

Simulation



11.1 Introduction

Let $\mathbf{X} = (X_1, \dots, X_n)$ denote a random vector having a given density function $f(x_1, \dots, x_n)$ and suppose we are interested in computing

$$E[g(\mathbf{X})] = \iint \cdots \int g(x_1, \dots, x_n) f(x_1, \dots, x_n) dx_1 dx_2 \cdots dx_n$$

for some n -dimensional function g . For instance, g could represent the total delay in queue of the first $[n/2]$ customers when the X values represent the first $[n/2]$ interarrival and service times.* In many situations, it is not analytically possible either to compute the preceding multiple integral exactly or even to numerically approximate it within a given accuracy. One possibility that remains is to approximate $E[g(\mathbf{X})]$ by means of simulation.

To approximate $E[g(\mathbf{X})]$, start by generating a random vector $\mathbf{X}^{(1)} = (X_1^{(1)}, \dots, X_n^{(1)})$ having the joint density $f(x_1, \dots, x_n)$ and then compute $Y^{(1)} = g(\mathbf{X}^{(1)})$. Now generate a second random vector (independent of the first) $\mathbf{X}^{(2)}$ and compute $Y^{(2)} = g(\mathbf{X}^{(2)})$. Keep on doing this until r , a fixed number of independent and identically distributed random variables $Y^{(i)} = g(\mathbf{X}^{(i)})$, $i = 1, \dots, r$ have been generated. Now by the strong law of large numbers, we know that

$$\lim_{r \rightarrow \infty} \frac{Y^{(1)} + \cdots + Y^{(r)}}{r} = E[Y^{(i)}] = E[g(\mathbf{X})]$$

* We are using the notation $[a]$ to represent the largest integer less than or equal to a .

and so we can use the average of the generated Y s as an estimate of $E[g(\mathbf{X})]$. This approach to estimating $E[g(\mathbf{X})]$ is called the *Monte Carlo simulation* approach.

Clearly there remains the problem of how to generate, or *simulate*, random vectors having a specified joint distribution. The first step in doing this is to be able to generate random variables from a uniform distribution on $(0, 1)$. One way to do this would be to take 10 identical slips of paper, numbered $0, 1, \dots, 9$, place them in a hat and then successively select n slips, with replacement, from the hat. The sequence of digits obtained (with a decimal point in front) can be regarded as the value of a uniform $(0, 1)$ random variable rounded off to the nearest $(\frac{1}{10})^n$. For instance, if the sequence of digits selected is 3, 8, 7, 2, 1, then the value of the uniform $(0, 1)$ random variable is 0.38721 (to the nearest 0.00001). Tables of the values of uniform $(0, 1)$ random variables, known as random number tables, have been extensively published (for instance, see The RAND Corporation, *A Million Random Digits with 100,000 Normal Deviates* (New York: The Free Press, 1955)). Table 11.1 is such a table.

However, this is not the way in which digital computers simulate uniform $(0, 1)$ random variables. In practice, they use pseudo random numbers instead of truly random ones. Most random number generators start with an initial value X_0 , called the seed, and then recursively compute values by specifying positive integers a , c , and m , and then letting

$$X_{n+1} = (aX_n + c) \text{ modulo } m, \quad n \geq 0$$

where the preceding means that $aX_n + c$ is divided by m and the remainder is taken as the value of X_{n+1} . Thus each X_n is either $0, 1, \dots$, or $m - 1$ and the quantity X_n/m is taken as an approximation to a uniform $(0, 1)$ random variable. It can be shown that subject to suitable choices for a , c , m , the preceding gives rise to a sequence of numbers that looks as if it were generated from independent uniform $(0, 1)$ random variables.

As our starting point in the simulation of random variables from an arbitrary distribution, we shall suppose that we can simulate from the uniform $(0, 1)$ distribution, and we shall use the term “random numbers” to mean independent random variables from this distribution. In Sections 11.2 and 11.3 we present both general and special techniques for simulating continuous random variables; and in Section 11.4 we do the same for discrete random variables. In Section 11.5 we discuss the simulation both of jointly distributed random variables and stochastic processes. Particular attention is given to the simulation of nonhomogeneous Poisson processes, and in fact three different approaches for this are discussed. Simulation of two-dimensional Poisson processes is discussed in Section 11.5.2. In Section 11.6 we discuss various methods for increasing the precision of the simulation estimates by reducing their variance; and in Section 11.7 we consider the problem of choosing the number of simulation runs needed to attain a desired level of precision. Before beginning this program, however, let us consider two applications of simulation to combinatorial problems.

Example 11.1 (Generating a Random Permutation) Suppose we are interested in generating a permutation of the numbers $1, 2, \dots, n$ that is such that all $n!$ possible orderings are equally likely. The following algorithm will accomplish this by first choosing one of the numbers $1, \dots, n$ at random and then putting that number in

Table 11.1 A Random Number Table

04839	96423	24878	82651	66566	14778	76797	14780	13300	87074
68086	26432	46901	20848	89768	81536	86645	12659	92259	57102
39064	66432	84673	40027	32832	61362	98947	96067	64760	64584
25669	26422	44407	44048	37937	63904	45766	66134	75470	66520
64117	94305	26766	25940	39972	22209	71500	64568	91402	42416
87917	77341	42206	35126	74087	99547	81817	42607	43808	76655
62797	56170	86324	88072	76222	36086	84637	93161	76038	65855
95876	55293	18988	27354	26575	08625	40801	59920	29841	80150
29888	88604	67917	48708	18912	82271	65424	69774	33611	54262
73577	12908	30883	18317	28290	35797	05998	41688	34952	37888
27958	30134	04024	86385	29880	99730	55536	84855	29080	09250
90999	49127	20044	59931	06115	20542	18059	02008	73708	83517
18845	49618	02304	51038	20655	58727	28168	15475	56942	53389
94824	78171	84610	82834	09922	25417	44137	48413	25555	21246
35605	81263	39667	47358	56873	56307	61607	49518	89356	20103
33362	64270	01638	92477	66969	98420	04880	45585	46565	04102
88720	82765	34476	17032	87589	40836	32427	70002	70663	88863
39475	46473	23219	53416	94970	25832	69975	94884	19661	72828
06990	67245	68350	82948	11398	42878	80287	88267	47363	46634
40980	07391	58745	25774	22987	80059	39911	96189	41151	14222
83974	29992	65381	38857	50490	83765	55657	14361	31720	57375
33339	31926	14883	24413	59744	92351	97473	89286	35931	04110
31662	25388	61642	34072	81249	35648	56891	69352	48373	45578
93526	70765	10592	04542	76463	54328	02349	17247	28865	14777
20492	38391	91132	21999	59516	81652	27195	48223	46751	22923
04153	53381	79401	21438	83035	92350	36693	31238	59649	91754
05520	91962	04739	13092	97662	24822	94730	06496	35090	04822
47498	87637	99016	71060	88824	71013	18735	20286	23153	72924
23167	49323	45021	33132	12544	41035	80780	45393	44812	12515
23792	14422	15059	45799	22716	19792	09983	74353	68668	30429
85900	98275	32388	52390	16815	69298	82732	38480	73817	32523
42559	78985	05300	22164	24369	54224	35083	19687	11062	91491
14349	82674	66523	44133	00697	35552	35970	19124	63318	29686
17403	53363	44167	64486	64758	75366	76554	31601	12614	33072
23632	27889	47914	02584	37680	20801	72152	39339	34806	08930

position n ; it then chooses at random one of the remaining $n - 1$ numbers and puts that number in position $n - 1$; it then chooses at random one of the remaining $n - 2$ numbers and puts it in position $n - 2$, and so on (where choosing a number at random means that each of the remaining numbers is equally likely to be chosen). However, so that we do not have to consider exactly which of the numbers remain to be positioned, it is convenient and efficient to keep the numbers in an ordered list and then randomly choose the position of the number rather than the number itself. That is, starting with any initial ordering $p_1, p_2, \dots, p_n$, we pick one of the positions $1, \dots, n$ at random and then interchange the number in that position with the one in position n . Now we

randomly choose one of the positions $1, \dots, n-1$ and interchange the number in this position with the one in position $n-1$, and so on.

To implement the preceding, we need to be able to generate a random variable that is equally likely to take on any of the values $1, 2, \dots, k$. To accomplish this, let U denote a random number—that is, U is uniformly distributed over $(0, 1)$ —and note that kU is uniform on $(0, k)$ and so

$$P\{i-1 < kU < i\} = \frac{1}{k}, \quad i = 1, \dots, k$$

Hence, the random variable $I = [kU] + 1$ will be such that

$$P\{I = i\} = P\{[kU] = i-1\} = P\{i-1 < kU < i\} = \frac{1}{k}$$

The preceding algorithm for generating a random permutation can now be written as follows:

Step 1: Let $p_1, p_2, \dots, p_n$ be any permutation of $1, 2, \dots, n$ (for instance, we can choose $p_j = j, j = 1, \dots, n$).

Step 2: Set $k = n$.

Step 3: Generate a random number U and let $I = [kU] + 1$.

Step 4: Interchange the values of p_I and p_k .

Step 5: Let $k = k - 1$ and if $k > 1$ go to step 3.

Step 6: $p_1, \dots, p_n$ is the desired random permutation.

For instance, suppose $n = 4$ and the initial permutation is $1, 2, 3, 4$. If the first value of I (which is equally likely to be either $1, 2, 3, 4$) is $I = 3$, then the new permutation is $1, 2, 4, 3$. If the next value of I is $I = 2$ then the new permutation is $1, 4, 2, 3$. If the final value of I is $I = 2$, then the final permutation is $1, 4, 2, 3$, and this is the value of the random permutation.

One very important property of the preceding algorithm is that it can also be used to generate a random subset, say of size r , of the integers $1, \dots, n$. Namely, just follow the algorithm until the positions $n, n-1, \dots, n-r+1$ are filled. The elements in these positions constitute the random subset. ■

Example 11.2 (Estimating the Number of Distinct Entries in a Large List) Consider a list of n entries where n is very large, and suppose we are interested in estimating d , the number of distinct elements in the list. If we let m_i denote the number of times that the element in position i appears on the list, then we can express d by

$$d = \sum_{i=1}^n \frac{1}{m_i}$$

To estimate d , suppose that we generate a random value X equally likely to be either $1, 2, \dots, n$ (that is, we take $X = [nU] + 1$) and then let $m(X)$ denote the number of times the element in position X appears on the list. Then

$$E\left[\frac{1}{m(X)}\right] = \sum_{i=1}^n \frac{1}{m_i} \frac{1}{n} = \frac{d}{n}$$