A SURVEY OF METHODS FOR SAMPLING FROM THE GAMMA DISTRIBUTION

Pandu R. Tadikamalla
Mark E. Johnson

ABSTRACT

Considerable attention has recently been directed at developing ever faster algorithms for generating gamma random variates on digital computers. This paper surveys the current state of the art including the leading algorithms of Ahrens and Dieter, Atkinson, Cheng, Fishman, Marsaglia, Tadikamalla and Wallace. General random variate generation techniques are explained with reference to these gamma algorithms. Computer simulation experiments on IBM and CDC computers are reported.

I. INTRODUCTION

The gamma distribution is a useful model for stochastic inputs to a wide variety of simulation applications. Computer generated gamma variates have been used to model interarrival and service times in queueing problems, as lead times and demand in inventory control, and as failure times in reliability models. The gamma distribution's popularity can be traced to the properties it obtains by the appropriate selection of its shape parameter. In this paper, we consider the gamma density in standardized form:

$$f(x) = x^{\alpha-1} \exp(-x)/\Gamma(\alpha),\ x \geq 0,\ \alpha > 0.$$

Several parameter values of α are particularly important. If $\alpha = 1$, then f is the density of the exponential distribution. If $\alpha = k$, an integer, then a k-Erlang distribution is obtained. Some simple transformations of gamma variates lead to other well-known distributions. If X has the density f, then 2X has a chi-squared distribution with 2α degrees of freedom. The ratio of independent chi-squared variates is an F variate; $X_1/(X_1 + X_2)$ is a beta variate if X_1 and X_2 are independent gamma variates. Finally, the limiting distribution of a gamma variate as $\alpha \to \infty$ is normal. Because of the gamma distribution's versatility and its appropriateness in simulation applications, considerable attention has been directed to improving methods for generating gamma random variates on a digital computer. The purpose of this paper is to examine various generation methods and to identify the state of the art. In section 2, some general univariate techniques are reviewed. Section 3 describes some of the leading algorithms. Finally, in section 4 we compare and contrast the algorithms. We also discuss future research directions.

II. GENERAL TECHNIQUES

The most common univariate random variate generation techniques can be roughly described as one of the following [18]: inverse probability integral transform, transformation, rejection and mixture. We will briefly describe each of these techniques in this section.

The first method is based on the following well-known result: if X is a continuous random variable with distribution function F, then U = F(X) has a uniform 0-1 distribution. The converse of this result leads to a random variate generation method: given a uniform 0-1 random number U, the variate $X = F^{-1}(U)$ has distribution function F. Application of this method is limited to variates having an inverse distribution function in simple closed form. For the gamma family, only the exponential distribution ($\alpha = 1$) enjoys this property. This leads to the exponential generation formula $X = -\ln(1 - U)$ or equivalently $X = -\ln(U)$, since U and 1 - U have the same uniform 0-1 distribution. For arbitrary shape parameter $\alpha \neq 1$, this approach fails since the evaluation of F^{-1} must be done numerically. However, for integer $\alpha = k$, we can apply the general transformation method. In particular, $X = -\ln(\pi_{i=1}^{n} U_i)$ has a gamma distribution with shape parameter k, if the U_i's are independent uniform 0-1. The transformation method can also be used to obtain chi-squared, F and beta variates as indicated in section 1.

The rejection method [10, 16, 19] has recently been the most fruitful approach to developing new algorithms for generating gamma variates. The algorithms of Ahrens and Dieter [1], Wallace [20], Atkinson [2], Marsaglia [13], Fishman [7], Cheng [5], and Tadikamalla [16, 17], are based on the rejection method. The idea of the rejection method is to generate variates from a density $h(x; \underline{\theta})$ which somewhat resembles the desired density $f(x)$. Occasionally, variates generated from h are rejected in such a way that the accepted variates have a distribution corresponding to f. Formally, let $f(x)$, $x \in \Omega$, be the density from which samples are required. Let $h(x; \underline{\theta})$ be another density which is easy to generate, has the same support as f, and which satisfies $f(x) \leq \delta h(x; \underline{\theta})$ for all $x \in \Omega$ and for some $\delta \geq 1$. The rejection method algorithm is:

1. Generate x having density $h(x; \underline{\theta})$.

2. Generate u which is uniform 0-1.

3. If $u \leq T(x) = f(x)/\delta h(x; \underline{\theta})$, go to 1. Otherwise, return x.

The variable δ is the expected number of "trials" until acceptance of x. The variable $1/\delta$ is generally referred to as the "efficiency" of the procedure. Several sometimes conflicting considerations enter into the selection of $h(x; \underline{\theta})$. They are summarized, as follows:

1. A fast, simple algorithm for generating variates from $h(x; \underline{\theta})$ must be available. (i.e., step 1 of the algorithm should be executed quickly.)

2. The efficency $1/\delta$ should be close to 1. (i.e., $h(x; \underline{\theta})$ should look like f.)

3. The acceptance-rejection test in step 3 should be simple. (i.e., T(x) should be easy to evaluate.)

4. δ must be computable from $\delta = \min\limits_{\underline{\theta}} [\max\limits_{x \in \Omega} f(x)/h(x; \underline{\theta})]$.

We will return to these considerations in relation to specific gamma algorithms in section 3.

The mixture method is based upon representing the density f from which variates are to be generated as $f(x) = p_1 f_1(x) + p_2 f_2(x) + \ldots p_n f_n(x)$, where $p_1 + p_2 + \ldots p_n = 1$ and each of the f_i's are densities. The rule of thumb in developing mixtures for f is to select the f_i's so that f_1 is the fastest f_i to generate, p_1 is close to 1, and $f_2, \ldots, f_n$ are not unduly difficult to generate. The corresponding mixture method algorithm is simply to generate variates from each f_i with probability p_i. The mixture method has not received the same attention for generating gamma variates as it has for normal and exponential variates (see Marsaglia [14, 15] and Kinderman and Ramage [11]). This is primarily due to the awkward problem that a different mixture must be used for each α value. Only in very large simulation studies could the effort in determining mixtures for the various values be justified.

III. LEADING ALGORITHMS

In this section we briefly describe some of the better algorithms for generating gamma variates. These algorithms are based on the rejection method--they differ only by their choice of $h(x;\underline{\theta})$. The simpler choices for $h(x;\underline{\theta})$ include the exponential (Fishman), the k-Erlang (Tadikamalla), the double exponential (Tadikamalla), and the log-logistic (Cheng) densities. Several authors have chosen $h(x;\underline{\theta})$ to be a mixture of two densities. These include a normal and an exponential (Ahrens and Dieter), a uniform and an exponential (Atkinson), and two k-Erlangs (Wallace). Marsaglia's "squeeze" method generates the cube root of a gamma variate using a normal density for $h(x;\underline{\theta})$.

Many of these algorithms have been streamlined considerably by their inventors to improve their relative performance. Preliminary fast acceptance tests to avoid evaluations of f(x) are employed in the published versions of the Cheng, Marsaglia and Atkinson algorithms. Another streamlining technique used in the Marsaglia and Atkinson algorithms involves generating a uniform variate via exp(-E), where E is a standard exponential variate. This leads to simplified acceptance tests by eliminating exponential function evaluations. For this technique to be worthwhile, however, a fast exponential generator must be used.

For formal statements of the algorithms, the reader is referred to the cited papers. Our hope is that the brief overview given here is sufficient to elucidate the commonality of the methods. In the next section we compare the algorithms on the basis of selected computer execution timings.

IV. SIMULATION RESULTS

In this section the results of our simulation experiments are reported. Seven of the leading algorithms are compared on the basis of computer execution times and core storage. The algorithms considered are preceded by their abbreviations, as follows

1. GO Ahrens and Dieter [1].
2. AT Atkinson [2].
3. GB Cheng [5].
4. GF Fishman [7].
5. MS Marsaglia [13].
6. T1 Tadikamalla's k-Erlang [16].
7. T2 Tadikamalla's double exponential [17].

Wallace's algorithm [20] was not considered, since Tadikamalla has shown T1 to be superior. Similarly, Greenwood's algorithm [8] was not included due to results given in [13].

The timings of our simulation experiments are given in Table 1 and Table 2. The algorithms were coded in FORTRAN using published versions where they were available. The results from Table 1 were obtained on the Kentucky Educational Network's IBM 370/165 computer. The individual times are based on generating 10,000 variates. The FORTRAN versions of Lurie and Mason's [12] uniform generator and Kinderman and Ramage's [11] normal generator were employed. The results from Table 2 were obtained on a Los Alamos Scientific Laboratory CDC 6600 computer. The individual times are based on generating 100,000 variates, and again employing the Kinderman-Ramage normal generator. CDC's RANF uniform generator was used.

The core storage requirements vary considerably for the algorithms. The IBM core storages in bytes are (GF, 526), (T1, 696), (GB, 768), (MS, 882), (T2, 980), (AT, 1068), and (GO, 1196). These values do not include the core storage of the uniform generator which was common to all the algorithms. For MS and GO, an additional 1162 bytes are required to store the Kinderman-Ramage normal generator.

From the preceding remarks and from the tables, we can make several

TABLE 1

IBM Timings (μ-seconds)

	1.5	2.25	3.5	4.5	5.5	10.5	15.5	50.5
GO	-	-	267	260	259	237	235	232
AT	181	197	230	232	256	309	337	523
GB	178	176	168	169	164	162	158	155
GF	161	212	245	277	297	416	490	-
MS	209	208	207	210	206	207	209	212
T1	161	195	233	248	271	406	492	-
T2	171	177	185	192	191	193	197	194

TABLE 2

CDC Timings (μ-seconds)

	1.5	2.25	3.5	4.5	5.5	10.5	15.5	50.5
GO	-	-	255	257	256	241	232	207
AT	224	251	281	294	311	376	427	656
GB	221	200	194	190	190	184	186	182
GF	213	267	326	362	398	541	574	1172
MS	194	190	185	184	186	180	177	176
T2	234	251	267	274	276	280	278	277

recommendations. Algorithms AT, GF and T1 are not competitive for α values greater than 2. Expected execution times increase considerably as α increases. On the IBM system, GF can be recommended for $\alpha \varepsilon$ [1, 2), if speed and simplicity are the dominant considerations (for these α, GF and T1 are equivalent). Algorithm GO improves nicely as α increases but is still inferior to both GB and MS. If only one algorithm could be recommended, we would advocate GB for the IBM. For CDC equipment and with speed the primary consideration, MS with the Kinderman-Ramage normal algorithm can be approved. Where both simplicity and speed are important, then GB can be advocated for both machines.

V. CONCLUSIONS

The state of the art for generating gamma random variates has been explored. The leading algorithms have been explained in a unifying framework and then compared on the basis of core storage and execution times. The algorithms of Cheng and Marsaglia appear to be the leading candidates presently. However, work should be continued in developing even faster algorithms. The appropriate direction would appear to be in the selection of $h(x; \underline{\theta})$.

BIBLIOGRAPHY

1. Ahrens, J. H., and Dieter, U. (1974) "Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions," Computing, 12, 223-246.
2. Atkinson, A. C. (1977) "An Easily Programmed Algorithm for Generating Gamma Random Variables," Applied Statistics, 140, 232-234.
3. Atkinson, A. C., and Pearce, M. C. (1976) "The Computer Generation of Beta, Gamma and Normal Random Variables," Journal of the Royal Statistical Society, A, 139, 431-461.
4. Box, G. E. P., and Muller, M. E. (1955) "A Note on the Generation of Normal Deviates," Annals of Mathematical Statistics, 29, 610-611.

5. Cheng, R. C. (1977) "The Generation of Gamma Random Variables with Non-integral Shape Parameter," Applied Statistics, 26, 71-75.
6. Dudewicz, E. J. (1975) "Speed and Quality of Random Numbers," Annual Technical Conference Transactions of ASQC, 29, 170-180.
7. Fishman, G. S. (1975) "Sampling from the Gamma Distribution on a Computer," Communications of the ACM, 19, 407-409.
8. Greenwood, A. J. (1974) "A Fast Generator for Gamma Distributed Random Variables," CompStat, Vienna: Physica Verlag, 19-27.
9. Johnk, M. D. (1964) "Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen," Metrika, 8, 5-15.
10. Johnson, M. E. (1976) Modeling and Generating Dependent Random Vectors, unpublished Ph.D. dissertation, The University of Iowa.
11. Kinderman, A. J., and Ramage, J. G. (1976) "Computer Generation of Normal Random Variables," Journal of the American Statistical Association, 71, 893-896.
12. Lurie, D., and Mason, R. L. (1973) "Empirical Investigation of Several Techniques for Computer Generation of Order Statistics," Communications in Statistics, 2, 363-371.
13. Marsaglia, G. (1977) "A Squeeze Method for Generating Random Variables," Computers and Mathematics with Applications, 3, 321-325.
14. Marsaglia, G. (1964) "Random Variables and Computers," in Trans. of the Third Prague Conf. on Information Theory, Statistical Decision Functions, Random Processes, June 1962, Prague: Publishing House of the Czechoslovak Academy of Sciences, 499-512.
15. Marsaglia, G. and Bray, T. A. (1964) "A Convenient Method for Generating Normal Random Variables," SIAM Review, 6, 260-264.
16. Tadikamalla, P. R. (1978) "Computer Generation of Gamma Random Variables," Communications of the ACM, 21, 419-422.
17. Tadikamalla, P. R. (1979) "Computer Generation of Gamma Random Variables II," Communications of the ACM, (to appear).
18. Tadikamalla, P. R. (1975) Modeling and Generating Stochastic Inputs for Simulation Studies, unpublished Ph.D. dissertation, The University of Iowa.
19. Tadikamalla, P. R., and Johnson, M. E. (1977) "Some Simple Rejection Methods for Sampling from the Normal Distribution," Proceedings of the First International Conference on Mathematical Modeling, St. Louis, MO, 573-578.
20. Wallace, N. D. (1974) "Computer Generation of Gamma Random Variates with Non-integral Shape Parameters," Communications of the ACM, 17, 691-695.
21. Whittekar, J. (1974) "Generating Gamma and Beta Variables with Non-integral Shape Parameters," Applied Statistics, 23, 210-213.
22. Wilson, E. B., and Hilferty, M. M. (1931) "The Distribution of Chi-Square," Proc. of the Nat. Academy of Sciences, 17, 688-689.