is the space of all possible parameter values and c is a normalising constant.

Indeed, it is traditional in Monte Carlo literature to focus attention on the computation of expectations with respect to some probability density under consideration, which need not necessarily be a Bayesian posterior. That is, given a general probability model $\pi(x)$, $x \in \Omega$, and a functional $f : \Omega \to \mathbb{R}$, interest is typically in:

$$\mathbb{E}_\pi[f(X)] := \int_\Omega f(x)\pi(x)\, dx \tag{2.2}$$

and this is the perspective that will be adopted in this chapter.

We complete our motivation of Monte Carlo in this Section by highlighting the generality of expectations of the form (2.2), followed by a short discussion of standard numerical integration techniques. In Sect. 2.2, the Monte Carlo estimator and its error analysis are recapped and contrasted with numerical integration. The core methods of Monte Carlo simulation are introduced in Sect. 2.3, with pointers to more advanced material in Sect. 2.4. Note that we will in places abuse formal notation where we believe it aids intuitive understanding since the goal of this chapter is to be a basic primer, not a rigorous treatment.[1] A first course in probability and statistics are assumed background.

The accompanying lecture from the UTOPIAE training school can be viewed at https://www.louisaslett.com/Courses/UTOPIAE/.

2.1.1 Generality of Expectations

The formulation in (2.2) may appear rather restrictive to the uninitiated reader. However, considering only expectations of this form does not result in any loss of generality. For example, (re-)defining:

$$\begin{aligned}\pi(x) &:= \frac{1}{c}\pi(x)\prod_{i=1}^{n}\pi(t_i \mid x)\\ f(x) &:= x\end{aligned}$$

means that (2.2) simply becomes the posterior expectation in (2.1). However, one should note that arbitrary statements of probability are also computable as expectations. That is,

$$\mathbb{P}(X < a) = \int_{-\infty}^{a} \pi(x)\,dx = \int_{\Omega} \mathbb{I}_{(-\infty,a]}(x)\pi(x)\,dx = \mathbb{E}_{\pi}[\mathbb{I}_{(-\infty,a]}(X)]$$

where for a general set $E \subseteq \Omega$,

$$\mathbb{I}_E(x) := \begin{cases} 1 & \text{if } x \in E \\ 0 & \text{if } x \notin E \end{cases}$$

That is, to evaluate the probability of an arbitrary event, $\mathbb{P}(X \in E)$, simply set $f(X) := \mathbb{I}_E(X)$ when evaluating (2.2).

2.1.2 Why Consider Monte Carlo?

In some special cases, the integral (2.2) may have an analytical solution and in such situations one should not resort to Monte Carlo or other methods. When there is no known analytical form for the integral, a reader with a general mathematical

[1] For example, we will write '$\mathbb{P}(X = x)$' even where X is continuous to emphasise the link to the density function and will use $\pi(x)$ to reference both a target distribution or prior where the meaning is clear from context. For the more advanced reader there are already many excellent more rigorous treatments in the literature, some of which we reference towards the end.

background may be tempted to reach for a numerical integration method, such as a simple mid-point Riemann integral or a more sophisticated quadrature approach.

Consider the mid-point Riemann integral in the simple 1-dimensional setting. Letting $g(x) := f(x)\pi(x)$, then the expectation would be approximated using n evaluations by:

$$\int_a^b g(x)\,dx \approx \frac{b-a}{n}\sum_{j=1}^n g(x_j), \tag{2.3}$$

where

$$x_j := a + \frac{b-a}{n}\left(j - \frac{1}{2}\right).$$

The absolute error in using (2.3) is bounded [24, Theorem 7.1]:

$$\left|\int_a^b g(x)\,dx - \frac{b-a}{n}\sum_{j=1}^n g(x_j)\right| \leq \frac{(b-a)^3}{24n^2}\max_{a\leq z\leq b}|g''(z)|.$$

Clearly, $\frac{(b-a)^3}{24}\max_{a\leq z\leq b}|g''(z)|$ is fixed by the problem at hand and cannot be altered by the engineer, so we achieve the accuracy we require by controlling n^{-2}—that is, by using a finer grid to compute the integral. As such, we say the error in the mid-point Riemann integral in 1 dimension is $O\left(n^{-2}\right)$—that is, if double the computational effort is expended by computing on a grid of twice as many points ($2n$), then the worst case error is reduced by a factor of 4. This fast reduction in error and an explicit bound on it are very attractive properties.

However, as the dimension of x increases, the Riemann integral's effectiveness diminishes substantially. In general, the error of mid-point Riemann integration in d-dimensions is $O\left(n^{-2/d}\right)$. For example, even in a modest 10-dimensional problem, when the computational effort is doubled the worst case error is only reduced by a factor of ≈ 1.15. Put another way, to halve the worst case error in a 10-dimensional problem requires $\exp\left(\frac{10}{2}\log 2\right) = 32$ times the computational effort. This problem has been coined the 'curse of dimensionality'.

Of course, the Riemann integral is not the best numerical integration method, but even Simpson's rule only improves this to $O\left(n^{-4/d}\right)$. In general Bakhvalov's Theorem bounds all possible quadrature methods by $O(n^{-r/d})$, where r is the number of continuous derivatives of $g(\cdot)$ which exist and are exploited by the quadrature rule [24].

The striking result which motivates the study of Monte Carlo methods is that for a d-dimensional problem, the (mean-square)[2] error is $O\left(n^{-1/2}\right)$. The most important point to note is the absence of d in the order of the error: increasing the computational effort by some fixed amount has the same relative effect on the worst case error regardless of dimension. Of course, the devil in the detail is that the constant

[2] Note that randomised simulation methods such as Monte Carlo typically report mean-square error rather than absolute error bounds.

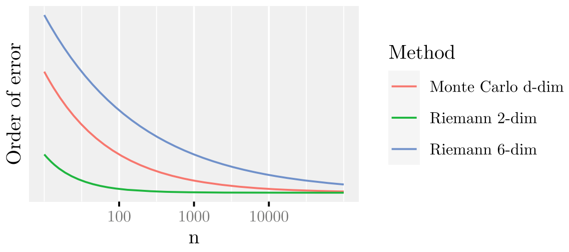

Fig. 2.1 The order of error reduction—that is, only the leading $O\left(n^{f(d)}\right)$ term—is plotted against different computational effort n. Note that all these curves would be multiplied by a different (fixed) problem dependent constant

factor which we are ignoring in that statement almost certainly has some dimension dependence, but this is true for quadrature methods too. Figure 2.1 illustrates the differences.

Consequently, Monte Carlo methods are well suited to address the problem of analysing complex probabilistic models of engineered systems, since this is precisely a setting where the parameter dimension is likely to be large.

2.2 Monte Carlo Estimators

The standard Monte Carlo estimator of the integral (2.2) is

$$\mu \triangleq \int_{\Omega} f(x)\pi(x)\,dx \approx \frac{1}{n}\sum_{j=1}^{n} f(x_j) \triangleq \hat{\mu}, \tag{2.4}$$

where $x_j \sim \pi(\cdot)$. In other words, the problem of integration is transformed instead into the problem of drawing random samples x_j distributed according to the probability density $\pi(\cdot)$. Importantly, this estimator is unbiased, that is, $\mathbb{E}[\hat{\mu}] = \mu$.

If the samples x_j are independently and identically distributed (iid) according to $\pi(\cdot)$, then the root mean-square error of the estimator $\hat{\mu}$ is

$$\text{RMSE} := \sqrt{\mathbb{E}_{\pi}\left[\left(\int f(x)\pi(x)\,dx - \frac{1}{n}\sum_{j=1}^{n} f(x_j)\right)^2\right]} = \frac{\sigma}{\sqrt{n}},$$

where $\sigma^2 = \text{Var}_\pi\,(f(X))$. Again, part of this error is (mostly) inherent to the problem[3]—σ in this case—so that we achieve desired accuracy by controlling $n^{-1/2}$. There are at least three very attractive conclusions we can draw from this form:

1. as mentioned already, the relative error reduction achieved by additional computational effort is independent of dimension;
2. there is no explicit dependence on how smooth the functional, $f(\cdot)$, or probability density, $\pi(\cdot)$, are (though these may influence σ);
3. in contrast to quadrature methods, an estimate of the error can be computed from the work already done to compute the integral, by computing the empirical standard deviation of the functional of the samples drawn from $\pi(\cdot)$.

Although an absolute error is not available for a randomised method like this, a simple application of Chebyshev's inequality does provide a probabilistic bound on the absolute error exceeding a desired tolerance:

$$\mathbb{P}(|\hat{\mu} - \mu| \geq \varepsilon) \leq \frac{\mathbb{E}_\pi[(\hat{\mu} - \mu)^2]}{\varepsilon^2} = \frac{\sigma^2}{n\varepsilon^2}.$$

Indeed, it is also possible to invoke the iid Central Limit Theorem so that asymptotically,

$$\mathbb{P}\left(\frac{\hat{\mu} - \mu}{\sigma n^{-1/2}} \leq z\right) \xrightarrow{n\to\infty} \Phi(z),$$

where $\Phi(z)$ denotes the standard Normal cumulative distribution function (CDF). This enables the formation of confidence intervals for μ based on large n samples.

The discussion to date has tacitly assumed that simulating from arbitrary probability distributions $\pi(\cdot)$ is possible and relatively efficient. In fact, most Monte Carlo research is devoted to this effort since, as touched on above, there is rich and well-established theory when such samples are available. Therefore, for the remainder of this chapter, our attention turns away from discussion of the integrals which are of primary interest and focuses on the problem of simulating from arbitrary probability distributions $\pi(\cdot)$. Once these samples are available, the results above can be used to analyse the resulting estimators.

2.3 Simple Monte Carlo Sampling Methods

In this section we introduce some simple Monte Carlo methods which enable sampling from a wide array of probability distributions. Note that understanding these simple methods is crucial as they are extensively used as building blocks of more sophisticated sampling methodology.

[3] There are advanced Monte Carlo methods which can reduce this variance, but this is beyond the scope of this chapter. See for example [24, Chap. 8].

Almost all Monte Carlo procedures start from the assumption that we have available an unlimited stream of iid uniformly distributed values, typically on the interval $[0, 1] \subset \mathbb{R}$. How to generate such an iid stream is beyond the scope of this introductory chapter, but the interested reader may consult [13, Chaps. 1–3] and [21]. Arguably the current gold standard algorithm remains that in [22]. Typically, the average user of Monte Carlo need not worry about such issues and may rely on the high quality generators built into software such as R [26].

Thus the objective hereinafter is to study how to convert a stream $u_i \sim \text{Unif}(0, 1)$ into a stream $x_j \sim \pi(\cdot)$, where x_j is generated by some algorithm depending on the stream of u_i. In more advanced methods (see MCMC), x_j may also depend on x_{j-1} or even $x_1, \dots, x_{j-1}$.

2.3.1 Inverse Sampling

Arguably the simplest example of generating non-uniform random variates is inverse sampling, which typically applies only to 1-dimensional probability distributions (though higher dimensional extensions have been studied). Let $F(x) := \mathbb{P}(X \le x)$ be the cumulative distribution function (CDF) for the target probability density function $\pi(\cdot)$. Then, inverse sampling requires the inverse of the cdf, $F^{-1}(\cdot)$, which is then applied to a uniform random draw. Precisely, see Algorithm 2.1.

Algorithm 2.1 Inverse sampling algorithm

```
1: procedure INVERSE SAMPLING($F^{-1}(\cdot)$)   ▷ Generate random sample from distribution with
                                                  inverse CDF $F^{-1}(\cdot)$
2:     $u \sim \text{Unif}(0, 1)$
3:     $x \leftarrow F^{-1}(u)$
4:     return $x$
5: end procedure
```

To prove that the sample returned by Algorithm 2.1 is distributed according to $\pi(\cdot)$ is straight-forward. We do so by computing the CDF, $\mathbb{P}(X \le x)$, of the X generated by this algorithm and show that this agrees with the CDF of $\pi(\cdot)$. The first step substitutes $X = F^{-1}(U)$, where $U \sim \text{Unif}(0, 1)$, as per the algorithm:

$$\begin{aligned}
\mathbb{P}(\, X \le x \,) &= \mathbb{P}(\, F^{-1}(U) \le x \,) \\
&= \mathbb{P}(\, F(F^{-1}(U)) \le F(x) \,) && \text{applying } F(\cdot) \text{ to both sides} \\
&= \mathbb{P}(\, U \le F(x) \,) && \text{Uniform CDF } \mathbb{P}(U \le u) = u \\
&= F(x).
\end{aligned}$$

Note that applying $F(\cdot)$ to both sides in the second line is valid, since the cumulative distribution function is a non-decreasing function by definition.

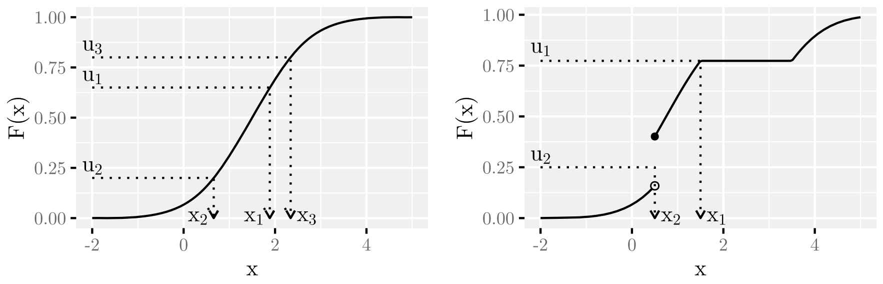

Fig. 2.2 Inverse sampling for both a continuous distribution function (left) and one containing jump discontinuities and regions of zero probability (right). Uniform random draws u are sampled and inverted through the distribution function in the obvious way (left), or by taking the infimum over values of x such that $F(x) \geq u$ (right). In the right illustration, the hypothetical ('Hypothetical' since strictly speaking this is an event of probability zero) u_1 coincides with the value at which $F(x)$ is constant and u_2 lies within the jump discontinuity

One subtlety to be aware of is that for discrete distributions or continuous distributions with jump discontinuities or areas of no support, we must define:

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}, \ \forall\, u \in [0, 1].$$

It may be tempting when $F^{-1}(\cdot)$ is not available to use a numerical solver to solve $F(x) = u$ in place of line 3 in Algorithm 2.1. However, caution is required since this can result in bias [10, p. 31]. The procedure of inverse sampling is illustrated in Fig. 2.2.

Notice that this is univariate, yet earlier we saw that numerical integration will give better error bounds than Monte Carlo for low dimensional problems—as such one may choose not to use inverse sampling to actually evaluate univariate expectations. However, we often need a set of random draws from non-uniform univariate distributions which feed into a broader Monte Carlo algorithm, which is itself sampling in higher dimensions: in such situations inverse sampling is very useful. Indeed, if you use the `rnorm` function in R [26], it has used inverse sampling to generate random draws from the Normal distribution since 2003 (see `/src/nmath/snorm.c` lines 265–270), prior to that using [18] since at least v0.62 in 1998.

A final comment: inverse sampling is a special case of general transformation sampling. If one can generate samples from one distribution, there may be an appropriate transformation to turn these into samples from another distribution that may be more tractable or faster than inverse sampling. For further details, see for example [24, Chap. 4.6].

2.3.1.1 Example

In order to use inversion sampling for a Weibull distribution with shape k and scale σ, $X \sim \text{Weibull}(k, \sigma)$, we note that

$$\pi(x) = \frac{k}{\sigma}\left(\frac{x}{\sigma}\right)^{k-1} e^{-\left(\frac{x}{\sigma}\right)^k}, \quad x \in [0, \infty), \sigma > 0, k > 0$$
$$F(x) = 1 - \exp\left\{-\left(\frac{x}{\sigma}\right)^k\right\}.$$

To find $F^{-1}(u)$, set $1 - \exp\left\{-\left(\frac{x}{\sigma}\right)^k\right\} = u$ and solve for x:

$$\implies x = F^{-1}(u) = \sigma\left(-\log(1-u)\right)^{1/k} \sim \pi(\cdot). \tag{2.5}$$

In order to generate samples from the Weibull we, therefore, take values, u, from a Uniform random number stream and transform them using (2.5).

2.3.2 Rejection Sampling

Our first higher dimensional method is an elegant algorithm, which actually crops up in more advanced guises at the cutting edge of modern Monte Carlo methods (e.g. [9, 25]). Here, the goal is to find another distribution, say $\tilde{\pi}(\cdot)$, which is easier to sample from (perhaps even using inverse sampling) and where we can construct a bound on the density function:

$$\pi(x) \le c\tilde{\pi}(x) \quad \forall x \in \Omega, \tag{2.6}$$

where $c < \infty$ and where π and $\tilde{\pi}$ need not be normalised probability densities. We call $\tilde{\pi}(\cdot)$ the 'proposal' density, since samples will be drawn from this and then exactly the correct proportion of them retained in order to end up with a stream of samples from $\pi(\cdot)$. The full procedure is detailed in Algorithm 2.2.

We will proceed based on the assumption that $\pi(\cdot)$ and $\tilde{\pi}(\cdot)$ are normalised densities. However, note that the algorithm is also valid for un-normalised densities, so long as there still exists a c satisfying (2.6) for the un-normalised densities.

The efficiency of the algorithm hinges entirely on the value of c, so that it should be chosen as small as possible. This is because, letting A be the random variable for acceptance of a proposed sample X, the acceptance probability is (abusing notation to aid intuition):

Algorithm 2.2 Rejection sampling algorithm

```
1: procedure REJECTION SAMPLING($\pi(\cdot)$, $\tilde{\pi}(\cdot)$, $c$)   ▷ Generate random sample from distribution with unnormalised density $\pi(\cdot)$
2:     $a \leftarrow$ FALSE
3:     while $a =$ FALSE do                                  ▷ Repeat until acceptance
4:         $u \sim \text{Unif}(0, 1)$
5:         $x \sim \tilde{\pi}(\cdot)$                                    ▷ Propose a possible sample
6:         if $u \le \frac{\pi(x)}{c\tilde{\pi}(x)}$ then                          ▷ Accept or reject proposal?
7:             $a \leftarrow$ TRUE
8:         end if
9:     end while
10:    return $x$
11: end procedure
```

$$
\begin{aligned}
\mathbb{P}(A = 1) &= \int_\Omega \underbrace{\mathbb{P}(A = 1 \mid X = x)}_{\text{Prob line 6 of Alg 2 gives TRUE.}} \quad \underbrace{\mathbb{P}(X = x)}_{\text{Proposal density } \tilde{\pi}.} \quad dx \\
&= \int_\Omega \underbrace{\mathbb{P}\left(U \le \frac{\pi(x)}{c\tilde{\pi}(x)}\right)}_{\text{Uniform CDF, } \mathbb{P}(U \le u) = F(u) = u.} \quad \tilde{\pi}(x)\, dx \\
&= \int_\Omega \frac{\pi(x)}{c\tilde{\pi}(x)}\, \tilde{\pi}(x)\, dx \\
&= \frac{1}{c} \int_\Omega \pi(x)\, dx \\
&= \frac{1}{c},
\end{aligned}
\tag{2.7}
$$

where Ω is the support of $\tilde{\pi}(\cdot)$, $\tilde{\pi}(x) > 0,\ \forall\, x \in \Omega$. The final line follows because the integral of a density over the whole space is 1.

Hence, the number of iterations of the loop on lines 3–9 in Algorithm 2.2 which must be performed to return a single sample from $\pi(\cdot)$ is Geometrically distributed with parameter $\frac{1}{c}$. Therefore, the expected number of random number generations and function evaluations which must be performed is $2c$ per sample from $\pi(\cdot)$.

To see that Algorithm 2.2 does indeed give a sample from $\pi(\cdot)$, we note that the samples returned are only those which are accepted, so we condition on this event:

$$
\begin{aligned}
\mathbb{P}(X \in E \mid A = 1) &= \frac{\mathbb{P}(A = 1 \mid X \in E)\, \mathbb{P}(X \in E)}{\mathbb{P}(A = 1)} \qquad \forall\, E \in \mathcal{B} \\
&= \frac{\int_E \frac{\pi(x)}{c\tilde{\pi}(x)}\, \tilde{\pi}(x)\, dx}{\frac{1}{c}} \\
&= \int_E \pi(x)\, dx.
\end{aligned}
$$

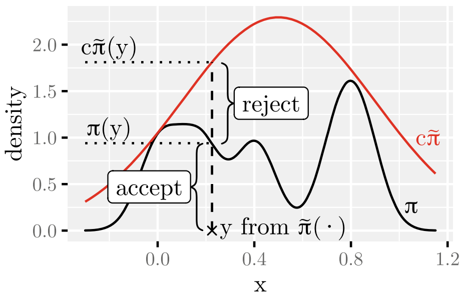

Fig. 2.3 Geometric interpretation of rejection sampling. First y is sampled from $\tilde{\pi}$ and then a uniform is sampled along the vertical dashed line at that location (i.e. between 0 and $c\tilde{\pi}(y)$). If the uniform sample falls below π then we accept and otherwise we reject. It is therefore clear that the closer $c\tilde{\pi}$ 'hugs' π the more efficient the rejection sampler

The last line is the probability of event E under the distribution with density $\pi(\cdot)$, as required.

There is a nice geometric interpretation of the rejection sampling algorithm which aids greatly with intuition. Notice that the condition in line 6 can be rewritten $uc\tilde{\pi}(x) \leq \pi(x)$. This means $uc\tilde{\pi}(x)$ is a uniform random number in the interval $[0, c\tilde{\pi}(x)]$, so that we can view rejection sampling as first drawing a value from $\tilde{\pi}(x)$, then moving it up to a uniformly distributed height under the curve $c\tilde{\pi}(x)$. The consequence of this is that we are effectively sampling uniformly points under the curve $c\tilde{\pi}(x)$ and accepting those that fall under the curve $\pi(x)$, as depicted in Fig. 2.3.

Care is required with rejection sampling in high dimensions because it is quite easy for the acceptance probability to become so small as to make the technique impractical. We will see that, as well as how to implement rejection sampling, in the following example.

2.3.2.1 Example

Consider the problem of sampling from a zero mean d-dimensional multivariate Normal distribution, having density:

$$\pi(\mathbf{x}) = (2\pi)^{-d/2} \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\Sigma^{-1}\mathbf{x}\right), \qquad \mathbf{x} \in \mathbb{R}^d,$$

where Σ is a $d \times d$ symmetric positive semi-definite covariance matrix. It is comparatively easy to sample univariate Normal random variables (e.g. using inverse sampling in R [26] as mentioned earlier, or via a transformation type approach like [3]). Thus we could consider using a multivariate Normal with diagonal covariance,

$\sigma^2 I$, as a proposal, because this simply requires sampling d univariate Normal random variables.

This would mean we need to determine $c < \infty$ such that

$$c \det(\sigma^2 I)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\sigma^{-2}I\mathbf{x}\right) \geq \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\Sigma^{-1}\mathbf{x}\right) \quad \forall\, \mathbf{x} \in \mathbb{R}^d \tag{2.8}$$

Σ is symmetric, so it has eigendecomposition $\Sigma = Q\Lambda Q^{\mathrm{T}} \implies \Sigma^{-1} = Q\Lambda^{-1}Q^{\mathrm{T}}$, where Q is an orthogonal matrix and Λ is a diagonal matrix with entries consisting of the eigenvalues $\lambda_1, \ldots, \lambda_d$. The orthogonal transformation $\mathbf{y} = Q^{\mathrm{T}}\mathbf{x}$ also spans $\mathbb{R}^d$, so that (2.8) $\iff$

$$\begin{aligned} c\sigma^{-d} \exp\left(-\frac{1}{2}\mathbf{y}^{\mathrm{T}}\sigma^{-2}I\mathbf{y}\right) &\geq \left(\prod_{i=1}^{d} \lambda_i\right)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{y}^{\mathrm{T}}\Lambda^{-1}\mathbf{y}\right), \quad \forall\, \mathbf{y} \in \mathbb{R}^d \\ \iff 2\log c &\geq \sum_{i=1}^{d}(\sigma^{-2} - \lambda_i^{-1})y_i^2 + 2d\log\sigma - \sum_{i=1}^{d}\log\lambda_i. \end{aligned}$$

If $\sigma^{-2} < \lambda_i^{-1}$ for any i, then the right-hand side cannot be bounded above (since the inequality must hold $\forall\ y_i \in \mathbb{R}$), so we must have $\max_i \lambda_i < \sigma^2$ and then clearly c is minimised for $\sigma^2 = \max_i \lambda_i$. Since every term in the first sum of the right-hand side is necessarily negative, the right-hand side is maximal for $y_i = 0\ \forall\, i$, so that the optimal c is

$$c = \left(\max_i \lambda_i\right)^{d/2} \left(\prod_{i=1}^{d} \lambda_i\right)^{-1/2},$$

when $\sigma^2 = \max_i \lambda_i$.

In summary, there is a constraint on our proposal $\tilde{\pi}(\cdot)$ when it is an uncorrelated multivariate Normal density, or else it cannot bound $\pi(\cdot)$. Moreover, we can explicitly compute the optimal proposal variance, σ^2, to give us the highest possible acceptance rate.

To make this example concrete, consider rejection sampling in this setting where

$$\Sigma = \begin{pmatrix} 1 & 0.9 & \cdots & 0.9 \\ 0.9 & 1 & \cdots & 0.9 \\ \vdots & \vdots & \ddots & \vdots \\ 0.9 & 0.9 & \cdots & 1. \end{pmatrix}$$

Note that Σ can be written as $0.1I + B$, where B is a matrix with 0.9 in every element. The rank of B is 1, so it has a single non-zero eigenvalue which must therefore equal $\operatorname{tr}(B) = 0.9d$, and the eigenvalues of $0.1I$ are all 0.1. Further-

Table 2.1 Optimal proposal variance and acceptance probability for rejection sampling a correlated multivariate Normal distribution using an uncorrelated multivariate Normal proposal

d	$\sigma^2 = \max_i \lambda_i$	Acceptance probability $\frac{1}{c}$
1	1	1
2	1.9	0.229
3	2.8	0.036
4	3.7	0.004
5	4.6	4.45×10^{-5}
$\vdots$	$\vdots$	$\vdots$
10	9.1	1.53×10^{-9}

more, $0.1I$ and B commute, therefore the eigenvalues of Σ are the sum of these eigenvalues: that is, $\lambda_1 = 0.9d + 0.1$ and $\lambda_i = 0.1 \; \forall \; i \neq 1$. As the dimension of $\pi(\cdot)$ increases, the spectral gap increases linearly and thus c grows very fast: $c = (10\lambda_1)^{(d-1)/2} = (9d+1)^{(d-1)/2}$. Indeed, this is faster than exponential and faster than factorial growth! Consequently, for growing dimension, the acceptance probability falls super-exponentially fast—not a desirable property. See Table 2.1 for some example values.

A whimsical observation to emphasise the problem: a modern laptop can produce roughly 15 million univariate Normal samples per second and the universe is estimated to be 4.32×10^{17} seconds old. Ignoring the time to evaluate the uniform draw u or acceptance/rejection, this means the expected number of samples that would be generated by Algorithm 2.2 for this multivariate Normal problem in d-dimensions—if run for as long as the universe has existed—would be

$$\frac{1.5 \times 10^7 \times 4.32 \times 10^{17}}{d(9d+1)^{(d-1)/2}}.$$

Consequently, even knowing the exactly optimal choice for σ^2 in our proposal, this would only be expected to render 5 samples for a 21-dimensional multivariate Normal with the innocuous looking Σ given above—rejection sampling in high dimensions can be problematic!

2.3.3 *Importance Sampling*

The final core standard Monte Carlo method we cover in this primer also starts from the perspective of having a proposal density $\tilde{\pi}(\cdot)$, though we no longer require it to be able to bound $\pi(\cdot)$. Importance sampling then dispenses with the notion of directly generating iid samples from $\pi(\cdot)$ and focuses on their use: in computing expectations using those samples in (2.4). Consequently, importance sampling weights the samples from $\tilde{\pi}(\cdot)$ in precisely the proportion that ensures these weighted samples produce expectations which are concordant with expectations under $\pi(\cdot)$ when used in (2.4). This is laid out precisely in Algorithm 2.3.

Algorithm 2.3 Importance sampling algorithm

```
1: procedure IMPORTANCE SAMPLING($\pi(\cdot), \tilde{\pi}(\cdot), c$)    ▷ Generate random sample from distribution with un-normalised density $\pi(\cdot)$
2:     $x \sim \tilde{\pi}(\cdot)$    ▷ Propose sample
3:     $w \leftarrow \frac{\pi(x)}{\tilde{\pi}(x)}$
4:     return $(x, w)$.
5: end procedure
```

To see that this weighting has the desired effect, consider the expectation which is our objective. We first consider the situation where both π and $\tilde{\pi}$ are normalised:

$$\begin{aligned}
\mathbb{E}_\pi[f(X)] &= \int_\Omega f(x)\pi(x)\,dx \\
&= \int_\Omega f(x)\frac{\tilde{\pi}(x)}{\tilde{\pi}(x)}\pi(x)\,dx && \text{multiply and divide by } \tilde{\pi}(x) \\
&= \int_\Omega \left(f(x)\frac{\pi(x)}{\tilde{\pi}(x)}\right)\tilde{\pi}(x)\,dx \\
&= \mathbb{E}_{\tilde{\pi}}\left[\frac{f(X)\pi(X)}{\tilde{\pi}(X)}\right].
\end{aligned}$$

That is, we use samples directly from $\tilde{\pi}(\cdot)$, and instead adjust (2.4) to target the expectation of the same functional under $\pi(\cdot)$.

$$\mu \triangleq \int_\Omega f(x)\pi(x)\,dx \approx \frac{1}{n}\sum_{j=1}^n f(x_j)\underbrace{\frac{\pi(x_j)}{\tilde{\pi}(x_j)}}_{=w_j} = \frac{1}{n}\sum_{j=1}^n f(x_j)w_j \triangleq \hat{\mu}, \tag{2.9}$$

where now $x_j \sim \tilde{\pi}(\cdot)$.

Some care is required, because although this estimator remains unbiased, the variance is no longer going to be the same as the usual Monte Carlo variance where $x_j \sim \pi(\cdot)$. Indeed, now

$$\mathrm{Var}(\hat{\mu}) = \frac{\sigma_{\tilde{\pi}}^2}{n} \text{ where } \sigma_{\tilde{\pi}}^2 = \int_\Omega \frac{(f(x)\pi(x) - \mu\tilde{\pi}(x))^2}{\tilde{\pi}(x)}\,dx,$$

which can be empirically estimated from the importance samples using,

$$\hat{\sigma}^2_{\tilde{\pi}} = \frac{1}{n} \sum_{j=1}^{n} \left(f(x_j) w_j - \hat{\mu} \right)^2 . \tag{2.10}$$

As such, $\sigma^2_{\tilde{\pi}}$ (or its empirical estimate $\hat{\sigma}^2_{\tilde{\pi}}$) provide a guide to when we have a 'good' importance sampling algorithm, since with $\tilde{\pi}(\cdot)$ fixed the only option to improve the estimate is to increase the sample size n.

Indeed, it can be shown [15] that the theoretically optimal proposal distribution which minimises the estimator variance is

$$\tilde{\pi}(x)_{\text{opt}} = \frac{|f(x)|\pi(x)}{\int_\Omega |f(x)|\pi(x)\,dx}.$$

In particular, note that this implies that importance sampling can achieve *super-efficiency* whereby it results in lower variance even than sampling directly from $\pi(\cdot)$ when $f(x) \neq x$. Specifically, if $f(x) \geq 0 \; \forall x$ then this proposal results in a zero-variance estimator! Of course, in practice we cannot usually sample from and evaluate this optimal proposal, since it is at least as difficult as the original problem we were attempting to solve. However, even though these optimal proposals are often unusable, they provide guidance towards the form of a good proposal for any given importance sampling problem.

2.3.3.1 Self-normalising Weights

The option to use rejection sampling with un-normalised densities is very helpful (e.g. in Bayesian settings where the normalising constant is often unknown). We can retain this advantage with importance sampling by using *self-normalising weights*. The algorithm to generate the weights remains as in Algorithm 2.3, but the computation of the estimator in (2.9) changes. The self-normalised version, rather than dividing by n, uses the sum of the weights,

$$\hat{\mu}^\star \triangleq \frac{\sum_{j=1}^{n} f(x_j) w_j}{\sum_{j=1}^{n} w_j},$$

thereby ensuring cancellation of the unknown normalising constant from the target and/or proposal distributions in the weights.

However, it is important to note that this estimator is no longer unbiased, though asymptotically it is. Additionally, the variance of this estimator is more complicated having only approximate form. An approximate estimate can be computed using,

$$\mathrm{Var}(\hat{\mu}^\star) \approx \frac{\hat{\sigma}_{\tilde{\pi}}^{\star 2}}{n} \text{ where } \hat{\sigma}_{\tilde{\pi}}^{\star 2} = \sum_{j=1}^{n} w_j^{\star 2} \left(f(x_j) - \hat{\mu}^\star \right)^2$$
$$\text{and } w_j^\star = \frac{w_j}{\sum_{i=1}^n w_i}.$$

Finally, the theoretically optimal (but usually unusable) proposal in the self-normalised weight case is

$$\tilde{\pi}(x)_{\text{opt}} \propto |f(x) - \mu| \pi(x).$$

In both regular and self-normalised weight settings, one can then compute appropriate confidence intervals in the usual manner.

2.3.3.2 Diagnostics

Additional care is required in the application of importance sampling when compared to using iid samples from the distribution of interest. In particular, because importance sampling uses a weighted collection of samples, it is not uncommon to be in a situation where a small number of samples with large weight dominate the estimate, so that simply having many importance samples does not equate to good estimation overall.

A common diagnostic for potential weight imbalance is derived by equating the variance of a weighted importance sampling approach to the standard iid Monte Carlo variance for an average computed using a fixed but unknown sample size n_e. Upon simple algebraic rearrangement one may then solve for n_e, the so-called *effective sample size*. This informally corresponds to the size of iid Monte Carlo sample one would expect to need to attain the same variance achieved via this importance sample, so that a low value indicates poor weight behaviour (since that corresponds to few iid samples).

$$\mathrm{Var}\left(\frac{\sum_{i=1}^n f(x_i) w_i}{\sum_{i=1}^n w_i} \right) = \frac{\sigma^2}{n_e} \quad \Longrightarrow \quad n_e = \frac{n \bar{w}^2}{\overline{w^2}},$$

where

$$\bar{w}^2 = \left(\frac{1}{n} \sum_{j=1}^n w_j \right)^2 \quad \text{and} \quad \overline{w^2} = \frac{1}{n} \sum_{j=1}^n w_j^2.$$

The reason to use such a diagnostic and not simply rely on the empirical variance estimates above is that they are themselves based on the sampling procedure and therefore may be poor estimates too.

Finally, it is critical to note that although small n_e does diagnose a problem with importance sampling, it is not necessarily true that large n_e means everything is ok:

it is, for example, entirely possible that the sampler has missed whole regions of high probability.

2.3.3.3 Example

Consider the toy problem of computing $\mathbb{P}(X > 3.09)$ when X is a univariate standard Normal random variable. The R-language [26] computes the distribution function of the standard Normal to at least 18 significant digits using a rational Chebyshev approximation [6] (see `/src/nmath/pnorm.c`) and we know the true answer to be 0.001001 (4 sf). However, if the reader suspends disbelief and imagines that we cannot accurately compute the distribution function, but can only compute the Normal density and draw random realisations from it, then evaluation of the above probability might be approximated using Monte Carlo methods instead (given the unbounded support and extreme tail location this may be preferred to numerical integration).

Since we are assuming the ability to generate random realisations from the Normal distribution, a standard Monte Carlo approach would draw many samples $x_i \sim \mathrm{N}(0, 1)$ and compute

$$\mathbb{P}(X > 3.09) = \mathbb{E}\left[\mathbb{I}_{[3.09,\infty)}(X)\right] = \frac{1}{n}\sum_{j=1}^{n} \mathbb{I}_{[3.09,\infty)}(x_j)$$

However, this will require many samples to achieve an accurate estimate of this tail probability.

In contrast, still only using simulations from a Normal distribution, we may elect to use importance sampling with a proposal $\mathrm{N}(m, 1)$ for some choice m. We know the fully normalised density of a Normal distribution and therefore will be using the estimator (2.9) with associated single sample variance which can be approximated using (2.10). Therefore, to select m, we perform a small grid search over possible proposals, computing $\hat{\sigma}^2_{\tilde{\pi}}$ each time, to find a good choice. This results in Fig. 2.4, showing that a proposal $\mathrm{N}(3.25, 1)$ is a good choice.

A further final run of $n = 100{,}000$ samples renders an estimate $\hat{\mu} = 0.001002$ (4 sf). The same pseudo-random number stream using standard Monte Carlo renders an estimate 0.001140 (4 sf), which is a relative error $163\times$ larger than the importance sampling estimate. To demonstrate this is not a 'fluke' result, we continue to repeat both importance sampling and standard Monte Carlo estimation with runs of size $n = 100{,}000$ and plot the density of estimates of $\mathbb{P}(X > 3.09)$ in Fig. 2.5.

Note that Fig. 2.5 demonstrates how much more accurate importance sampling is for the same sample size $n = 100{,}000$ when computing this event probability compared to standard Monte Carlo. One may reasonably object that we have ignored the 25 pilot runs of $n = 100{,}000$ importance samples used to select $m = 3.25$, so that the total computational effort expended on importance sampling was at least $26\times$ that of standard Monte Carlo. However, it is a simple calculation to determine

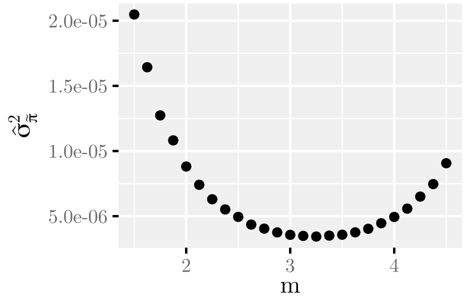

Fig. 2.4 The estimate of $\hat{\sigma}^2_{\tilde{\pi}}$ using (2.10) for 25 different values of m in the Normal proposal $\mathrm{N}(m, 1)$ used in an importance sampling estimator of $\mathbb{P}(X > 3.09)$, where $X \sim \mathrm{N}(0, 1)$. Each estimate of $\hat{\sigma}^2_{\tilde{\pi}}$ is based on $n = 100,000$ samples. m varies on an equally spaced grid from 1.5 to 4.5. The minimum is at $m = 3.25$

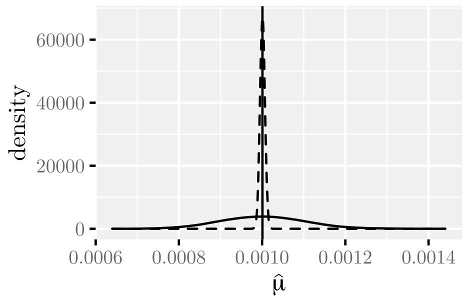

Fig. 2.5 A total of 10, 000 runs of both importance sampling and standard Monte Carlo, each of size $n = 100{,}000$. Each run was used to compute the estimate of $\mathbb{P}(X > 3.09)$ where $X \sim \mathrm{N}(0, 1)$ and a kernel density plot of these estimates produced (importance sampling = dashed, standard Monte Carlo = solid). The vertical line is the ground truth computed using `pnorm(3.09, lower.tail = FALSE)`. The same pseudo-random number stream was used for each method to ensure a fair comparison

that based on the standard deviation of the samples used to generate Fig. 2.5 and the $\sqrt{n}$ convergence of Monte Carlo, that it would require a standard Monte Carlo sample of size $n = 295 \times 100{,}000$ to achieve the same accuracy profile as importance sampling. Therefore, even accounting for the pilot computational effort to select a proposal distribution, there is a substantial benefit to using importance sampling.

2.4 Further Reading

A textbook length introduction with a solid emphasis on implementation details in R can be found in [28]. The same authors have a more advanced textbook going into the theoretical aspects more deeply [27]. Both these books also introduce Markov

Chain Monte Carlo methods, which are often used in practice in high dimensional problems. A nice tutorial paper introduction to MCMC is [1] and [4] is an excellent collection of chapters on the topic.

A classic Monte Carlo text is [10], which is now freely (and legally) available online and contains many results not easily found elsewhere.

Although standard Monte Carlo and Markov Chain Monte Carlo arguably represent the mainstay of most practical uses of Monte Carlo, there are an array of advanced methods which are particularly well suited to different settings. Some excellent review texts as jumping off points to explore some of these include [8] (Sequential Monte Carlo), [19, 20] (Approximate Bayesian Computation), [14] (Multi-Level Monte Carlo), and [2] (MLMC in engineering reliability).

Excellent software choices for practically performing inference in complex models via sampling methods includes Stan [5] and Birch [23].

Dedication

This chapter is dedicated to the memory of Brett Houlding (1982–2019). The tragic news of Brett's passing was received while I was in the act of writing this chapter. He will be sorely missed.

References

1. C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for Machine Learning. *Machine Learning*, 50:5–43, 2003.
2. L. J. M. Aslett, T. Nagapetyan, and S. J. Vollmer. Multilevel Monte Carlo for Reliability Theory. *Reliability Engineering & System Safety*, 165:188–196, 2017.
3. G. E. P. Box and M. E. Muller. A note on the generation of Normal deviates. *Annals of Mathematical Statistics*, 29(2):610–611, 1958.
4. S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.
5. B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017. https://mc-stan.org/.
6. W. J. Cody. Rational Chebyshev approximations for the error function. *Mathematics of Computation*, 23(107):631–637, 1969.
7. D. R. Cox. Applied statistics: a review. *The Annals of Applied Statistics*, 1(1):1–16, 2007.
8. C. Dai, J. Heng, P. E. Jacob, and N. Whiteley. An invitation to sequential Monte Carlo samplers, 2020. arXiv:2007.11936 [stat.CO].
9. H. Dai, M. Pollock, and G. O. Roberts. Monte Carlo Fusion. *arXiv preprint* arXiv:1901.00139, 2019.
10. L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986. Available at http://luc.devroye.org/rnbookindex.html.
11. G. Feng, E. Patelli, M. Beer, and F. P. A. Coolen. Imprecise system reliability and component importance based on survival signature. *Reliability Engineering & System Safety*, 150:116–125, 2016.

12. N. Friel, R. Rastelli, J. Wyse, and A. E. Raftery. Interlocking directorates in Irish companies using a latent space model for bipartite networks. *Proceedings of the National Academy of Sciences*, 113(24):6629–6634, 2016.
13. J. E. Gentle. *Random number generation and Monte Carlo methods*. Springer Science & Business Media, 2006.
14. M. B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.
15. G. Goertzel. *Quota sampling and importance functions in stochastic solution of particle problems*, volume 2793. US Atomic Energy Commission, Technical Information Division, 1950.
16. J. Haslett, M. Whiley, S. Bhattacharya, M. Salter-Townshend, S. P. Wilson, J. R. M. Allen, B. Huntley, and F. J. G. Mitchell. Bayesian palaeoclimate reconstruction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 169(3):395–438, 2006.
17. B. Houlding and S. P. Wilson. Considerations on the UK re-arrest hazard data analysis. *Law, Probability & Risk*, 10(4):303–327, 2011.
18. A. J. Kinderman and J. G. Ramage. Computer generation of Normal random variables. *Journal of the American Statistical Association*, 71(356):893–896, 1976.
19. J. M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.
20. P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *PNAS*, 100(26):15324–15328, 2003.
21. G. Marsaglia. Random number generators. *Journal of Modern Applied Statistical Methods*, 2(1):2, 2003.
22. M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.
23. L. M. Murray and T. B. Schön. Automated learning with a probabilistic programming language: Birch. *Annual Reviews in Control*, 46:29–43, 2018. https://www.birch.sh/.
24. A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.
25. M. Pollock, A. M. Johansen, and G. O. Roberts. On the exact and ε-strong simulation of (jump) diffusions. *Bernoulli*, 22(2):794–856, 2016.
26. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.
27. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag New York, 2004.
28. C. P. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Springer, New York, NY, 2010.
29. S. P. Wilson and M. J. Costello. Predicting future discoveries of European marine species by using a non-homogeneous renewal process. *Journal of the Royal Statistical Society: Series C*, 54(5):897–918, 2005.
30. E. Zio, P. Baraldi, and E. Patelli. Assessment of the availability of an offshore installation by Monte Carlo simulation. *International Journal of Pressure Vessels and Piping*, 83(4):312–320, 2006.