METHODS FOR GENERATING VARIATES FROM PROBABILITY DISTRIBUTIONS

by

J S DAGPUNAR, B.A., M.Sc.

A thesis submitted in fulfilment of the requirements for the award of the degree of Doctor of Philosophy.

Department of Mathematics & Statistics,
Brunel University,

May 1983

J S Dagpunar (1983) Methods for generating variates from probability distributions. Ph.D. Thesis, Department of Mathematics, Brunel University

ABSTRACT

Diverse probabilistic results are used in the design of random univariate generators. General methods based on these are classified and relevant theoretical properties derived. This is followed by a comparative review of specific algorithms currently available for continuous and discrete univariate distributions. A need for a Zeta generator is established, and two new methods, based on inversion and rejection with a truncated Pareto envelope respectively are developed and compared. The paucity of algorithms for multivariate generation motivates a classification of general methods, and in particular, a new method involving envelope rejection with a novel target distribution is proposed. A new method for generating first passage times in a Wiener Process is constructed. This is based on the ratio of two random numbers, and its performance is compared to an existing method for generating inverse Gaussian variates. New "hybrid" algorithms for Poisson and Negative Binomial distributions are constructed, using an Alias implementation, together with a Geometric tail procedure. These are shown to be robust, exact and fast for a wide range of parameter values. Significant modifications are made to Atkinson's Poisson generator (PA), and the resulting algorithm shown to be complementary to the hybrid method. A new method for Von Mises generation via a comparison of random numbers follows, and its performance compared to that of Best and Fisher's Wrapped Cauchy rejection method. Finally new methods are proposed for sampling from distribution tails, using optimally designed Exponential envelopes. Timings are given for Gamma and Normal tails, and in the latter case the performance is shown to be significantly better than Marsaglia's tail generation procedure.

ACKNOWLEDGEMENTS

It is a pleasure to record my gratitude to Professor P Macdonald for first suggesting I write this thesis, for subsequently supervising it, and for the helpful advice and guidance he has given me. I should like to acknowledge the financial assistance towards the cost of registration fees provided by the Governors of Dundee College of Technology. I should also like to record my appreciation to Mrs C Peters for her careful typing of the thesis. My greatest personal debt is to my wife Bridget, who has been a constant source of encouragement and support.

CONTENTS

PREFACE

In the area of Statistical Computing, the generation of observations (random variates) from probability distributions, has, within the last ten years, been the subject of active research. Until the early 70's, apart perhaps from the Normal and Negative Exponential distributions, many experimenters were still using inversion of the cumulative distribution function, frequently necessitating time consuming numerical work or approximations. More recently, there has been interest in developing fast algorithms, which generate "exact" variates without undue numerical work. It is perplexing to understand why the period 1950-1970 was such an inactive one. The probabilistic ideas underlying the algorithms now available were all well known, simulation was a recognised tool in scientific and management science investigations, and computers were beginning to be used. Indeed, it might have been expected, that the scarce availability of computing power at the beginning of this period would have motivated the early development of efficient algorithms.

The present situation is quite different. There is now a selection of methods available for most standard distributions. New methods continue to be proposed, with the result that a relatively recent review appearing in Fishman's, "Principles of Discrete Event Simulation" (1978), is now somewhat incomplete and dated. For example, no mention is made in that book of Band rejection, indexed search, or the ratio of uniforms methods.

One aim of the thesis therefore, is to fill this gap in the literature, by an extensive review of the current situation. Chapter 1 classifies general approaches to variate generation, the classification based on structural properties rather than historical order. This provides a framework for Chapters 2 and 3 which compare algorithms available for specific univariate distributions. In these first three chapters, emphasis is placed upon the theoretical properties which are likely to affect the speed of the generator. One reason for this is that it allows the designer of algorithms to exclude unpromising approaches at an early stage in the development. A second reason is that experimental measures of performance such as speed or storage requirements vary according to the computer in use. Thus it is useful to have measures, albeit imperfect ones, which are machine independent. Theoretical analyses provide such measures. A third reason is that theoretical measures provide a basis for explaining and understanding experimental results, and for providing qualitative descriptions of the likely behaviour on other machines. Finally, in reviewing material, it has been found that the theoretical properties in the original sources are sometimes incomplete, and by deriving them here, a firm foundation is placed for the new algorithms to be developed within the thesis.

The remainder of the thesis describes new algorithms or methods which have been developed. The first of these concerns the Zeta distribution, for which no generation methods could be found in the literature. Two complementary algorithms are developed within Chapter 3, allowing such variates to be generated for all values of the parameter of the distribution.

The situation described above is rather different when Multivariate distributions are considered. Although a few specific algorithms have appeared in the literature, the designer of algorithms has no classification of general methods to rely upon, as exists in the univariate case. This deficiency motivated the identification of three general methods in Chapter 4. A new method is proposed, whereby initial sampling is from a distribution constructed from the product of marginals of the distribution in question.

Several physical processes can be described by a Wiener process, and the problem of generating first passage times in such a process is considered in Chapter 5. It is shown that simple transformations allow any such variate to be generated by sampling from a single parameter (standardised) Wald distribution. An algorithm based on the ratio of uniform variates is developed. In constructing an enclosing rectangle for the acceptance region, a new Theorem is proved, which relates the size of the rectangle to the properties of the reciprocal distribution.

Recently, the Alias method for sampling from discrete distributions has received attention, but in its present form it is limited to distributions having a finite number of mass points. Chapter 6 describes robust "hybrid" algorithms for the Poisson and Negative Binomial distributions, utilising the Alias technique, with a Geometric tail procedure. These algorithms are extremely fast, but require considerable setting-up and storage. In the case of the Poisson, this has motivated a complementary and portable algorithm, details of which are also given in Chapter 6.

The Von Mises distribution is proving useful for modelling angular random variables. Chapter 7 describes an efficient method of generation based on a comparison of random numbers, and compares its performance with existing algorithms.

The ability to generate from the tail of a distribution is important, either because the investigator requires truncated variates, or because such a routine can be used as part of an algorithm for generating from the complete distribution.

Chapter 8 describes new methods for Gamma and Normal distribution tails, employing optimally designed Exponential envelopes.

CHAPTER 1

GENERAL METHODS FOR GENERATING RANDOM VARIATES

In this chapter general methods for generating random variates are reviewed. Amongst these we include : inversion of the cumulative distribution function (c.d.f.), stochastic model methods, rejection methods, the alias rejection method, and methods for sampling from discrete empirical distributions. In the next chapter, when procedures for specific distributions are given, it will be apparent that some procedures use a mixture of two or more of these methods.

1.1 Inversion Method

Given that we wish to generate random variates with a probability density function (p.d.f.) $f_X(\cdot)$, and c.d.f. $F_X(\cdot)$, we can show that

$$X = F_X^{-1}(R) \qquad (1.1.1)$$

has the required distribution, where $R \sim U(0,1)$. For, under the stated transformation,

$$\begin{aligned} P(X \le x) &= P\left(F_X^{-1}(R) \le x\right) \\ &= P\left(R \le F_X(x)\right) \\ &= F_X(x) \ . \end{aligned} \qquad (1.1.2)$$

If X is a discrete random variable, defined on the ordered values $S \equiv \{x^{(1)}, x^{(2)}, \ldots\}$, then variates may be generated by finding the smallest value of $X \in S$, such that

$$F_X(X) \ge R \ . \qquad (1.1.3)$$

To illustrate the inversion method for a continuous random variable, consider generation from the three parameter Weibull distribution with p.d.f.,

$$f_X(x) = \begin{cases} c(x-a)^{c-1} e^{-(\frac{x-a}{b})^c} / b^c & (x \geq a) \\ 0 & (x < a) \end{cases} \qquad (1.1.4)$$

where $b > 0$ and $c > 0$. The c.d.f. is

$$F_X(x) = \begin{cases} 1 - e^{-(\frac{x-a}{b})^c} & (x \geq a) \\ 0 & (x < a) \end{cases} . \qquad (1.1.5)$$

Using (1.1.1), we obtain

$$X = F_X^{-1}(R) = a + b[-\ell n(1-R)]^{1/c} ,$$

or, since R is identically distributed to $(1-R)$,

$$X = a + b[-\ell n\ R]^{1/c} . \qquad (1.1.6)$$

As an example of discrete random variate generation, consider the Geometric distribution with probability mass function (p.m.f.)

$$f_X(x) = (1-p)^{x-1} p \qquad (x = 1,2,\ldots) , \qquad (1.1.7)$$

where $0 < p < 1$. The c.d.f. is

$$F_X(x) = 1 - (1-p)^x , \qquad (1.1.8)$$

and using (1.1.3) we must find the smallest integer X satisfying

$$1 - (1-p)^X \geq R ,$$

or

$$\ell n(1-R) \geq X\ \ell n(1-p) .$$

As before $(1-R)$ may be replaced by R to give

$$X = 1 + \langle \ell n\ R/\ell n(1-p) \rangle .^{\dagger} \qquad (1.1.9)$$

The suitability of the inversion method depends mainly on whether the c.d.f. can be inverted analytically. If it can, it is often a good method to use, since it requires only one random number per variate generated and has low storage requirements since no look-up table for $F_X(\cdot)$ is required.

---

† $\langle x \rangle$ denotes the integer part of a non-negative real x.

If the c.d.f. cannot be inverted analytically, then, for a continuous random variable, one possibility is to find an approximation to the inverse c.d.f.. For example, Page (1977) uses the following approximation to the standard normal c.d.f.:

$$\Phi^{-1}(p) \simeq u - \frac{1}{3a_2 u} \qquad (p > 0.5) \ , \qquad (1.1.10)$$

where

$$u^3 = \{y + (y^2 + 4/[27a_2])^{\frac{1}{2}}\}/2a_2 \ ,$$

$$y = \ell n[p/(1-p)]/2a_1 \ .$$

$$a_1 = (2/\pi)^{\frac{1}{2}}$$

and $$a_2 = 0.044715 \ .$$

In the case of the Normal distribution, it is perhaps difficult to imagine circumstances where such a method will be used, since fast and exact procedures are available using other methods (to be discussed later). As Hoaglin (1975) says, "In general we should now regard approximate algorithms as a last resort, to be turned to only when exact ones are unavailable or hopelessly inefficient".

There is however a positive advantage in using an approximate inversion method in some cases which Hoaglin does not comment upon. The random variables $X = F_X^{-1}(R)$ and $Y = F_X^{-1}(1-R)$ will frequently display negative correlation (the correlation is -1 if the distribution is symmetric). This is useful from an experimental design viewpoint, since a reduction in variance can be achieved by "pooling" the two responses - the method of antithetic variates. Some of the other methods for generating random variates, in particular rejection methods, are unlikely to yield the same degree of negative correlation, because of the difficulty in ensuring that the same random numbers are used to generate both the primary and antithetic variates. There is however, no advantage in using an approximate inversion method for

the Normal distribution, or indeed any symmetric distribution, since a perfectly negatively correlated pair (X,Y) may be obtained by setting

$$Y = \mu - (X-\mu) = 2\mu - X \ . \qquad (1.1.11)$$

To utilise (1.1.11) it is of course essential that X can be replicated on the antithetic run. This can be achieved by reserving a random number stream for the X-values, or alternatively by storing the generated X-values for use in calculating Y-values during the antithetic run.

For an asymmetric distribution, approximate inversion methods should not be totally disregarded, contrary to Hoaglin's view. For example, although many exact Gamma generation algorithms have been devised using rejection methods, the rejection step is likely to reduce the induced negative correlation between primary and antithetic variates. One method, based on approximating the inverse c.d.f. is due to Tadikamalla and Ramberg (1975). A Gamma distribution with mean μ^* and variance σ^{*2} is approximated by a four parameter distribution having c.d.f.

$$F_X(x) = 1 - \{1 + [\mu + \left(\frac{x-\mu^*}{\sigma^*}\right)\sigma]^c\}^{-k} \ , \qquad (1.1.12)$$

where $x \geq \mu^* - \mu\sigma^*/\sigma$, and μ, σ, $c(>0)$, $k(>0)$ are parameters chosen so that the first four standardised moments about the mean are identical to the corresponding moments of the Gamma distribution. Tadikamalla and Ramberg produce regression equations which facilitate the fitting process. Since (1.1.12) can be inverted analytically, this approximate inversion method may be useful on those occasions when the generation of high quality antithetic variates is of more importance than the generation of exact variates.

1.2 Stochastic Model Methods

In this class of methods, a process or sampling procedure which gives rise to a statistic having the required distribution is identified. The process is then simulated to give typical values of the statistic.

For example, a Poisson variate X having p.m.f.,

$$f_X(x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad (x = 0,1,\ldots) \ , \qquad (1.2.1)$$

represents the number of events per unit time in a Poisson process, rate λ. The inter-event time is Negative Exponential, mean λ^{-1}. Thus X is the number of complete and independent Negative Exponential variates (mean λ^{-1}) that can be fitted into a unit time interval. Negative Exponential variates may be obtained by setting $a = 0$, $b = \lambda^{-1}$ and $c = 1$ in (1.1.5). Consequently to generate Poisson variates, given a stream of random numbers $\{R_i\}$, we require the largest integer X satisfying

$$\sum_{i=1}^{X} -\lambda^{-1} \ell n \, R_i \leq 1 \ , \qquad (1.2.2)$$

or equivalently

$$\prod_{i=1}^{X} R_i \geq e^{-\lambda} \ . \qquad (1.2.3)$$

As a second discrete distribution example, consider the Hypergeometric distribution with p.m.f.,

$$f_X(x) = \frac{\binom{g}{x}\binom{N-g}{n-x}}{\binom{N}{n}} \ ,$$

where X represents the number of reliable items from a random sample of $n(\leq N)$, drawn without replacement, from a population

consisting of g reliable items and (N-g) defective ones. To simulate this process, note that the probability that the first draw results in a reliable item is g/N , while the future probabilities depend upon previous outcomes. For example, the (conditional) probability of the second draw resulting in a reliable item is (g-1)/(N-1) or g/(N-1) according to whether the first draw was reliable or defective. Hence an algorithm for generating typical values is :

Algorithm 1.1

1. i = 1 , X = 0 .

2. p = g/N .

3. generate $R_i \sim U(0,1)$. If $R_i > p$ go to 5.

4. X = X + 1 , g = g - 1. If g = 0 deliver X.

5. N = N - 1.

6. If i = n deliver X, else i = i + 1 and go to 2.

1.3 Rejection Methods

A number of rejection methods exist, but the feature common to all of them is that a prospective sample variate is subjected to a random test which results either in acceptance or rejection of the sample variate.

1.3.1 Envelope Rejection

Suppose it is required to generate variates having a p.d.f. $f_X(\cdot)$ and c.d.f. $F_X(\cdot)$. The rejection method samples prospective variates from a different (target) distribution with p.d.f. $g_Y(\cdot)$ and c.d.f. $G_Y(\cdot)$, and applies a rejection/acceptance criterion, such that accepted Y have the desired distribution. The p.d.f. $g_Y(\cdot)$ should be chosen with two factors in mind. Firstly, the

generation of Y values should be fast and exact. Secondly, $g_Y(\cdot)$ should imitate $f_X(\cdot)$ as closely as possible, otherwise the proportion of rejected variates becomes unacceptably high. A measure of the dissimilarity between $f_X(\cdot)$ and $g_Y(\cdot)$ is provided by

$$M = \max_x \left\{ \frac{f_X(x)}{g_Y(x)} \right\} \quad , \tag{1.3.1}$$

the maximisation being over all x satisfying $f_X(x) > 0$.

The general sampling procedure is :

Algorithm 1.2

0. $M = \max_x \left\{ \frac{f_X(x)}{g_Y(x)} \right\}$.

1. generate Y from $g_Y(\cdot)$ (first-stage sampling) and $R \sim U(0,1)$.

2. If, $R > \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)}$ go to 1, else deliver $X = Y$.

The validity of the algorithm is established by showing that the c.d.f. of Y conditional upon $RM < f_X(Y)/g_Y(Y)$ is $F_X(\cdot)$. Let $h_{Y,R}(y,r) = g_Y(y)$ be the joint p.d.f. of Y and R. Then the c.d.f. of accepted variates is

$$P\left(Y \le x \mid R < \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)}\right) = \frac{P\left(R < \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)} \; ; \quad Y \le x\right)}{P\left(R < \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)}\right)} \tag{1.3.2}$$

$$= \frac{\int_{-\infty}^{x} dy \int_{0}^{\frac{1}{M}\frac{f_X(y)}{g_Y(y)}} h_{Y,R}(y,r)dr}{\int_{-\infty}^{\infty} dy \int_{0}^{\frac{1}{M}\frac{f_X(y)}{g_Y(y)}} h_{Y,R}(y,r)dr}$$

$$= \frac{\int_{-\infty}^{x} g_Y(y)dy \int_0^{\frac{1}{M}\frac{f_X(y)}{g_Y(y)}} dr}{\int_{-\infty}^{\infty} g_Y(y)dy \int_0^{\frac{1}{M}\frac{f_X(y)}{g_Y(y)}} dr}$$

$$= \frac{\int_{-\infty}^{x} g_Y(y)\left[\frac{1}{M}\frac{f_X(y)}{g_Y(y)}\right]dy}{\int_{-\infty}^{\infty} g_Y(y)\left[\frac{1}{M}\frac{f_X(y)}{g_Y(y)}\right]dy} = \frac{F_X(x)/M}{F_X(\infty)/M} \qquad (1.3.3)$$

$$= F_X(x) \ , \qquad (1.3.4)$$

as required. The denominator in (1.3.3) indicates that the probability of acceptance (the sampling efficiency) is M^{-1}. Clearly $M \geq 1$, but a good choice of $g_Y(\cdot)$ will result in M being close to unity.

To illustrate the method, suppose it is desired to generate standard Normal Variates X, using Variates Y from a standard Cauchy distribution. Then

$$f_X(x) = (2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}x^2} \ , \qquad (1.3.5)$$

and

$$g_Y(x) = [\pi(1+x^2)]^{-1} \ . \qquad (1.3.6)$$

Thus,

$$M = \max_x \left\{ \left(\frac{\pi}{2}\right)^{\frac{1}{2}} (1+x^2)e^{-\frac{1}{2}x^2} \right\} = \left(\frac{2\pi}{e}\right)^{\frac{1}{2}} = 1.520 \ ,$$

and the acceptance condition in step 2 of the algorithm becomes

$$R > \tfrac{1}{2}(1+Y^2)e^{-\frac{1}{2}(Y^2-1)} \ . \qquad (1.3.7)$$

First stage sampling from the Cauchy distribution is conveniently performed by noting that

$$G_Y(x) = \tfrac{1}{2} + \frac{1}{\pi}\arctan x \ . \qquad (1.3.8)$$

Given a random number R_1, inversion leads to

$$Y = \tan[\pi(R_1 - \tfrac{1}{2})] \ . \qquad (1.3.9)$$

Thus an algorithm for generating exact Normal variates is:

Algorithm 1.3

1. generate $R, R_1 \sim U(0,1)$.
2. $X = \tan[\pi(R_1 - \tfrac{1}{2})]$.
3. If $R > \tfrac{1}{2}(1+X^2)e^{-\frac{1}{2}(X^2-1)}$ go to 1, else deliver X.

Since $M = 1.520$, the procedure requires a mean of 3.04 random numbers per generated variate. Since the procedure also requires evaluation of Exponential and Tangent functions, it is unlikely to be very fast. The tangent evaluation may be eliminated by noting that if $U_1, U_2 \sim (-\tfrac{1}{2}, \tfrac{1}{2})$ then $\theta = \tan^{-1}(U_2/U_1) \sim U(0,2\pi)$ and is independently distributed of $U_1^2 + U_2^2 \sim U(0,\tfrac{1}{4})$, subject to $U_1^2 + U_2^2 \leq \tfrac{1}{4}$. Subject to this condition step 2 may be replaced by $X = U_2/U_1$, while step 4 becomes

$$4(U_1^2 + U_2^2) > \tfrac{1}{2}\left\{\frac{U_1^2 + U_2^2}{U_1^2}\right\} e^{-\frac{1}{2}(X^2-1)}$$

or

$$U_1^2 e^{\frac{1}{2}X^2} > \tfrac{1}{8}e^{\frac{1}{2}} \ .$$

1.3.2 Band Rejection

Payne (1977) refers to this little known method, which is a refinement of the envelope rejection method, applicable only to p.d.f.'s having a finite domain. His brief description of the method implies the following algorithm for sampling from a p.d.f. $f_X(x)$ with $0 \leq x \leq a$.

Algorithm 1.4

0. $M = \max_{0 \le x \le a} \left[\frac{f_X(x) + f_X(a-x)}{g_Y(x)} \right]$.

1. generate : $Y \sim g_Y(\cdot)$ (first stage sampling),

$R \sim U(0,1)$.

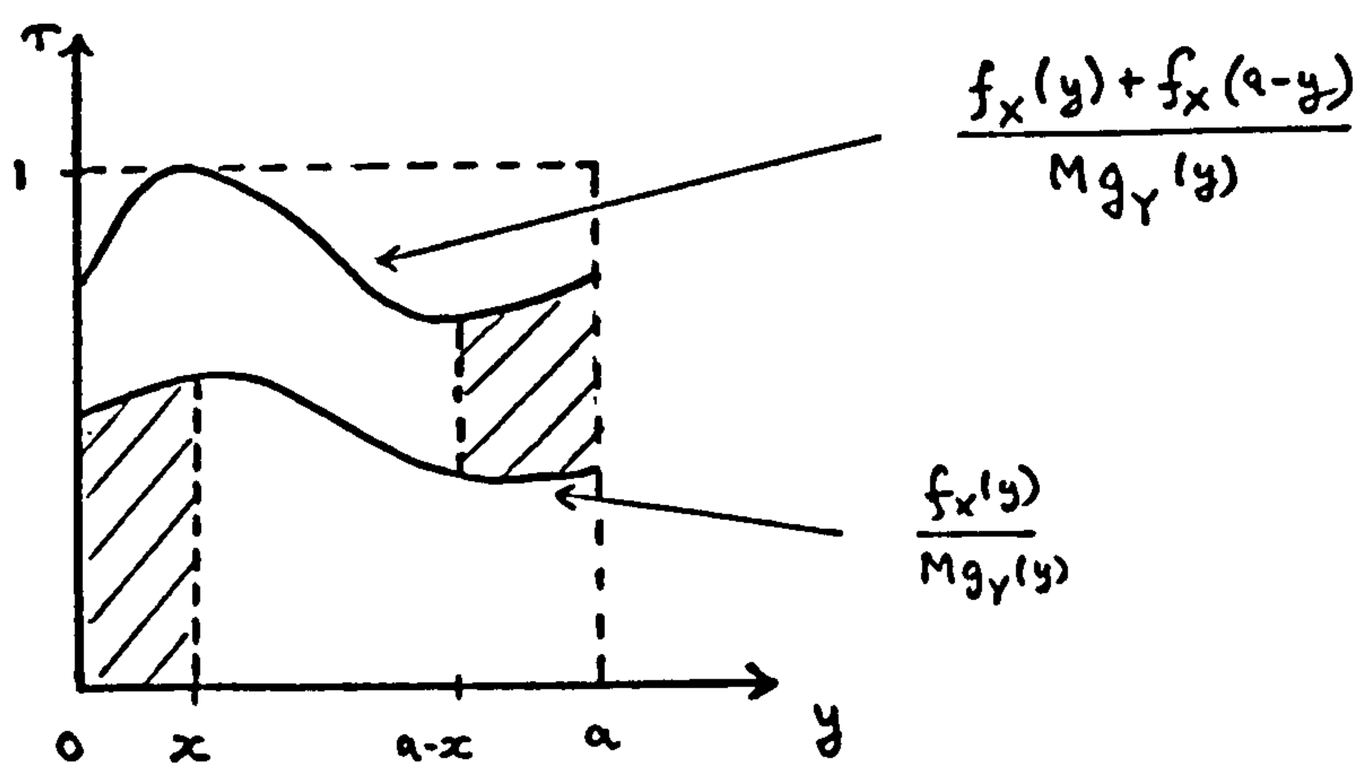
2. If $R < \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)}$ deliver $X = Y$.

3. If $R < \frac{1}{M} \frac{f_X(Y) + f_X(a-Y)}{g_Y(Y)}$ deliver $X = a - Y$.

4. Go to 1.

To demonstrate the validity of the algorithm we will show that the c.d.f. of X constructed in this way is $F_X(\cdot)$.

Validity:



Let $f_{Y,R}(y,r) = g_Y(y)$ be the joint p.d.f. of Y and R. Then,

$$P(X \le x) = P\left[\left\{Y \le x;\ R < \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)}\right\} \cup \left\{a-Y \le x;\ R > \frac{1}{M} \frac{f_X(Y)}{g_Y(Y)}\right\} \middle|\ R < \frac{f_X(Y)+f_X(a-Y)}{Mg_Y(Y)}\right] \tag{1.3.10}$$

$$= P\left[\left\{Y\le x; R < \frac{1}{M}\frac{f_X(Y)}{g_Y(Y)}\right\} \cup \left\{Y\ge a-x; R > \frac{f_X(Y)}{Mg_Y(Y)}\right\} \,\middle|\, R < \frac{f_X(Y)+f_X(a-Y)}{Mg_Y(Y)}\right]$$

$$= \frac{\displaystyle\int_0^x dy \int_0^{\frac{f_X(y)}{Mg_Y(y)}} f_{Y,R}(y,r)dr + \int_{a-x}^{a} dy \int_0^{\frac{f_X(y)+f_X(a-y)}{Mg_Y(y)}} f_{Y,R}(y,r)dr}{\displaystyle\int_0^a dy \int_0^{\frac{f_X(y)}{Mg_Y(y)}} f_{Y,R}(y,r)dr + \int_0^a dy \int_0^{\frac{f_X(y)}{Mg_Y(y)}} f_{Y,R}(y,r)dr} \quad . \qquad (1.3.11)$$

Since

$$\int_0^{\frac{f_X(y)}{Mg_Y(y)}} f_{Y,R}(y,r)dr = g_Y(y)\int_0^{\frac{f_X(y)}{Mg_Y(y)}} dr = \frac{f_X(y)}{M} \quad , \qquad (1.3.12)$$

and similarly for the other integrals, we have

$$P(X \le x) = \frac{\displaystyle\int_0^x \frac{f_X(y)}{M}dy + \int_{a-x}^{a} \frac{f_X(a-y)}{M}dy}{\displaystyle\int_0^a \frac{f_X(y)}{M}dy + \int_0^a \frac{f_X(a-y)}{M}dy} \qquad (1.3.13)$$

$$= \frac{\displaystyle\int_0^x \frac{f_X(y)}{M}dy + \int_0^x \frac{f_X(u)}{M}du}{\displaystyle\int_0^a \frac{f_X(y)}{M}dy + \int_0^a \frac{f_X(a-y)}{M}dy}$$

$$= \frac{2F_X(x)/M}{2F_X(a)/M} \quad . \qquad (1.3.14)$$

$$\therefore \quad P(X\le x) = F_X(x) \quad . \qquad (1.3.15)$$

We note from (1.3.14) that the sampling efficiency is $2F_X(a)/M = 2/M$. For a uniform target distribution,

$$M \leq 2 \, \text{Max}[f_X(x)/g_Y(x)] \quad , \tag{1.3.16}$$

and so in this case the sampling efficiency can never be worse than that of the conventional envelope rejection method.

1.3.3 Ratio of Uniforms Method

This is a method due to Kinderman and Monahan (1977) which is based on the acceptance or rejection of a variate which is generated from the ratio of two random numbers. It relies on the following result:

Let (U,V) be two random variables uniformly distributed over $C \equiv \{(u,v) : 0 \leq u \leq f_X^{\frac{1}{2}}(v/u) \; ; \; v/u \in S\}$ where S is the set of values over which a p.d.f. $f_X(\cdot)$ is defined. Then the p.d.f. of $Z = V/U$ is $f_X(\cdot)$.

To prove this result, consider a transformation $(u,v) \rightarrow (u,z)$ where $z = v/u$. The joint p.d.f. of U and Z is

$$f_{U,Z}(u,z) = f_{U,V}\big(u,v(u,z)\big)\,|J| \tag{1.3.17}$$

where

$$J = \begin{vmatrix} 1 & \frac{\partial u}{\partial z} \\ 0 & \frac{\partial v}{\partial z} \end{vmatrix} = u \; .$$

Thus,

$$f_{U,Z}(u,z) = u \Big/ \iint_C du\,dv \tag{1.3.18}$$

where $0 \leq u \leq f_X^{\frac{1}{2}}(z)$ and $z \in S$.

Hence the marginal p.d.f. of Z is

$$f_Z(z) = \int_0^{f_X^{\frac{1}{2}}(z)} u \, du \; / \iint_C du\,dv$$

$$= f_X(z)/2 \iint_C du\,dv \; . \tag{1.3.19}$$

Since $f_Z(z)$ must integrate to 1, it follows that

$$\iint_C dudv = \tfrac{1}{2} , \qquad (1.3.20)$$

and that

$$f_Z(z) = f_X(z) , \qquad (1.3.21)$$

showing that V/U has the desired distribution.

To generate from $f_X(\cdot)$, a conveniently shaped region D (usually, but not necessarily, rectangular or triangular) which completely encloses C is identified. Points with coordinates (U,V) are generated uniformly within D. If a point also lies in C, then X = V/U is accepted as a sample variate, otherwise the point is rejected.

Kinderman and Monahan apply the method to the Normal, Cauchy and Reciprocal uniform distributions. Cheng and Feast (1979) have also applied it to the Gamma distribution.

If the method is applied to a p.d.f. $f_X(\cdot)$ defined for $x \in (a,b)$, where (b-a) is finite, and if the p.d.f. is always bounded, with $f(x) \leq A$, then a possible choice for the enclosing area is the triangular region,

$$D \equiv \{(u,v) : 0 \leq u \leq A^{\frac{1}{2}} \; ; \; a \leq v/u \leq b\} , \qquad (1.3.22)$$

with area $\frac{1}{2}(b-a)A$. In this case the probability that points uniformly distributed through D fall in C is

$$P(\text{acceptance}) = \iint_C dudv \; / \iint_D dudv$$

$$= \tfrac{1}{2}/\{\tfrac{1}{2}(b-a)A\} = (b-a)^{-1}A^{-1} . \qquad (1.3.23)$$

This is also the probability of acceptance using the envelope rejection method with a uniform target distribution.

An important qualification therefore, which Kinderman and Monahan do not make, is that their method is not preferable to the

conventional envelope rejection method with such distributions, if the setting of the maximum value of u is performed merely by looking at the maximum value of $f_X^{\frac{1}{2}}(\cdot)$ and a triangular enclosing region is used.

The method would appear to have potential for p.d.f.s defined over an infinite range or having an unbounded p.d.f.. An application is given in Chapter 5, where the method is used to generate first passage times in a Wiener process.

At this point we will note that a contributory factor to the efficiency of the general method is the degree to which C can be 'tightly' enclosed by a polygon, within which points may be generated uniformly. The obvious choices for such polygons are rectangles and triangles, but better fits may be obtained through other shapes. In this respect a method due to Hsuan (1979) for generating uniform polygonal random pairs without rejection may prove useful.

1.3.4 Comparison of Random Numbers

This is based on a method alluded to by von Neumann in 1949 in a procedure for generating Negative Exponential Variates, using just a comparison of random numbers. At the time, von Neumann suggested that the method could be generalised to generate from any p.d.f. satisfying a first order differential equation. Forsythe (1972) gives an exposition of what he assumes von Neumann had in mind.

Suppose X_j is a random variable uniformly distributed in the interval (q_{j-1}, q_j) and $h_j(x)$ is a known continuous-valued function of x with $0 \leq h_j(x) \leq 1$ within the interval. Let $\{R_i\}$ denote a sequence of random numbers, and N be an integer such that:

$$N = 1 \quad \text{if} \quad h_j(X_j) < R_1 \ ,$$

$$N = n \ (>1) \quad \text{if} \quad h_j(X_j) \geq R_1 \ \ldots \geq R_{n-1} < R_n \ . \tag{1.3.24}$$

It is easily shown (see for example Fishman 1978, pp 400-401) that the p.d.f. of X_j conditional upon N being odd-valued is

$$\psi_j(x) = e^{-h_j(x)} \Big/ \int_{q_{j-1}}^{q_j} e^{-h_j(x)}\,dx \qquad (q_{j-1} \leq x \leq q_j). \tag{1.3.25}$$

Suppose we wish to sample from $f_X(\cdot)$. Then the p.d.f. may be expressed as the probability mixture,

$$f_X(x) = \sum_j p_j \psi_j(x) \ , \tag{1.3.26}$$

where

$$\psi_j(x) = \begin{cases} e^{-h_j(x)} \Big/ \int_{q_{j-1}}^{q_j} e^{-h_j(u)}\,du & (q_{j-1} \leq x \leq q_j) \\ 0 & \text{elsewhere} \ , \end{cases} \tag{1.3.27}$$

$$p_j = \int_{q_{j-1}}^{q_j} f_X(u)\,du \ , \tag{1.3.28}$$

and $q_0 < q_1 < q_2 \ldots < q_{j-1} < q_j < \ldots$.

The critical requirement for the result (1.3.27) to be useful in Variate generation is that the intervals (q_{j-1}, q_j) and the functions $h_j(x)$ are chosen in such a way that

$$0 \leq h_j(x) \leq 1 \qquad (q_{j-1} \leq x \leq q_j) \ . \tag{1.3.29}$$

Given the representation (1.3.26), the sampling procedure is to generate from $\psi_j(x)$ with probability p_j. Given a random number R, the correct interval, (q_{j-1}, q_j) is selected by finding the smallest j such that

$$\sum_{s=1}^{j} p_s \geq R \ . \qquad (1.3.30)$$

Sampling from $\psi_j(x)$ is accomplished by accepting X_j if N is odd, otherwise rejecting it.

The advantage of "Forsythe's method", as it is now called, is that $h_j(x)$ is often an easy function to evaluate, and the comparison (1.3.24) is relatively fast, providing N is not too large. A disadvantage is that in most cases (Negative Exponential excluded) a table of p_j values must be calculated and stored. One major advantage is that by reserving one random number for interval selection, antithetic intervals may be generated on primary and antithetic runs.

In assessing the efficiency of any algorithm using this method, it is useful to compute the expected number of random numbers required to generate one variate from $\psi_j(\cdot)$. It is shown in Fishman (1978, pp 401) that this is

$$N_j = \frac{(q_j - q_{j-1}) + \int_{q_{j-1}}^{q_j} e^{h_j(u)}\, du}{\int_{q_{j-1}}^{q_j} e^{-h_j(u)}\, du} \ , \qquad (1.3.31)$$

and hence that the mean number of random numbers required (excluding the one used for interval selection) to generate one variate from $f_X(\cdot)$ is

$$\bar{N} = \sum_j \left\{ \frac{(q_j - q_{j-1}) + \int_{q_{j-1}}^{q_j} e^{h_j(u)}\, du}{\int_{q_{j-1}}^{q_j} e^{-h_j(u)}\, du} \right\} p_j \ . \qquad (1.3.32)$$

To illustrate the method, consider generation from a folded standard Normal distribution with p.d.f.

$$f_X(x) = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} \qquad (x \geq 0) \quad . \tag{1.3.33}$$

Generation from the usual standard Normal distribution follows by assigning a random sign to X. Using the representation (1.3.26), we define

$$h_j(x) = \tfrac{1}{2}(x^2 - q_{j-1}^2) \qquad (q_{j-1} \leq x \leq q_j) \quad . \tag{1.3.34}$$

The q_j-values are selected so that the interval widths are maximised, thereby minimising the number of intervals required. Since $h_j(x)$ is an increasing function of x this is equivalent to solving the difference equations:

$$\begin{aligned} q_0 &= 0 \quad , \\ \tfrac{1}{2}(q_j^2 - q_{j-1}^2) &= 1 \quad . \end{aligned} \tag{1.3.35}$$

The solution to (1.3.36) is

$$q_j = (2j)^{\frac{1}{2}} \quad . \tag{1.3.36}$$

Hence,

$$p_j = \int_{(2[j-1])^{\frac{1}{2}}}^{(2j)^{\frac{1}{2}}} \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} dx \qquad (j = 1,2,\ldots,k) \quad . \tag{1.3.37}$$

The number of intervals k, and therefore the truncation point for the distribution, is chosen so that the tail probability is negligible. For example, if the tolerable tail probability is 2×10^{-4}, then, from standard Normal tables, we require the smallest k such that

$$3.72 \leq (2k)^{\frac{1}{2}} \quad ,$$

that is 7 intervals.

As Parker (1976) has independently noted, the choice of intervals differs from that proposed by Forsythe and Atkinson and Pearce (1976). The latter state that intervals defined by

$$q_0 = 0 \ , \qquad q_j = (2j - 1)^{\frac{1}{2}} \quad (j \geq 1) \ , \tag{1.3.38}$$

have maximum width. Clearly this is not the case as the first interval has width 1, compared with a width of $\sqrt{2}$ under the scheme (1.3.36). The reason for their choice would appear to stem from a misleading statement in Forsythe's paper that the interval width should never exceed 1. Under the scheme (1.3.38) a tail probability of 2×10^{-4} would require the smallest k satisfying

$$3.72 \leq (2k - 1)^{\frac{1}{2}} \ , \tag{1.3.39}$$

a requirement of 8 intervals, compared with 7 resulting from (1.3.36).

This in itself does not necessarily favour the scheme (1.3.36), particularly as narrow intervals will require less random numbers per interval, due to the smaller variation of values taken by the exponent $h_j(x)$. Offset against this is the fact that a larger number of intervals will require slightly more time to select the appropriate interval.

Finally in this section we remark that Monahan (1979) has generalised von Neumann's (Exponential) and Forsythe's methods to yield a class of algorithms suitable for sampling from c.d.f.s having certain power series representations.

1.4 Alias Rejection Method

The Alias Rejection method devised by Walker (1974) is an efficient procedure for generating discrete random variables having a finite number (m) of mass points. It is in part an envelope rejection procedure, with first stage sampling from a uniform

distribution. However, if the acceptance test is not passed, the prospective Variate undergoes a deterministic transformation, the transformed variate being accepted. It is therefore a highly attractive method in that "rejected values" are not lost.

Kronmal and Peterson (1979) give a theoretical justification for the procedure which does not appear in Walker's account. They show that any m-point discrete distribution may be represented by a certain equi-probable mixture of m, two-point distributions.

To demonstrate this, consider, without loss of generality, a discrete distribution $f_m(x)$ $(x = 0,1,2,\ldots,m-1)$. Then $f_m(x)$ may be represented as the following mixture of a (m-1) point distribution $f_{m-1}(x)$ and a two-point distribution $g_{j_m}(x)$:

$$f_m(x) = \left(\frac{m-1}{m}\right) f_{m-1}(x) + \left(\frac{1}{m}\right) g_{j_m}(x) \qquad (1.4.1)$$

where,

$$f_{m-1}(x) = \begin{cases} \left(\frac{m}{m-1}\right) f_m(x) & (x \neq j_m, a_{j_m}) \\ 0 & (x = j_m) \\ \left(\frac{m}{m-1}\right)[f_m(j_m) + f_m(a_{j_m})] - \left(\frac{1}{m-1}\right) & (x = a_{j_m}) \end{cases} \qquad (1.4.2)$$

and

$$g_{j_m}(x) = \begin{cases} 0 & (x \neq j_m, a_{j_m}) \\ mf_m(j_m) & (x = j_m) \\ 1 - mf_m(j_m) & (x = a_{j_m}), \end{cases} \qquad (1.4.3)$$

j_m and a_{j_m} being values belonging to the domain of $f_m(x)$ such that

$$f_m(j_m) \leq \left(\frac{1}{m}\right) \qquad (1.4.4)$$

and

$$f_m(a_{j_m}) \geq \left(\frac{1}{m}\right). \qquad (1.4.5)$$

To establish the validity of the representation (1.4.1) - (1.4.5) we need only demonstrate (i) that a pair j_m and a_{j_m} satisfying (1.4.4)-(1.4.5) will always exist, (ii) that $f_{m-1}(x)$ is a bona-fide probability distribution, (iii) that $g_{j_m}(x)$ is a bona-fide probability distribution. (i) follows otherwise $\sum_x f_m(x) \neq 1$; (ii) follows since $f_{m-1}(x) \geq 0$ $(x \neq a_{j_m})$, $f_{m-1}(a_{j_m}) \geq 0$ as a consequence of (1.4.5), and $\sum_x f_{m-1}(x) = 1$; (iii) follows since $g_{j_m}(j_m) \geq 0$, $g_{j_m}(a_{j_m}) \geq 0$ as a consequence of (1.4.4), and $\sum_x g_{j_m}(x) = 1$.

Applying the representation (1.4.1) - (1.4.5) recursively (m-1) times we find that

$$f_m(x) = \left(\frac{1}{m}\right) \sum_{j=0}^{m-1} g_j(x) \qquad (1.4.6)$$

where

$$g_j(x) = \begin{cases} 0 & (x \neq j,\ a_j) \\ c_j & (x = j) \\ 1-c_j & (x = a_j) \end{cases} . \qquad (1.4.7)$$

Thus $f_m(x)$ is an equi-probable mixture of m two-point distributions. Note that one of the mass points of $g_j(x)$ is j itself, with probability c_j (the 'cut-off' value). The other mass point is a_j, the 'alias' of j. The cut-off and alias values are determined algorithmically through (1.4.1) - (1.4.5). Kronmal and Peterson (1978) give a 40 statement Fortran listing for such a procedure, and note that execution time is proportional to m, the number of mass points.

Once the setting up of c_j and a_j values is effected, the sampling routine takes the following simple form:

Algorithm 1.5

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

2. $X = Rm$.

3. If $X - \langle X \rangle \leq c_{\langle X \rangle}$ deliver $\langle X \rangle$, else deliver $a_{\langle X \rangle}$.

Note that $\langle X \rangle$ is uniformly distributed on the integers $[0,m-1]$, that $X - \langle X \rangle \sim U(0,1)$, and that consequently step 3 ensures that $\langle X \rangle$ is delivered with probability $c_{\langle X \rangle}$, and $a_{\langle X \rangle}$ with probability $1 - c_{\langle X \rangle}$, as required. Only one random number is required per Variate generation. The method as described by Walker and Kronmal and Peterson can only be applied to distributions having a finite number of mass points. In Chapter 6, the method is modified to generate variates from distributions having an infinite number of mass points.

An analagous procedure to the Alias Rejection method is the Rejection Mixture method for continuous distributions described originally in Krommal, Peterson and Lundberg (1978) and more recently under the name of the Acceptance Complement Method in Kronmal and Peterson (1981).

Suppose it is required to generate Variates X where,

$$f_X(x) = pf_Z(x) + (1-p)f_W(x) \quad (0 \leq p \leq 1). \qquad (1.4.8)$$

Let $h_U(\cdot)$ be any p.d.f. dominating $pf_Z(\cdot)$. Then Algorithm (1.6) below generates X-values.

Algorithm 1.6

1. generate $U \sim h_U(\cdot)$ and $R \sim U(0,1)$.

2. If $R < pf_Z(U)/h_U(U)$ deliver $X = U$, else generate $W \sim f_W(\cdot)$ and deliver $X = W$.

It is clear that accepting W, following rejection of U, is analogous to the acceptance of the alias value in the discrete case.

As no proof of the algorithm is given by Kronmal, Peterson and Lundberg it is instructive to establish the validity. The joint p.d.f. of R, U and W is $h_U(\cdot)f_W(\cdot)$ and hence

$$P(X \le x) = \int_{-\infty}^{\infty} f_W(w)dw \int_{-\infty}^{x} h_U(u)du \int_{0}^{pf_Z(u)/h_U(u)} dr$$

$$+ \int_{-\infty}^{x} f_W(w)dw \int_{-\infty}^{\infty} h_U(u)du \int_{pf_Z(u)/h_U(u)}^{1} dr \quad . \qquad (1.4.9)$$

Since $h_U(u) \ge pf_Z(u)$,

$$\int_{0}^{pf_Z(u)/h_U(u)} dr = pf_Z(u)/h_U(u)$$

and hence

$$P(X \le x) = \int_{-\infty}^{x} pf_Z(u)du + \int_{-\infty}^{\infty}[h_U(u)-pf_Z(u)]du \int_{-\infty}^{x} f_W(w)dw \ , \qquad (1.4.10)$$

whence

$$f_X(x) = pf_Z(x) + (1-p)f_W(x) \ , \qquad (1.4.11)$$

as required.

1.5 Polynomial Sampling

This idea has been used in connection with the generation of Normal Variates (Ahrens and Dieter, 1972), but would appear to be of wider applicability. Suppose we wish to generate variates having p.d.f.

$$f_X(x) = \sum_{j=0}^{k} b_j x^j \qquad (0 \le x \le 1) \ , \tag{1.5.1}$$

where the coefficients $\{b_j\}$ are not necessarily all non-negative.

Define sets S and T as,

$$S = \{j \mid b_j \ge 0 \ ; \ j \ne 0\} \quad ; \quad T = \{j \mid b_j < 0 \ ; \ j \ne 0\}. \tag{1.5.2}$$

Then (1.5.1) may be expressed as

$$f_X(x) = \sum_{j=0}^{k} w_j f_{X_j}(x) \ , \tag{1.5.3}$$

where

$$f_{X_0}(x) = 1 \tag{1.5.4}$$

$$f_{X_j}(x) = \begin{cases} (j+1)x^j & (j \in S) \\ \left(\frac{j+1}{j}\right)(1-x^j) & (j \in T) \ , \end{cases} \tag{1.5.5}$$

$$w_0 = b_0 + \sum_{j \in T} b_j \ , \tag{1.5.6}$$

and

$$w_j = \begin{cases} b_j/(j+1) & (j \in S) \\ -j\, b_j/(j+1) & (j \in T) \ . \end{cases} \tag{1.5.7}$$

We note that the $\{f_{X_j}(\cdot)\}$ are bona-fide p.d.f.s, that $w_j \ge 0$ $(j \ge 1)$, and that $w_0 \ge 0$ providing

$$b_0 + \sum_{j \in T} b_j \ge 0 \ . \tag{1.5.8}$$

(1.5.8) is the condition for (1.5.3) to be a valid probability mixture representation of (1.5.1). If (1.5.8) is satisfied, then sampling can proceed by generating X_j with probability w_j. Unless all the $\{b_j\}$ are non-negative, then sampling from the $\{f_{X_j}(\cdot)\}$ is non-trivial, since some of the associated c.d.f.s cannot be inverted analytically.

However the following method involving the generation of (j+1) random numbers can be used to sample from $f_{X_j}(\cdot)$, for both $j \in S$ and $j \in T$. Let $\{R_1, R_2, \ldots, R_{j+1}\}$ be a set of (j+1) random numbers and

$$V \sim U(0,U) , \tag{1.5.9}$$

where

$$U = \text{Max}(R_1, R_2, \ldots, R_{j+1}) . \tag{1.5.10}$$

Then the c.d.f.s of U and V are

$$F_U(x) = x^{j+1} \qquad (0 \le x \le 1) , \tag{1.5.11}$$

and

$$F_V(x) = \int_x^1 (x/u)(j+1)u^j du + \int_0^x (j+1)u^j du$$

$$= \left(\frac{j+1}{j}\right)x - \frac{x^{j+1}}{j} \qquad (0 \le x \le 1) , \tag{1.5.12}$$

which are identical to the c.d.f.s associated with (1.5.5).

Algorithm 1.7 below generates variates from p.d.f.s (1.5.4)-(1.5.5). If $j = 0$, then X_0 is simply delivered as a random number. If $j \neq 0$ and $b_j \ge 0$, then U, the maximum of (j+1) random numbers is delivered. If $j \neq 0$ and $b_j < 0$, then V is generated by delivering R_{j+1} if it is not the maximum of $R_1, R_2, \ldots, R_j, R_{j+1}$, otherwise R_j, the previous random number is delivered.

Algorithm 1.7

1. If $j = 0$, deliver $X_0 = R_1$.

2. $i = 1$, $R_{max} = -\infty$.

3. Generate R_i. If $i = j + 1$ go to 5.

4. $R_{prev} = R_i$. If $R_i > R_{max}$, $R_{max} = R_i$.
   $i = i + 1$, go to 3.

5. If $b_j < 0$ go to 6. If $R_i > R_{max}$ deliver $X_j = R_i$, else deliver $X_j = R_{max}$.

6. If $R_i > R_{max}$, deliver $X_j = R_{prev}$, else deliver $X_j = R_i$.

1.6 Sampling from Discrete Empirical Distributions

Some of the methods already dealt with are applicable to discrete distributions, for example Poisson sampling via a simulation of the Poisson process, and Geometric sampling through inversion of the c.d.f.. In many simulation studies, it appears that the underlying distribution is empirical, reflecting the raw data, with no attempt being made to fit a standard distribution to the data†. In these cases, methods are required for generating variates from tables. Clearly the methods to be developed are also applicable to standard distributions, which can always (if required) be expressed in tabular form. In reviewing the various methods, we will define

$$\left.\begin{aligned} p_x &= f_X(x) \\ q_x &= \sum_{j \le x} p_j \end{aligned}\right\} \quad (x = 1,2,\ldots,n) , \tag{1.6.1}$$

$$q_0 = 0 \tag{1.6.2}$$

where, without loss of generality, the random variable is defined on the integers $\{1,2,\ldots,n\}$, and if necessary (i.e. if the probability mass function has infinite domain), the distribution is truncated at same suitable point n.

1.6.1 Sequential Search

This is simply the inversion method applied to a c.d.f., represented in tabular form, and results in algorithm 1.8 below.

---

† It could be argued that it is preferable to fit a standard distribution to any empirical distribution. From a modelling viewpoint, subsequent experiments involving parametric analysis may be performed simply by altering the parameter values of the standard distribution.

Algorithm 1.8

0. $q_x = \sum_{j \le x} p_j \qquad (x = 1,2,\ldots,n)$.

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

2. $j = 1$.

3. If $q_j \ge R$, deliver $X = j$.

4. $j = j + 1$ and go to 3.

The speed of the algorithm is related to the mean number of comparisons (step 3) required, prior to delivery of X. This is given by

$$E = \sum_{i=1}^{n} ip_i = n - \sum_{k=1}^{n-1} q_k \,. \qquad (1.6.3)$$

1.6.2 Ordered Sequential Search

Expression (1.6.3) indicates that E may be reduced by a preliminary sort of the integers $\{1,2,\ldots,n\}$, such that the sorted table of probabilities is non-increasing. This gives rise to algorithm 1.9.

Algorithm 1.9

0. Find a permutation $\{r_1,r_2,\ldots,r_n\}$ of $\{1,2,\ldots,n\}$, such that

$$p_{r_1} \ge p_{r_2} \ge \ldots \ge p_{r_n}.$$

$$q'_x = \sum_{j \le x} p_{r_j} \qquad (x = 1,2,\ldots,n).$$

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

2. $j = 1$.

3. If $q'_j \ge R$, deliver $X = r_j$.

4. $j = j + 1$ and go to 3.

Comparing the two algorithms $q_j' \geq q_j$, with equality for all j, only if the original distribution was non-increasing for all j. Thus, excluding time for the sort, (1.6.3) indicates that algorithm 1.9 is no slower, and in general faster than algorithm 1.8.

1.6.3 Marsaglia's Method

Both the previous methods can be speeded up, at the cost of increased storage, by a scheme due to Marsaglia (1963). Suppose the probabilities $\{p_x\}$ are expressed to m decimal places, and are all non zero. Define,

$$A(j) = \begin{cases} 1 & (0 < j \leq 10^m q_1) \\ 2 & (10^m q_1 < j \leq 10^m q_2) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ n & (10^m q_{n-1} < j \leq 10^m q_n) \end{cases} . \qquad (1.6.4)$$

Then algorithm 1.10 generates variates from the probability distribution.

Algorithm 1.10

1. generate $R \sim U(0,1)$.
2. $X = A(\langle 10^m R \rangle + 1)$.

The validity follows immediately, since

$$P(X = i) = P(10^m q_{i-1} < \langle 10^m R \rangle + 1 \leq 10^m q_i)$$
$$= \frac{10^m q_i - 10^m q_{i-1}}{10^m} , \qquad (1.6.5)$$

since $\langle 10^m R \rangle + 1$ is uniformly distributed on the integers $\{1,2,\ldots,10^m\}$. From (1.6.5) we obtain

$$P(X = i) = q_i - q_{i-1} = p_i , \qquad (1.6.6)$$

as required.

The algorithm is fast, but rapidly becomes impracticable, since 10^m storage locations are required.

1.6.4 Method of Norman and Cannon

This was first suggested by Marsaglia (1963). Implementation details are given by Norman and Cannon (1972). For ease of exposition the method is explained first with the aid of a small example. Subsequently an algorithm will be given and verified, as no formal proof is given by Norman and Cannon.

Consider the distribution defined by :

p_1 =	0.3	5	6
p_2 =	0.2	7	9
p_3 =	0.3	6	5
	8	18	20

Each probability is expressed to three decimal places, and the entries in the fourth row give the column sums for each decimal place. An array A(j) containing 46 elements is set up. Of the first eight elements, the first three are filled with ones, the next two with twos, and the last three with threes. Of the next 18 elements, the first five are filled with ones, the next seven with twos and the last six with threes. Of the final 20 elements, the first six are filled with ones, the second nine with twos and the last five with threes.

Given a random number R, if <10R> < 8, then the random variate is delivered as A(<10R> + 1). The effect of this is that, conditional upon $0 \leq R < 0.8$, the numbers 1, 2 and 3 are returned with probabilities $\frac{3}{8}$, $\frac{2}{8}$ and $\frac{3}{8}$ respectively. If the first test fails, then a check is made to establish whether $8 \times 10 \leq$ <100R> $< 8 \times 10 + 18$. If so, then the random variate is delivered as A(8 + <100R> - 80 + 1). Thus conditional upon $0.8 \leq R < 0.98$, the probabilities of obtaining

1, 2 and 3 are $\frac{5}{18}$, $\frac{7}{18}$ and $\frac{6}{18}$ respectively. If this test fails, it must be the case that $8 \times 100 + 18 \times 10 \leq \langle 1000R \rangle < 8 \times 100 + 18 \times 10 + 20 = 1000$, in which case the random variable is delivered as $A(26 + \langle 1000R \rangle - 980 + 1)$. Thus, conditional upon $0.98 \leq R < 1$, the probabilities of obtaining 1, 2 and 3 are $\frac{6}{20}$, $\frac{9}{20}$ and $\frac{5}{20}$ respectively. Using this procedure we can, for example, deduce the probability that the random variate $X = 1$:

$$\begin{aligned} P(X = 1) &= \tfrac{3}{8}\, P(\langle 10R \rangle < 8) + \tfrac{5}{18}\, P(80 \leq \langle 100R \rangle < 98) \\ &\quad + \tfrac{6}{20}\, P(980 \leq \langle 1000R \rangle < 1000) \\ &= \tfrac{3}{8}\,(0.8) + \tfrac{5}{18}\,(0.18) + \tfrac{6}{20}\,(0.02) \\ &= 3.10^{-1} + 5.10^{-2} + 6.10^{-3} = 0.356 , \end{aligned}$$

as required.

To formalise the procedure, suppose p_i is expressed to m decimal places:

$$p_i = 0 \cdot \delta_{i1}\, \delta_{i2} \ldots \delta_{im} \qquad (i = 1,2,\ldots,n) . \tag{1.6.7}$$

Define,

$$\left.\begin{aligned} \delta_{0j} &= 0 \\ n_j &= \sum_{i=1}^{n} \delta_{ij} \end{aligned}\right\} \qquad (j = 1,2,\ldots,m) , \tag{1.6.8}$$

$$\begin{aligned} n_0 &= 0 , \\ f_j &= n_j + 10 f_{j-1} \qquad (j = 1,2,\ldots,m) , \end{aligned} \tag{1.6.9}$$

and

$$f_0 = 0 .$$

Define the array $A(\cdot)$ by,

$$A(s) = i ,$$

if there exists a j, such that

$$\sum_{k=0}^{i-1} \delta_{k_j} < s - \sum_{l=0}^{j-1} n_l \le \sum_{k=0}^{i} \delta_{k_j} , \qquad (1.6.10)$$

for $s = 1,2,\ldots,\sum_{j=1}^{m} n_j$, and $i = 1,2,\ldots,n$.

Then the following algorithm generates variates from the n point distribution having probabilities $P_1,P_2,\ldots,P_n$.

Algorithm 1.11

1. $j = 1$.
2. If $f_j - n_j \le \langle 10^j R\rangle < f_j$ deliver $X = A(\sum_{l=0}^{j-1} n_l + \langle 10^j R\rangle - f_j + n_j + 1)$.
3. $j = j + 1$ and go to 2.

To verify the procedure, we must first show that the events

$$f_j - n_j \le \langle 10^j R\rangle < f_j \qquad (j = 1,2,\ldots,m) \qquad (1.6.11)$$

are mutually exclusive. Repeated application of (1.6.9) gives

$$f_j = n_j + 10n_{j-1} + \ldots + 10^{j-1}n_1 \quad . \qquad (1.6.12)$$

Thus (1.6.11) becomes

$$10^{-j}(10n_{j-1}+\ldots+10^{j-1}n_1) \le R < 10^{-j}(n_j+10n_{j-1}+\ldots+10^{j-1}n_1)$$

or

$$10^{-1}n_1 + \ldots + 10^{-(j-1)}n_{j-1} \le R < 10^{-1}n_1 + \ldots + 10^{-(j-1)}n_{j-1} + 10^{-j}n_j. \qquad (1.6.13)$$

As j runs over the integers 1 to m, we see that (1.6.13) and hence (1.6.11) generates non-overlapping intervals spanning the interval $[0,1)$.

Because of this mutual exclusivity, we have from step 2 of the algorithm that

$$P(X=i) = \sum_{j=1}^{m} P\{A(\sum_{l=0}^{j-1} n_l + \langle 10^j R\rangle - f_j + n_j + 1) = i\} , \qquad (1.6.14)$$

which in conjunction with (1.6.10) gives

$$P(X=i) = \sum_{j=1}^{m} P\{ \sum_{k=0}^{i-1} \delta_{k_j} + f_j - n_j < \langle 10^j R \rangle + 1 \leq \sum_{k=0}^{i} \delta_{k_j} + f_j - n_j \}. \qquad (1.6.15)$$

Since $\langle 10^j R \rangle + 1$ is uniformly distributed on the integers $\{1,2,\ldots,10^j\}$ (1.6.15) becomes

$$P(X=i) = \sum_{j=1}^{m} 10^{-j} \{ \sum_{k=0}^{i} \delta_{k_j} - \sum_{k=0}^{i-1} \delta_{k_j} \}$$

$$= \sum_{j=1}^{m} 10^{-j} \delta_{ij}$$

$$= 0.\delta_{i1}\delta_{i2} \cdots \delta_{im} ,$$

as required.

The main features of this algorithm are that it is fast, and has low storage requirements (compared to Marsaglia's method). However it requires some time to set up the array $A(\cdot)$. The reduction in storage requirements over Marsaglia's method can be significant. By way of an illustrative example, consider the Binomial distribution with parameters $n = 10$ and $p = 0.3$. Expressed to four decimal places, the probabilities are $p_0 = 0.0282$, $p_1 = 0.1211$, $p_2 = 0.2335$, $p_3 = 0.2668$, $p_4 = 0.2001$, $p_5 = 0.1030$, $p_6 = 0.0367$, $p_7 = 0.0090$, $p_8 = 0.0015$, $p_9 = 0.001$ and $p_{10} = 0.0000$. Using Marsaglia's method an array of size 10^4 is required, whereas the Norman and Cannon implementation requires $\sum_{j=1}^{4} n_j = 8 + 16 + 37 + 30 = 91$ storage locations.

1.6.5 Indexed Search

This is a refinement of the sequential search and is described by Chen and Asau (1974). The probability distribution is first divided into k equal parts ($k = 1,2,\ldots$). A Random Number is used to determine in which of the k intervals the generated value lies, then the selected interval is searched for the correct value.

Given a distribution with probabilities $(p_1,p_2,\ldots p_n)$ and cumulative probabilities $(q_1,q_2,\ldots,q_n)$, intervals (s_{j-1},s_j) $(j = 1,2,\ldots,k)$ are constructed where s_j is the smallest integer satisfying

$$q_{s_j} \geq \frac{j}{k}$$

and

$$s_0 = 1.$$

The following algorithm then generates Variates from the distribution:

Algorithm 1.12

1. generate $R \sim U(0,1)$.
2. $j = \langle kR + 1 \rangle$.
3. Find smallest $i \in (s_{j-1},s_j)$ such that $q_i \geq R$, and deliver $X = i$.

To illustrate the method, consider the ten-point distribution having cumulative probabilities; $q_1 = 0.03$, $q_2 = 0.17$, $q_3 = 0.48$, $q_4 = 0.65$, $q_5 = 0.85$, $q_6 = 0.95$, $q_7 = 0.97$, $q_8 = 0.98$, $q_9 = 0.99$, $q_{10} = 1.00$. If we employ $k = 5$ intervals, the boundaries are $s_0 = 1$, $s_1 = 3$, $s_2 = 3$, $s_3 = 4$, $s_4 = 5$, $s_5 = 10$. If, for example a random number 0.7234 is generated then $\langle kR+1 \rangle = \langle 4.6170 \rangle = 4$, directing the search to the interval $(s_3,s_4) = (4,5)$. The smallest i, such that $q_i \geq 0.7234$ necessarily lies in this interval and is 5. Thus $X = 5$ is delivered.

CHAPTER 2

METHODS OF GENERATION FROM SPECIFIC CONTINUOUS DISTRIBUTIONS

2.1 Negative Exponential

A well known method for generating standard Negative Exponential variates with p.d.f.

$$f_X(x) = e^{-x} \qquad (x \geq 0) \ , \tag{2.1.1}$$

is to set $X = -\ell n\ R$, where R is a random number.

An alternative method, due to von Neumann (1951), employs a comparison of random numbers only. Using the notation of section 1.3.4,

$$\left.\begin{aligned} \psi_j(x) &= e^{-(x-[j-1])}/(1-e^{-1}) \qquad (2.1.2) \\ h_j(x) &= x - [j-1] \qquad (2.1.3) \end{aligned}\right\} \ (j-1 \leq x < j) \ ,$$

$$p_j = e^{-(j-1)}(1-e^{-1}) \ , \tag{2.1.4}$$

for $j = 1,2,\ldots$. There is no necessity to calculate interval probabilities p_j, for it can be shown (see for example Fishman 1978, pp 400, 403) that p_j is precisely the probability that the j^{th} sequence of random number comparisons is the first to terminate in an odd value of N. Further, sampling from $\psi_j(x)$ is equivalent to sampling from $\psi_1(x)$ and adding $(j-1)$. The resulting method is to generate sequences of random numbers $\{R^{(k)}, R_1^{(k)}, R_2^{(k)} \ldots\}$, $(k = 1,2,\ldots)$, stopping at $k = j$, when $N^{(k)}$ is odd-valued, where

$$N^{(k)} = 1 \quad \text{if} \quad R^{(k)} < R_1^{(k)}, \tag{2.1.5}$$

and $N^{(k)} = n$ if $R^{(k)} \geq R_1^{(k)} \geq \ldots \geq R_{n-1}^{(k)} < R_n^{(k)}$, $(n \geq 2)$,

$X = j-1 + R^{(j)}$ is then delivered as the sample variate. The mean number of random numbers required is $(e^2/[e-1]) = 4.30$, and the

algorithm requires comparisons only. Nevertheless Ahrens and Dieter (1972) found a Fortran implementation of this (NE) about 50% slower than for the logarithmic method (LG). In fact they are confident enough to state that "... there is no method which can match LG in any high level language". In contrast an Assembler implementation of NE took $\frac{1}{3}$ of the Fortran time, but no corresponding results for LG were given.

A modification of this method due to Marsaglia (1961) samples from

$$\psi_1(x) = e^{-x}/(1-e^{-1}) \qquad (0 \le x \le 1), \tag{2.1.6}$$

by generating a random variable,

$$Y = \text{Min } (R_1, R_2, \ldots, R_M) ,$$

where M is a random variable having p.m.f.

$$f_M(m) = [(e-1)m!]^{-1} \qquad (m = 1,2,\ldots). \tag{2.1.7}$$

The sample variate is then delivered as $X = Y + j - 1$, where j is sampled from (2.1.4). The method involves look-up tables for M and j, and is clearly not as elegant as von Neumann's. Ahrens and Dieter's Fortran implementation (MA) was marginally slower than NE, while the Assembler implementation was slightly faster.

A modification by Ahrens and Dieter (SA) relies implicitly on generation from the p.d.f.,

$$f_X(x) = \ln 2 . e^{-x \ln 2} \qquad (x \ge 0) , \tag{2.1.8}$$

which may be represented as the probability mixture

$$f_X(x) = \sum_{j=1}^{\infty} p_j \, \psi_j(x) ,$$

where

$$p_j = 2^{-j} \tag{2.1.9}$$

and

$$\psi_j(x) = 2\ \ell n\ 2.2^{-(x-[j-1])} \qquad (j-1 \le x < j)\ , \qquad (2.1.10)$$

for $j = 1,2,\ldots$. For fixed j , sampling from $\psi_j(x)$ is performed by sampling from $\psi_1(x)$ and adding $(j-1)$ to the result. Sampling from (2.1.9) is performed by examining a sequence of independent bits taking value zero or one with equal probability. Ahrens and Dieter found that the Fortran implementation was approximately 10% slower than LG, while the Assembler implementation was faster than MA and NE.

MacLaren, Marsaglia and Bray (1964) represent the Exponential p.d.f. as

$$f_X(x) = \sum_{i=1}^{3} a_i\ g_i(x)\ , \qquad (2.1.11)$$

where

$$\left.\begin{aligned} g_1(x) &= a_1^{-1}\ e^{-(k+1)c} \\ g_2(x) &= a_2^{-1}(e^{-x} - e^{-(k+1)c}) \end{aligned}\right\} \quad (kc \le x < (k+1)c)\ ; \quad (k = 0,1,\ldots,\tfrac{4}{c} - 1)\ , \qquad (2.1.12)$$

$$g_3(x) = a_3^{-1}\ e^{-x} \qquad (4 \le x),$$

$$a_1 = \sum_{k=0}^{\frac{4}{c}-1} e^{-(k+1)c}$$

$$a_3 = e^{-4} \qquad (2.1.13)$$

$$a_2 = 1 - a_1 - a_3\ ,$$

and c is chosen such that $(4/c)$ is integer, $g_1(x)$ is a piecewise uniform distribution and sampling is via the generation of a discrete random variate. Sampling from $g_2(x)$, ("the wedge") is via a discrete random variate and a continuous random variate, the latter constructed from the minimum of Z random numbers where Z itself is a discrete random variable. Sampling from $g_3(x)$ is accomplished by adding 4 to an Exponential variate. The Exponential variate is obtained by

re-entering the whole algorithm (and this process continues until eventually sampling is effected from $g_1(x)$ or $g_2(x)$). The algorithm is "efficient" in the sense that when $c = \frac{1}{16}$ for example, a mean of only 1.11 random numbers is required. However it carries an overhead in computing constants and the associated storage.

In assessing the suitability of these algorithms, the available computational evidence suggests that for Fortran implementations, the logarithmic method is not substantially slower than the others. For programming in machine code, comparative results involving the logarithmic method are not readily available. In view of the simplicity, and the ability to generate high quality antithetic variates, with correlation -0.645 (Fishman 1973, pp 320-321), the logarithmic method has much to commend it.

2.2 Normal Distribution

One of the earliest and also most convenient methods is due to Box and Muller (1958). Given a pair of random numbers R_1 and R_2, a pair of independent Standard Normal deviates X_1 and X_2 may be obtained via,

$$X_1 = (-2 \ln R_1)^{\frac{1}{2}} \cos 2\pi R_2 ,$$
$$X_2 = (-2 \ln R_1)^{\frac{1}{2}} \sin 2\pi R_2 . \qquad (2.2.1)$$

The sine and cosine evaluations may be time consuming on some computers and so the "Polar Marsaglia" modification, Marsaglia and Bray (1964), is often preferred. Given two independent deviates $V_1, V_2 \sim U(-1,1)$, then $\tan^{-1}(V_2/V_1)$ and $V_1^2 + V_2^2$ are independently distributed as $U(0,2\pi)$ and $U(0,1)$ respectively, subject to $V_1^2 + V_2^2 \leq 1$. Subject

to this restriction on V_1 and V_2, (2.2.1) may be re-expressed as

$$X_1 = [-2\ \ell n(V_1^2+V_2^2)]^{\frac{1}{2}} V_1 (V_1^2+V_2^2)^{-\frac{1}{2}} ,$$
$$X_2 = [-2\ \ell n(V_1^2+V_2^2)]^{\frac{1}{2}} V_2 (V_1^2+V_2^2)^{-\frac{1}{2}} . \qquad (2.2.2)$$

Another early generation procedure is due to Batchelor and referred to by Tocher (1963, pp 27). Standard Normal deviates are obtained by generating variates from a folded standard Normal distribution,

$$f_X(x) = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} \qquad (x \geq 0) , \qquad (2.2.3)$$

and applying a random sign. (2.2.3) is expressed as a probability mixture,

$$f_X(x) = 0.6827\ f_1(x) + 0.3173\ f_2(x) , \qquad (2.2.4)$$

where

$$f_1(x) = \begin{cases} \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} / 0.6827 & (0 \leq x \leq 1) \\ 0 & (\text{elsewhere}) \end{cases} \qquad (2.2.5)$$

and

$$f_2(x) = \begin{cases} \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} / 0.3173 & (x \geq 1) \\ 0 & (\text{elsewhere}). \end{cases} \qquad (2.2.6)$$

Generation from $f_1(x)$ is via envelope rejection, using a uniform target distribution, with a sampling efficiency of $M_1^{-1} = 0.6827(\pi/2)^{\frac{1}{2}} = 0.8556$. Sampling from $f_2(x)$ is again via an envelope rejection method, using a target distribution,

$$g_2(x) = \begin{cases} 2e^{-2(x-1)} & (x \geq 1) \\ 0 & (\text{elsewhere}) . \end{cases} \qquad (2.2.7)$$

The sampling efficiency for the latter (0.7953) is obtained by noting that

$$M_2 = \text{Max}\left\{ \frac{f_2(x)}{g_2(x)} \right\} = \underset{x\geq 1}{\text{Max}} \left\{ \frac{e^{-\frac{1}{2}(x-2)^2}}{(2\pi)^{\frac{1}{2}}\ 0.3173} \right\} \qquad (2.2.8)$$

$$= 1.2573 ,$$

and the acceptance condition (given a random number R) is

$$R < e^{-\frac{1}{2}(X-2)^2} ,$$

or

$$E > \tfrac{1}{2}(X-2)^2 , \qquad (2.2.9)$$

where E is a standard Exponential variate.

Batchelor's method is of some historical interest, since it seems to be the first application of the envelope rejection method to sampling from the tail of a Normal distribution. Marsaglia (1964) provides an alternative and frequently used method to sample from the truncated Normal distribution,

$$f_X(x) = \begin{cases} Ae^{-\frac{1}{2}x^2} & (x \geq a > 0) \\ 0 & (\text{elsewhere}) . \end{cases} \qquad (2.2.10)$$

The method is to generate two random numbers R_1 and R_2 and to deliver

$$X = (a^2 - 2\ \ell n\ R_1)^{\frac{1}{2}} ,$$

subject to

$$R_2 < a(a^2 - 2\ \ell n\ R_1)^{-\frac{1}{2}} .$$

Marsaglia gives no proof, but clearly this may be interpreted as an envelope rejection procedure using a target distribution,

$$g_Y(x) = \begin{cases} xe^{-\frac{1}{2}(x^2-a^2)} & (x \geq a) \\ 0 & (\text{elsewhere}) . \end{cases} \qquad (2.2.11)$$

Thus,

$$f_X(x)/g_Y(x) = Ae^{-\frac{1}{2}a^2}/x \qquad (x \geq a),$$

$$M = Ae^{-\frac{1}{2}a^2}/a \ , \qquad (2.2.12)$$

and, (given a random number R_2) the acceptance condition becomes

$$R_2 < a/Y \ , \qquad (2.2.13)$$

where, using the inversion method on (2.2.11),

$$Y = (a^2 - 2\ \ell n\ R_1)^{\frac{1}{2}} \ . \qquad (2.2.14)$$

Another tail generation method (for values larger than 3) is mentioned in Marsaglia, MacLaren and Bray (1964). If X_1 and X_2 are independent standard Normal deviates, define

$$R^2 = X_1^2 + X_2^2 \qquad \text{and} \qquad \Theta = \tan^{-1}(X_2|X_1) \ . \qquad (2.2.15)$$

The joint distribution of R and Θ, conditional upon $R \geq 3$ and $0 \leq \Theta \leq \frac{\pi}{2}$ is

$$f_{R|R\geq 3,\ 0\leq\Theta\leq\frac{\pi}{2}}(r,\theta) = \left\{\frac{2r}{\pi}\right\} e^{-\frac{1}{2}(r^2-9)} \qquad (2.2.16)$$

for $r \geq 3$ and $0 \leq \theta \leq \frac{\pi}{2}$. Thus the conditioned variates R and Θ are independently distributed. Using the inversion method on the marginal distributions,

$$R = (9 - 2\ \ell n\ R_1)^{\frac{1}{2}} \qquad (2.2.17)$$

and

$$\Theta = \left\{\frac{\pi}{2}\right\} R_2 \ , \qquad (2.2.18)$$

where R_1 and R_2 are two random numbers. Inverting (2.2.15),

$$\begin{aligned} X_1 &= (9-2\ \ell n\ R_1)^{\frac{1}{2}}\cos(R_2\pi/2) \ , \\ X_2 &= (9-2\ \ell n\ R_1)^{\frac{1}{2}}\sin(R_2\pi/2) \ , \end{aligned} \qquad (2.2.19)$$

give variates X_1 and X_2 subject to $X_1^2 + X_2^2 \geq 9$, $X_1 \geq 0$ and $X_2 \geq 0$. Sine and cosine evaluation may be avoided by noting that, given two random numbers V_1 and V_2, subject to $V_1^2 + V_2^2 \leq 1$,

$V_1^2 + V_2^2 \sim U(0,1)$, $\tan^{-1}(V_2/V_1) \sim U(0,\pi/2)$, and are independently distributed. Thús subject to $V_1^2 + V_2^2 \leq 1$, (2.2.19) may be replaced by

$$X_1 = [9 - 2\,\ell n(V_1^2+V_2^2)]^{\frac{1}{2}} V_1 (V_1^2+V_2^2)^{-\frac{1}{2}} ,$$
$$X_2 = [9 - 2\,\ell n(V_1^2+V_2^2)]^{\frac{1}{2}} V_2 (V_1^2+V_2^2)^{-\frac{1}{2}} . \qquad (2.2.20)$$

With probability 0.49, at least one of the generated variates will have values exceeding 3 and thus may be used as a tail value.

Another convenient method is described in Fishman (1978, pp 413). Using envelope rejection, Variates are sampled from a folded Normal distribution via an Exponential target distribution,

$$g_Y(x) = e^{-x} \qquad (x \geq 0) .$$

In this case,

$$\frac{f_X(x)}{g_Y(x)} = \left(\frac{2e}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}(x-1)^2} , \qquad (2.2.21)$$

and

$$M = \left(\frac{2e}{\pi}\right)^{\frac{1}{2}} = 1.315 . \qquad (2.2.22)$$

Thus the sampling efficiency, $M^{-1} = 0.760$, and the acceptance condition (given a random number R) is

$$R < e^{-\frac{1}{2}(Y-1)^2}$$

or

$$E_1 > \tfrac{1}{2}(E_2-1)^2 , \qquad (2.2.23)$$

where E_1 and E_2 are independent standard Exponential variates.

Payne (1977) describes an elegant algorithm using the Band rejection technique. The folded Normal distribution $f_W(\cdot)$ is expressed as the probability mixture,

$$f_W(x) = 0.9545\, f_X(x) + 0.0455\, f_Z(x) , \qquad (2.2.24)$$

where

$$f_X(x) = \begin{cases} \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2}/0.9545 & (0 \le x \le 2) \\ 0 & (\text{elsewhere}) , \end{cases}$$

and

$$f_Z(x) = \begin{cases} \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2}/0.0455 & (x \ge 2) \\ 0 & (\text{elsewhere}) . \end{cases}$$

Sampling from $f_Z(\cdot)$ is via Marsaglia's tail generation procedure (2.2.13)-(2.2.14) with $a = 2$ (sampling efficiency 0.8425, not 0.8409 as given by Payne). Sampling from $f_X(\cdot)$ uses the Band rejection technique described in section 1.3.2 . Using the notation of algorithm 1.4, the target distribution is

$$g_Y(x) = \begin{cases} 0.5 & (0 \le x \le 2) \\ 0 & (\text{elsewhere}) , \end{cases} \qquad (2.2.25)$$

$$\frac{f_X(x)}{g_Y(x)} = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2}/(0.5 \times 0.9545) , \qquad (2.2.26)$$

$$\frac{f_X(x) + f_X(2-x)}{g_Y(x)} = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} \left\{ \frac{e^{-\frac{1}{2}x^2} + e^{-\frac{1}{2}(x-2)^2}}{0.5 \times 0.9545} \right\} , \qquad (2.2.27)$$

and

$$M = \left(\frac{2}{\pi e}\right)^{\frac{1}{2}} \cdot \left(\frac{4}{0.9545}\right) = 2.028 , \qquad (2.2.28)$$

giving a sampling efficiency of $2M^{-1} = 0.9862$, for X. The primary acceptance condition (step 2) is

$$R < f_X(Y)/Mg_Y(Y) = \tfrac{1}{2}e^{-\frac{1}{2}(Y^2-1)} . \qquad (2.2.29)$$

If this is satisfied then $X = Y$ is delivered. If not, the secondary acceptance condition (step 3) applies and is

$$R < \frac{f_X(Y) + f_X(2-Y)}{M\, g_Y(Y)} = \left[\frac{e^{\frac{1}{2}}}{2}\right]\left(e^{-\frac{1}{2}Y^2} + e^{-\frac{1}{2}(Y-2)^2}\right) . \qquad (2.2.30)$$

If this is satisfied, then $X = 2-Y$ is delivered. The overall sampling efficiency is $0.9545(0.9862) + 0.0455(0.8425) = 0.9797$.

Kinderman and Monahan (1977) employ the ratio of uniform variates method. Using the notation of section 1.3.3, consider a region,

$$C = \{(u,v) : 0 \le u \le (2\pi)^{-\frac{1}{4}} e^{-v^2/4u^2}\}$$

$$= \{(u,v) : 0 \le u \le (2\pi)^{-\frac{1}{4}};\ |v| \le u[-\ell n(2\pi u^4)]^{\frac{1}{2}}\}. \quad (2.2.31)$$

If (U,V) is uniformly distributed over C, then the p.d.f. of V/U is standard Normal. Kinderman and Monahan give no specific details of how to generate points uniformly over C. However one obvious method is to generate points uniformly over the enclosing rectangle,

$$D = \{(u,v) : 0 \le u \le (2\pi)^{-\frac{1}{4}}\ ;\ |v| \le (2\pi)^{-\frac{1}{4}}(2/e)^{\frac{1}{2}}\}\ , \quad (2.2.32)$$

and reject those that do not satisfy the condition,

$$|V| \le U\{-\ell n(2\pi U^4)\}^{\frac{1}{2}}\ . \quad (2.2.33)$$

The area of D is $(4/\pi e)^{\frac{1}{2}}$ and thus the sampling efficiency, being the ratio of the two areas $C : D$ is $(\frac{1}{2})/(4/\pi e)^{\frac{1}{2}} = 0.731$. Robertson and Walls (1980) discuss various methods of improving the efficiency of the above scheme, including a pre-test which reduces the number of logarithmic evaluations. They also consider as an alternative to a rectangle, a tighter fitting trapezium-shaped region, which increases the sampling efficiency to 0.922 .

The algorithms dealt with so far are attractive in that they are all easy to program and require little storage. This is an important feature when pre-programmed routines do not exist, or when an experimenter needs a method which matches the experimental design configuration for the simulation. The methods to be considered below are generally less elegant, less easy to program, require more

storage, but in many cases have lower execution times (excluding time required to calculate constants and set-up tables. Given the characteristics of these methods, it is still felt that ease of programming is an important feature and that one should be willing to trade off some efficiency for this. In particular, the rapid change in computer technology has resulted, and will probably continue to result in substantial increases in processing speed for a given hardware cost. In view of this, the fact that one algorithm happens to be slightly faster than another on a particular machine does not automatically favour the former.

One of the early algorithms that fall into this second category is the Rectangle-Wedge-Tail method of Marsaglia, MacLaren and Bray (1964). The folded Normal distribution is expressed as a probability mixture,

$$f_X(x) = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} = \sum_{i=1}^{3} a_i g_i(x) , \qquad (2.2.34)$$

where

$$\left.\begin{aligned}
a_1 &= 0.1 \sum_{j=1}^{30} f_X(0.1j) = 0.9578 , \\
a_2 &= \int_3^{\infty} f_X(x)dx = 0.0395 , \\
a_3 &= 1 - a_1 - a_2 = 0.0027 , \\
g_1(x) &= a_1^{-1} \begin{cases} f_X(0.1) & (0 \le x \le 0.1) \\ f_X(0.2) & (0.1 < x \le 0.2) \\ \vdots & \\ f_X(3.0) & (2.9 < x \le 3.0) \\ 0 & (\text{elsewhere}) \end{cases} \\
g_2(x) &= \begin{cases} \{f_X(x) - a_1 g_1(x)\}/a_2 & (0 < x \le 3.0) \\ 0 & (\text{elsewhere}) , \end{cases} \\
g_3(x) &= \begin{cases} f_X(x)/a_3 & (x > 3) \\ 0 & (\text{elsewhere}) . \end{cases}
\end{aligned}\right\} \qquad (2.2.35)$$

$g_1(x)$ is piecewise uniform and sampling is via an efficient look-up method, based on generation of a discrete random variate. Sampling from $g_2(x)$ (the "wedge") is via envelope rejection utilising the approximate piecewise linearity of the function, and sampling from $g_3(x)$ uses the polar tail method, (2.2.20).

Marsaglia and Bray (1964) in their "convenient" method represent the Normal p.d.f. as the probability mixture,

$$f_X(x) = (2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}x^2} = \sum_{i=1}^{4} a_i g_i(x) \ , \qquad (2.2.36)$$

where

$$\left.\begin{aligned}
&g_1(x) \equiv \text{p.d.f. of } \ 2(R_1+R_2+R_3-1.5) \ \text{ where} \\
&\qquad R_1, \ R_2 \ \text{ and } \ R_3 \ \text{ are random numbers,} \\
&g_2(x) \equiv \text{p.d.f. of } \ 1.5(R_1+R_2-1) \ , \\
&g_3(x) = \begin{cases} \left(f_X(x) - a_1 g_1(x) - a_2 g_2(x)\right)/a_3 & (|x| \le 3) \\ 0 & \text{(elsewhere),} \end{cases} \\
&g_4(x) = \begin{cases} f_X(x)/a_4 & (|x| > 3) \\ 0 & \text{(elsewhere)} \ , \end{cases} \\
&a_1 = 0.8638 \ , \\
&a_2 = 0.1107 \ , \\
&a_3 = 0.0228002039 \ , \\
&a_4 = 1 - a_1 - a_2 - a_3 = 0.0026997961 \ .
\end{aligned}\right\} \qquad (2.2.37)$$

Sampling from $g_3(x)$ uses envelope rejection with a uniform target distribution. Sampling from $g_4(x)$ uses the polar method, (2.2.20).

Ahrens and Dieter (1972) use similar ideas in their trapezoidal method, (Algorithm TR). The Normal p.d.f. is represented as the probability mixture,

$$f_X(x) = \sum_{i=1}^{3} a_i g_i(x) \ , \tag{2.2.38}$$

where

$$\left.\begin{aligned}
&g_1(x) \ \text{has the shape of an isosceles trapezium inscribed in the curve } f_X(x) \ \text{for } |x| < 2.7140280 \ , \\
&g_2(x) = \{f_X(x) - a_1 g_1(x)\}/a_2 \qquad (|x| < 2.7140280) \ , \\
&g_3(x) = f_X(x)/a_3 \qquad (|x| \geq 2.7140280) \ , \\
&a_1 = 0.9195444 \ , \\
&a_2 = 0.0459427 \ , \\
&a_3 = 0.0345129 \ .
\end{aligned}\right\} \tag{2.2.39}$$

Sampling from $g_1(x)$ is via a linear combination of two random numbers, from $g_3(x)$ via the tail generation procedure of Marsaglia, (2.2.13)-(2.2.14), and from $g_2(x)$ via an envelope rejection procedure.

Polynomial sampling figures in algorithm TS due to Ahrens and Dieter (1972). They represent the folded Normal p.d.f. as

$$f_X(x) = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}x^2} = \sum_{i=1}^{5} a_i g_i(x) \ , \tag{2.2.40}$$

where

$$\left.\begin{aligned}
&a_1 = \tfrac{1}{2}, \ a_2 = \tfrac{1}{4}, \ a_3 = \tfrac{1}{8}, \ a_4 = \tfrac{1}{16}, \ a_5 = \tfrac{1}{16}, \\
&g_1(x) = \begin{cases} f_X(x)/a_1 & (0 < x \leq 0.67449) \\ 0 & (\text{elsewhere}), \end{cases} \\
&g_2(x) = \begin{cases} f_X(x)/a_2 & (0.67449 < x \leq 1.1503) \\ 0 & (\text{elsewhere}) \ , \end{cases} \\
&g_3(x) = \begin{cases} f_X(x)/a_3 & (1.1503 \leq x \leq 1.5341) \\ 0 & (\text{elsewhere}) \ , \end{cases} \\
&g_4(x) = \begin{cases} f_X(x)/a_4 & (1.534 \leq x \leq 1.8627) \\ 0 & (\text{elsewhere}) \ , \end{cases} \\
&g_5(x) = \begin{cases} f_X(x)/a_5 & (x \geq 1.8627) \\ 0 & (\text{elsewhere}) \ . \end{cases}
\end{aligned}\right\} \tag{2.2.41}$$

The p.d.f.'s $g_i(x)$ $(i = 1,2,3,4)$ are approximated by Taylor series expansions,

$$g_i(x) \doteq \sum_{j=0}^{k} b_j^{(i)} x^j \ , \qquad (2.2.42)$$

where k is of the order of 20. The degree of the polynomial ensures that the approximation is extremely good. Sampling from (2.2.42) uses the polynomial sampling method of section 1.5 . Although k is of the order of 20, only on a very small proportion of occasions will anything approaching k random numbers be required, since the $b_j^{(i)}$ become small as j increases. Sampling from $g_5(x)$ uses the tail method, (2.2.13)-(2.2.14). Ahrens and Dieter (1972) also use polynomial sampling in algorithm RT, to modify the rectangle-wedge-tail algorithm.

Kinderman and Ramage (1976) use a probability mixture method, representing the standard Normal p.d.f. as

$$f_X(x) = (2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}x^2} = \sum_{i=1}^{3} a_i g_i(x) \ , \qquad (2.2.43)$$

where

$$\left.\begin{aligned} g_1(x) &= (2.216 - |x|)/(2.216)^2 && (|x| < 2.216), \\ g_2(x) &= \{f_X(x) - a_1 g_1|x|\}/a_2 && (|x| < 2.216), \\ g_3(x) &= f_X(x)/a_3 && (|x| \geq 2.216), \\ a_1 &= 0.884 \ , \\ a_2 &= 0.089 \ , \\ a_3 &= 0.027 \ . \end{aligned}\right\} \qquad (2.2.44)$$

The inversion method is used to sample from $g_1(x)$, the tail method (2.2.13)-(2.2.14) from $g_3(x)$, and an envelope rejection method comprising a series of triangular target distributions from $g_2(x)$.

A number of methods have arisen from Forsythe's generalisation of von Neumann's combinatorial method for Exponential variate generation. Using the notation of section 1.3.4, the main methods are:

(a) Forsythe (1972). Intervals are defined by:

$$q_0 = 0 \ , \quad q_j = (2j-1)^{\frac{1}{2}} \qquad (j \geq 1) \ , \tag{2.2.45}$$

each variate requiring a mean of 3.036 random numbers (excluding the one used for interval selection).

(b) Parker (1976) has suggested intervals defined by:

$$q_0 = 0 \ , \quad q_j = (2j)^{\frac{1}{2}} \qquad (j \geq 1) \ . \tag{2.2.46}$$

(c) Dieter and Ahrens (1973) in their "centre-tail" method use one interval defined by:

$$q_0 = 0 \ , \quad q_1 = \sqrt{2} \ . \tag{2.2.47}$$

Sampling from the tail $(\sqrt{2},\infty)$ uses Marsaglia's tail procedure. They claim a mean requirement of 4.879 random numbers.

(d) Ahrens and Dieter (1973), (algorithm FT), use intervals defined by:

$$q_j = R^{-1}(2^{-j}) \qquad (j \geq 0) \ , \tag{2.2.48}$$

where

$$R(x) = \int_x^\infty \left(\frac{2}{\pi}\right)^{\frac{1}{2}} e^{-\frac{1}{2}u^2} \, du \ .$$

The mean requirement is 2.539 random numbers, excluding one used for interval selection.

(e) Ahrens and Dieter (1973), (algorithm FL) use intervals defined by:

$$q_0 = 0 \ , \quad q_1 = R^{-1}(31/32), \quad q_2 = R^{-1}(30/32) \ \ldots, \ q_{31} = R^{-1}(1/32),$$
$$q_{32} = R^{-1}(1/64), \quad q_{33} = R^{-1}(1/128) \ \ldots \ . \tag{2.2.49}$$

Because the intervals are of small width, the mean requirement of random numbers would normally be close to 2. Referring to (1.3.24), this figure is reduced further by comparing R_1, not with $h_j(X_j)$, but with $h_j^* = \text{Max}[h_j(X_j)]$. If $R_1 > h_j^*$ (as it is on most occasions) then $X_j \sim U(q_{j-1}, q_j)$ is delivered. In this case X_j may be generated by "conditioning" R_1, giving

$$X_j = q_{j-1} + \left(\frac{R_1 - h_j^*}{1 - h_j^*}\right) \cdot (q_j - q_{j-1}) \ , \qquad (2.2.50)$$

without recourse to a separate random number. In this way the mean requirement is reduced to 1.232 random numbers.

(f) Brent (1974) has devised a method which combines the intervals defined in (d) with the "conditioning" technique of (e), giving a mean requirement of 1.37 random numbers.

2.3 Gamma Distribution

The Gamma distribution with shape parameter α (> 0) and scale parameter λ (> 0) has p.d.f.

$$f_Z(x) = \frac{\lambda^\alpha x^{\alpha-1} e^{-\lambda x}}{\Gamma(\alpha)} \qquad (x \geq 0). \qquad (2.3.1)$$

Variates may be generated by sampling from a standard Gamma distribution having p.d.f.

$$f_X(x) = \frac{x^{\alpha-1} e^{-x}}{\Gamma(\alpha)} \qquad (x \geq 0), \qquad (2.3.2)$$

and setting $Z = X/\lambda$.

When α is integer-valued, a simple method is to deliver X as the sum of α independent standard Exponential variates. In this connection, Greenwood (1974) emphasises the danger of taking the logarithm of the product of α random numbers. Our view is that the approach is probably safe when α is not very large. Taking a machine with a hypothetical floating point underflow at 10^{-37}, the procedure will attempt to compute the logarithm of zero if the true variate value exceeds $-\ell n(10^{-37}) = 85.2$. This difficulty may be eliminated by continuously checking the cumulative product of the random numbers to ascertain whether floating point underflow is possible on the next product. If it is, then the logarithm of the cumulative product must be taken, and the remainder of the α random numbers must have their logarithms taken individually. For values of α where these difficulties are likely to arise, the method is very unlikely to be used, due to the high requirement of random numbers.

For $\alpha < 1$, Johnk (1964) sets $X = EY$, where E is a standard Exponential variate and Y a Beta variate with parameters α and $(1-\alpha)$. The latter is obtained by delivering

$$Y = R_1^{1/\alpha}/(R_1^{1/\alpha} + R_2^{1/[1-\alpha]}) \ , \qquad (2.3.3)$$

subject to $R_1^{1/\alpha} + R_2^{1/(1-\alpha)} \leq 1$, where R_1 and $R_2 \sim U(0,1)$. The probability of acceptance never falls below $0.25\pi = 0.785$. The algorithm may be extended to cases where $\alpha > 1$, by expressing the shape parameter as

$$\alpha = <\alpha> + (\alpha - <\alpha>) \ , \qquad (2.3.4)$$

generating from a Gamma distribution with shape parameter $\alpha - <\alpha>$ using Johnk's method, and from a Gamma distribution with shape parameter $<\alpha>$, using the sum of exponentials method.

For $\alpha < 1$ another procedure is due to McGrath and Irving (1973), who use envelope rejection with a target distribution,

$$g_Y(x) = \begin{cases} p\,\alpha\, x^{\alpha-1}/\gamma^{\alpha} & (0 \le x \le \gamma) \\ (1-p)e^{-(x-\gamma)} & (x > \gamma), \end{cases} \tag{2.3.5}$$

where $0 \le p \le 1$ and γ are free parameters. This gives

$$\frac{f_X(x)}{g_Y(x)} = \begin{cases} e^{-x}\gamma^{\alpha}/\{p\Gamma(\alpha+1)\} & (0 \le x \le \gamma) \\ x^{\alpha-1}e^{-\gamma}/\{(1-p)\Gamma(\alpha)\} & (x > \gamma), \end{cases} \tag{2.3.6}$$

with maximum,

$$M = \mathrm{Max}[\gamma^{\alpha}/\{p\Gamma(\alpha+1)\},\ \gamma^{\alpha-1}e^{-\gamma}/\{(1-p)\Gamma(\alpha)\}] \ . \tag{2.3.7}$$

By choosing $p = \gamma/(\gamma + \alpha e^{-\gamma})$, both terms in the argument of (2.3.7) become equal, giving a sampling efficiency of

$$M^{-1} = \frac{\gamma^{1-\alpha}\Gamma(\alpha+1)}{(\gamma + \alpha e^{-\gamma})} \ . \tag{2.3.8}$$

McGrath and Irving set $\gamma = 1$, which interestingly gives exactly the same method as the much quoted GS algorithm of Ahrens and Dieter (1974). Fishman (1978, pp 423) indicates that M^{-1} may be maximised by allowing γ to satisfy

$$\gamma(e^{\gamma} - 1) = 1 - \alpha \ . \tag{2.3.9}$$

To avoid numerical effect required to solve (2.3.9) Fishman suggests uses of a near optimal value of $\gamma = 0.5$. Independently, the author has devised an algorithm using $\gamma = 1 - \alpha$. Table 2.1 shows sampling efficiencies for these four choices for the basic McGrath and Irving algorithm. If numerical optimisation is to be avoided, the choice lies between $\gamma = 1$ (Ahrens and Dieter), $\gamma = 0.5$ (Fishman) and $\gamma = 1-\alpha$ (Dagpunar). We note that results for $\gamma = 1-\alpha$, but not $\gamma = 0.5$ dominate those for $\gamma = 1$, for all values of α . Further the performance of $\gamma = 1-\alpha$ is significantly better than the other

two when α is very close to 1. We conclude therefore that the choice $\gamma = 1-\alpha$ is likely to be preferred to the other two.

Table 2.1 Sampling Efficiencies for McGrath and Irving method using optimal γ, $\gamma = 0.5$, $\gamma = 1$, $\gamma = 1-\alpha$.

Sampling Efficiency \ α	0.01	0.1	0.3	0.5	0.7	0.8	0.9	1.0
optimal γ (Fishman)	0.991	0.921	0.824	0.785	0.798	0.828	0.882	1.000
$\gamma = 0.5$ (Fishman)	0.990	0.909	0.810	0.780	0.798	0.823	0.858	0.904
$\gamma = 1$ (Ahrens & Dieter)	0.991	0.918	0.808	0.749	0.723	0.720	0.723	0.731
$\gamma = 1-\alpha$ (Dagpunar)	0.991	0.920	0.824	0.780	0.774	0.790	0.836	1.000

Algorithm 2.1 below specifies details of a method based on $\gamma = 1-\alpha$, which has been used in a simulation of a telephone answering service, Gregory (1979).

Algorithm 2.1

0. $A = 1-\alpha$; $p = A/(A+\alpha e^{-A})$; $B = A + \ell n(1-p)$; $C = 1/\alpha$.

1. generate a random number R, and a standard exponential variate E.
2. If $R > p$ go to 5.
3. $X = A(R/p)^{C}$.
4. If $E > X$ deliver X, else go to 1.
5. $X = B - \ell n(1-R)$.
6. If $E > A\,\ell n(X/A)$ deliver X, else go to 1.

The frequency of logarithmic evaluations could be reduced by inserting appropriate acceptance pretests into steps 4 and 6.

When $\alpha \geq 1$, Ahrens and Dieter (1974) propose algorithm GC, using envelope rejection with a Cauchy target distribution,

$$h_Y(x) = \beta/[\pi\{\beta^2 + (x-\delta)^2\}] \ , \qquad (2.3.10)$$

where β and δ are constants. As Fishman (1978, pp 427) implies, it is better to use a truncated Cauchy distribution,

$$g_Y(x) = A\beta/[\pi\{\beta^2 + (x-\delta)^2\}] \qquad (x \geq 0) \ , \qquad (2.3.11)$$

where

$$A^{-1} = \{\pi + 2\tan^{-1}(\delta/\beta)\}/(2\pi). \qquad (2.3.12)$$

In this case,

$$\frac{f_X(x)}{g_Y(x)} = \frac{x^{\alpha-1}e^{-x}\pi\{\beta^2 + (x-\delta)^2\}}{A\ \beta\ \Gamma(\alpha)} \ , \qquad (2.3.13)$$

which has turning points at

$$\left\{\frac{\alpha - 1}{x}\right\} - 1 + \frac{2(x-\delta)}{\beta^2 + (x-\delta)^2} = 0 \ ,$$

or

$$(\alpha-1-x)\{B^2 + (x-\delta)^2\} + 2x(x-\delta) = 0 \ . \qquad (2.3.14)$$

Parameters β and δ are chosen as

$$\beta = (2\alpha-1)^{\frac{1}{2}} \quad \text{and} \quad \delta = \alpha - 1 \ , \qquad (2.3.15)$$

so the turning points appear at

$$(\alpha-1-x)(x-\alpha)^2 = 0 \ . \qquad (2.3.16)$$

Investigation of these shows that $x = \alpha - 1$ is a maximum for $\alpha \geq 1$, and $x = \alpha$ a point of inflection. Thus the maximum of (2.3.13) lies at $x = \alpha - 1$, giving

$$M = (\alpha-1)^{\alpha-1}e^{-(\alpha-1)}\pi(2\alpha-1)^{\frac{1}{2}}/\{A\ \Gamma(\alpha)\} \ ,$$

or

$$M = \frac{(\alpha-1)^{\alpha-1}e^{-(\alpha-1)}(2\alpha-1)^{\frac{1}{2}}[\pi + 2\tan^{-1}\{(\alpha-1)/(2\alpha-1)^{\frac{1}{2}}\}]}{2\Gamma(\alpha)} \quad . \qquad (2.3.17)$$

When $\alpha = 1$, $M = \pi/2$, and use of Stirling's formula shows that as $\alpha \to \infty$, $M \to \sqrt{\pi} = 1.77$, giving the useful property that the sampling efficiency is bounded for all $\alpha \geq 1$.

Also described in the same paper is algorithm GO. Envelope rejection is used, the target distribution being Exponential in the tail, and Normal in the body of the distribution. The sampling efficiency is bounded as $\alpha \to \infty$. Using pre-tests on the acceptance condition, it is generally accepted to be a fast method of generating Gamma variates, providing a cheap source of Normal deviates is available. Unfortunately the method is only valid for shape parameters $\alpha > 2.53$. A development from Wallace (1974) uses envelope rejection with a target distribution consisting of a mixture of Erlang distributions with shape parameters $<\alpha>$ and $<\alpha+1>$ respectively. Thus

$$g_Y(x) = px^{<\alpha>-1}e^{-x}/\Gamma(<\alpha>) + (1-p)x^{<\alpha>}e^{-x}/\Gamma(<\alpha+1>), \qquad (2.3.18)$$

where $0 \leq p \leq 1$. Hence,

$$\frac{g_Y(x)}{f_X(x)} = \frac{px^{<\alpha>-\alpha}\Gamma(\alpha)}{\Gamma(<\alpha>)} + \frac{(1-p)x^{<\alpha>-\alpha+1}\Gamma(\alpha)}{\Gamma(<\alpha+1>)} \quad , \qquad (2.3.19)$$

which is minimised at

$$x = \frac{p(\alpha-<\alpha>)<\alpha+1>}{(1-p)(1-\alpha+<\alpha>)} \quad . \qquad (2.3.20)$$

By choosing $p = 1 - \alpha + <\alpha>$, this conveniently reduces to $x = <\alpha+1>$, giving a sampling efficiency of

$$M^{-1} = <\alpha+1>^{<\alpha>-\alpha}\Gamma(\alpha+1)/\Gamma(<\alpha+1>) \quad . \qquad (2.3.21)$$

Although this is close to one for most values of α, the procedure is inefficient for large α, due to the time required to generate the associated Erlang variate.

Fishman (1976a) proposes a simple envelope rejection method for $\alpha \geq 1$, using an Exponential target distribution with mean α. Thus

$$g_Y(x) = \alpha^{-1} e^{-x/\alpha} \qquad (x \geq 0) \ , \qquad (2.3.22)$$

$$\frac{f_X(x)}{g_Y(x)} = \frac{\alpha x^{\alpha-1} e^{-x(1-\frac{1}{\alpha})}}{\Gamma(\alpha)} \ , \qquad (2.3.23)$$

with maximum value,

$$M = e^{-(\alpha-1)} \alpha^{\alpha} / \Gamma(\alpha) \ . \qquad (2.3.24)$$

As $\alpha \to \infty$, M behaves as $\alpha^{\frac{1}{2}}$, and so the method is not well suited to large values of α.

Greenwood (1974) exploits the Wilson-Hilferty (1936) approximation to the chi-squared distribution with ν degrees of freedom. This takes the form

$$\chi^2_\nu \doteq \nu\{1 - (2/9\nu)^{-1} + (2/9\nu)^{-\frac{1}{2}} Z\} \ , \qquad (2.3.25)$$

where $Z \sim N(0,1)$. Since $0.5\chi^2_\nu$ is a standard Gamma variate with shape parameter $(\nu/2)$, envelope rejection is suggested using a target variate Y, where

$$Y = \alpha\{1 - (1/9\alpha)^{-1} + (1/9\alpha)^{-\frac{1}{2}} Z\} \ . \qquad (2.3.26)$$

The maximum value of $f_X(x)/g_Y(x)$ can be obtained analytically and is shown to be 2.189 for $\alpha = \frac{1}{3}$, 1.337 for $\alpha = \frac{1}{2}$, approaching an asymptotic value of 1 as $\alpha \to \infty$.

Cheng (1977) also uses envelope rejection, this time with a log-logistic target distribution,

$$g_Y(x) = \lambda\mu\, x^{\lambda-1}/(\mu+x^{\lambda})^2 \qquad (x \geq 0)\ , \qquad (2.3.27)$$

where $\mu = \alpha^{\lambda}$ and $\lambda = (2\alpha-1)^{\frac{1}{2}}$ for $\alpha \geq 1$, and $\lambda = \alpha$ for $0 < \alpha < 1$. The sampling efficiency increases from $(e/4)$ at $\alpha = 1$ to an asymptotic value of $\sqrt{\pi}/2$ as $\alpha \to \infty$. For $\alpha < 1$, the method is not so suitable, the sampling efficiency decreasing to 0.25 as $\alpha \to 0$.

Atkinson (1977) uses envelope rejection for $\alpha \geq 1$, with a target distribution,

$$g_Y(x) = \begin{cases} k(1-\mu)^t & (x \leq t) \\ ke^{-\mu x} & (x > t)\ , \end{cases} \qquad (2.3.28)$$

where $k^{-1} = t(1-\mu)^t + \mu^{-1}e^{-\mu t}$, with t and μ (<1) being free parameters. t is set to $\alpha - 1$, allowing M to be conveniently evaluated as

$$M = (\alpha-1)^{\alpha-1}e^{-(\alpha-1)}/\{k\Gamma(\alpha)(1-\mu)^{\alpha-1}\}, \qquad (2.3.29)$$

and by choosing

$$\mu = \{(4\alpha-3)^{\frac{1}{2}} - 1\}/\{2(\alpha-1)\}\ ,$$

the sampling efficiency is maximised. Atkinson reports that the method is slightly faster than Cheng's (1977) method for the range $1.5 \leq \alpha \leq 3.5$. For large α, we remark that $M \doteq \sqrt{\{(\alpha-1)/2\pi\}}$, indicating that the method is not so suitable for such values. Finally we note that both Cheng's and Atkinson's methods incorporate pre-tests which avoid logarithmic evaluation on some occasions.

Another envelope rejection method (for $\alpha \geq 1$) is due to Tadikamalla (1978a). The target distribution is Erlang, with shape parameter $<\alpha>$, having the same mean as that of the Gamma. Thus

$$g_Y(x) = \lambda^{<\alpha>}x^{<\alpha-1>}e^{-\lambda x}/\Gamma(<\alpha>)\ , \qquad (2.3.30)$$

where

$$\lambda = <\alpha>/\alpha\ ,$$

giving

$$\frac{f_X(x)}{g_Y(x)} = \frac{x^{\alpha-<\alpha>} e^{-x(1-\lambda)} \Gamma(<\alpha>)}{\lambda^{<\alpha>} \Gamma(\alpha)} \quad ,$$

which has a maximum at $x = \alpha$. Thus

$$M = \alpha^{\alpha-<\alpha>} e^{-\alpha+<\alpha>} \Gamma(<\alpha>)/\{\lambda^{<\alpha>} \Gamma(\alpha)\} \quad . \qquad (2.3.31)$$

The theoretical efficiencies are good, e.g. 0.795 for $\alpha = 1.5$, 0.930 for $\alpha = 2.25$, but the algorithm is not efficient for large α , due to the time required for generation of the Erlang variate.

Tadikamalla (1978b) suggests a Laplace p.d.f.,

$$g_Y(x) = \frac{\lambda e^{-\lambda|x-\alpha+1|}}{2 - e^{-\lambda(\alpha-1)}} \qquad (x \geq 0,\ 0 < \lambda < 1) \ , \qquad (2.3.32)$$

as an alternative target distribution for $\alpha \geq 1$. Proofs of the sampling efficiency were not given and so are derived below. Given (2.3.32), we have,

$$H(x) = \frac{f_X(x)}{g_Y(x)} = \begin{cases} \dfrac{x^{\alpha-1} e^{-x(1+\lambda)} e^{\lambda(\alpha-1)} (2-e^{-\lambda(\alpha-1)})}{\lambda\Gamma(\alpha)} & (x \leq \alpha-1) \\ \dfrac{x^{\alpha-1} e^{-x(1-\lambda)} e^{-\lambda(\alpha-1)} (2-e^{-\lambda(\alpha-1)})}{\lambda\Gamma(\alpha)} & (x \geq \alpha-1), \end{cases} \qquad (2.3.33)$$

which has stationary values (both maxima) at

$$x_1 = \frac{\alpha - 1}{1 + \lambda} \qquad \text{and} \qquad x_2 = \frac{\alpha - 1}{1 - \lambda} \quad . \qquad (2.3.34)$$

The global maximum is therefore at $x = x_2$ providing $H(x_2) \geq H(x_1)$, which is evidently true since $\lambda > 0$. Thus the sampling efficiency,

$$M^{-1} = \{H(x_2)\}^{-1}$$

$$= \frac{\lambda(1-\lambda)^{\alpha-1}\Gamma(\alpha)}{(\alpha-1)^{\alpha-1} e^{-(\alpha-1)(1+\lambda)} (2-e^{-x(\alpha-1)})}, \qquad (2.3.35)$$

and this is maximised (for given α) when

$$e^{-\lambda(\alpha-1)} = \frac{2\{\lambda^2(\alpha-1) - 1 + \lambda\}}{\{2\lambda^2(\alpha-1) - (\alpha-2)\lambda - 1\}} \quad . \tag{2.3.36}$$

Equation (2.3.36) gives the optimal value of λ to be used, and is in contrast to Tadikamalla, who gives

$$\lambda^{-1} = 0.5 + 0.5(4\alpha-3)^{\frac{1}{2}} \quad . \tag{2.3.37}$$

The difference arises because Tadikamalla has apparently ignored the fact that the normalising factor of (2.3.32) depends upon λ, since it is a truncated distribution. Table 2.2 gives the sampling efficiencies under both (2.3.36) and (2.3.37) for specimen values of α.

Table 2.2 Sampling efficiencies for Tadikamalla's Laplace method, using optimal and non-optimal λ.

α	1.0	1.5	2.0	3.0	5.0	10.0	20.0	25.0
$\lambda_1 \equiv$ (2.3.36)	1.000	0.699	0.249	0.457	0.360	0.269	0.201	0.182
$\lambda_2 \equiv$ (2.3.37)	1.000	0.732	0.259	0.500	0.390	0.282	0.205	0.184
$M^{-1}(\lambda_1)$ (Dagpunar)	1.000	0.868	0.821	0.776	0.740	0.717	0.715	0.717
$M^{-1}(\lambda_2)$ (Tadikamalla)	1.000	0.864	0.815	0.769	0.735	0.715	0.715	0.717

We conclude from the table that the improvement in efficiency using (2.3.36) over (2.3.37) is not significant. In view of this and the numerical procedure required to solve (2.3.36), it is recommended that Tadikamalla's original suggestion for setting λ, (2.3.37), be employed when implementing this algorithm. Tadikamalla gives no results for the sampling efficiency as $\alpha \to \infty$. However for large α,

$X \to N(\alpha,\alpha)$. The optimal Laplace envelope for such a distribution would be one centred on α (not $\alpha-1$) with parameter $\lambda = 1/\sqrt{\alpha}$, giving a sampling efficiency of $\sqrt{\{\pi/(2e)\}} = 0.760$. This therefore represents an upper bound on the asymptotic sampling efficiency for algorithms employing (2.3.36) or (2.3.37).

Schmeiser and Lal (1980) have also used envelope rejection (for $\alpha > 1$) in two algorithms, G2PE and G4PE. The former uses a target distribution which is uniform in the body of the distribution and exponential in the tail(s). G4PE uses a more complete target distribution, comprising six uniform, two triangular and two exponential distributions. In both implementations, pre-tests are incorporated under the name of the "squeeze" technique referred to in Marsaglia (1977). Schmeiser and Lal found both algorithms to be very efficient on a time per-variate generated basis, and in fact G4PE was found to be uniformly faster than others tested. The authors draw attention to the increased set-up time and memory requirements over some other methods. We remark that because of the complexity of the mixture distributions in G4PE, the algorithm could probably not be classed as one which may be rapidly implemented by an experimenter.

Cheng and Feast (1979) use the ratio of uniforms method for $\alpha \geq 1$. Using the notation of section 1.3.3,

$$C = \{(u,v) : 0 \leq u \leq (v/u)^{\frac{\alpha-1}{2}} e^{-v/2u}/\Gamma^{\frac{1}{2}}(\alpha)\}, \qquad (2.3.38)$$

and the minimal rectangle (with sides parallel to the axes) which encloses C is

$$D = \{(u,v) : 0 \leq u \leq \left(\frac{\alpha-1}{e}\right)^{\frac{\alpha-1}{2}} / \Gamma^{\frac{1}{2}}(\alpha) \; ; \; 0 \leq v \leq \left(\frac{\alpha+1}{e}\right)^{\frac{\alpha+1}{2}} /\Gamma^{\frac{1}{2}}(\alpha)\}. \qquad (2.3.39)$$

By generating points uniformly within D, and accepting those for which

$$U \le (V/U)^{\frac{\alpha-1}{2}} e^{-V/2U}/\Gamma^{\frac{1}{2}}(\alpha) , \qquad (2.3.40)$$

the accepted values of V/U have the required distribution. Cheng and Feast do not in fact use the minimal rectangle (2.3.39). Instead they use a looser fitting rectangle with ratio of side lengths (aspect ratio) $\alpha - (6\alpha)^{-1}$. This is to be compared with the aspect ratio of D which is

$$\left(\frac{\alpha+1}{\alpha-1}\right)^{\alpha/2} \left(\frac{\alpha^2-1}{e^2}\right)^{\frac{1}{2}} . \qquad (2.3.41)$$

Their argument for choosing a looser fitting rectangle is that the marginal loss in efficiency is warranted by not having to calculate (2.3.41). This argument seems invalid when α does not vary, since the aspect ratio may be calculated once and saved between calls. Cheng and Feast reduce the number of logarithmic evaluations by incorporating a pre-test into their procedure. The sampling efficiency is not given in their paper, but the area of the rectangle D is

$$\left(\frac{\alpha^2-1}{e^2}\right)^{\alpha/2} \left(\frac{\alpha+1}{\alpha-1}\right)^{\frac{1}{2}}/\Gamma(\alpha) ,$$

and that of C is $\frac{1}{2}$. Thus the sampling efficiency is

$$E = \tfrac{1}{2}\left(\frac{\alpha-1}{\alpha+1}\right)^{\frac{1}{2}} \left(\frac{e^2}{\alpha^2-1}\right)^{\alpha/2} \Gamma(\alpha) , \qquad (2.3.42)$$

which is evaluated for specimen values of α in table 2.3 .

Table 2.3 Sampling Efficiency (E) for Ratio of Uniforms method, using minimal rectangle.

α	1	2	3	4	5	6
E	0.680	0.711	0.628	0.564	0.515	0.380

Using Stirling's formula we may show that $E \to \sqrt{(\pi/2\alpha)}$ as $\alpha \to \infty$. Geometrically the efficiency is decreasing for large α, because the acceptable points are concentrated in the region $\alpha-1 \le v/u \le \alpha+1$. Consequently, for $\alpha > 2.5$, Cheng and Feast have implemented a version in which the region D takes the form of a parallelogram. This encloses C more tightly, and leads to improved sampling efficiencies for these values of α. An alternative strategy for improving sampling efficiency is due to Kinderman and Monahan (1980). Variates are generated from a relocated Gamma distribution with mode at zero. The effect of this is to rotate the u-v axes so that the u-axis is now aligned along the line $v/u = \alpha-1$. In this way the region C is more tightly enclosed by a rectangle, giving a sampling efficiency between $e/4$ $(\alpha = 1)$ and $\sqrt{(\pi e)}/4$ $(\alpha \to \infty)$. When $\alpha < 1$ none of the above methods can be used directly since $f_X(0)$ is unbounded. However Cheng and Feast (1980) have extended the range down to $\alpha = n^{-1}$, by generating a transformed variate $Y = X^{1/n}$, where n is any positive integer.

Atkinson and Pearce (1976) give a brief description of an implementation of Forsythe's method. Specific details of the algorithm are not given in their paper, but we note that for $\alpha \ge 1$, the function $x - (\alpha-1)\ \ell n\ x$ is increasing for $x \ge \alpha - 1$, and decreasing for $x < \alpha - 1$. Using the notation of section 1.3.4, the functions $h_j(x)$ and the intervals (q_{j-1}, q_j) may be defined by:

$$\left.\begin{aligned} h_j(x) &= (x-q_{j-1}) - (\alpha-1)\ \ell n(x/q_{j-1}) \qquad (q_{j-1}<x\le q_j) \\ g_{max} &= (q_j-q_{j-1}) - (\alpha-1)\ \ell n(q_j/q_{j-1}) \end{aligned}\right\} \ (j\ge 1) \qquad (2.3.43)$$

$$q_0 = \alpha - 1 \ ,$$

and

$$\left.\begin{aligned} h_j(x) &= (q_{j-1}-x) - (\alpha-1)\ \ell n(q_{j-1}/x) \qquad (q_{j-1}<x\le q_j) \\ g_{max} &= (q_{j-1}-q_j) - (\alpha-1)\ \ell n(q_{j-1}/q_j) \end{aligned}\right\} (m\le j\le 0), \quad (2.3.44)$$

where $0 < g_{max} \le 1$, and is chosen to control interval width, and m is the largest integer for which $q_{m-1} \le 0$. We note that the difference equations for $\{q_j\}$ are solved first for $j \ge 1$, given q_0, then for $j < 0$, given q_0. When m has been identified, q_{m-1} is set to zero, making the first interval $(0,q_m)$. For $x \ge \alpha - 1$, Atkinson and Pearce suggest the distribution be truncated at some suitable point. In this connection, we note that the tail of the distribution need not be "lost" in this way, if the Gamma tail generation procedure, developed in Chapter 8 is employed.

For $\alpha < 1$, $x - (\alpha-1)\ \ell n\ x$ is increasing for all x, but is unbounded as $x \to 0$. Atkinson and Pearce recommend that the lower boundary of the first interval be set to some small finite value ε (>0). By setting $q_0 = \varepsilon$, the equations (2.3.43) can then be solved for $\{q_j\}$. We remark in passing, that ignoring that part of the distribution which has the highest density of observations is an undesirable feature of the algorithm. It is true that this may be mitigated by making ε extremely small, but since $x - (\alpha-1)\ \ell n\ x$ is changing rapidly as $x \to 0$, the number of intervals required will increase correspondingly.

Atkinson and Pearce report that for $\alpha < 1$, best results are obtained by making g_{max} quite small (0.1 for $\alpha = 0.9$), whereas for $\alpha > 1$, little improvement was noticed by setting g_{max} below 0.9. A disadvantage of the method is the effort required to solve the difference equations in $\{q_j\}$. Indeed, Atkinson and Pearce report that a Fortran program of 160 statements is required just to calculate the constants. A good feature of the algorithm is that the efficiency

remains high as $\alpha \to \infty$. However, there are alternative algorithms which possess this feature, yet are easier to implement and understand, an important feature for the experimenter who has to write a routine quickly for a particular investigation.

For $\alpha \geq 1$, algorithms falling into this latter category include the Cauchy method of Ahrens and Dieter, Greenwood's algorithm, Tadikamalla's Laplace method, Cheng's log logistic method and Kinderman and Monahan's relocated Gamma method. When $\alpha < 1$ algorithms with a high sampling efficiency include those due to McGrath and Irving, while Cheng and Feast found the extended parameter ratio of uniformsmethod to be faster than Ahrens and Dieter's GS algorithm.

2.4 Beta Distribution

This has p.d.f.

$$f_X(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \quad (0 \leq x \leq 1 \; ; \; \alpha > 0, \; \beta > 0), \qquad (2.4.1)$$

and will be referred to as Beta (α,β).

For integral values of the shape parameters α and β, a relationship between the order statistics of random numbers and the Beta distribution provides one method of generation. If $R_{(1)},R_{(2)},\ldots,R_{(n)}$ is an ordered sample of n random numbers, then $R_{(k)}$ is distributed as Beta $(k,n-k+1)$. (See for example Johnson and Kotz 1970b, pp 38). In order to determine the k^{th} smallest of n random numbers, a simple sort based on pairwise comparison (e.g. subroutine PAIRS, Lurie and Mason, 1973) may be used. This requires $\sum_{j=1}^{k} (n-j) = k(2n-1-k)/2$ comparisons. To generate from Beta (α,β) we set $k = \alpha$ and $n = \alpha + \beta - 1$. The method, as it stands is rather unsuitable when both α and β are large, due to the large number of pairwise comparisons.

Newby (1979) suggests that order statistics may be generated more efficiently through their conditional distributions. Given a set of order statistics $X_{(1)},X_{(2)},\ldots,X_{(n)}$ and defining (for the unordered observations) the cumulative hazard function $H(x)$ in terms of the hazard function $\phi(x)$ by

$$H(x) = \int_0^x \phi(u)du ,$$

Newby shows that the conditional distribution of $X_{(j+1)}$ given $X_{(j)}$ is

$$F_{X_{(j+1)}|X_{(j)}}(x_{j+1},x_j) = 1 - e^{-[n-j][H(x_{j+1})-H(x_j)]} \qquad (j=0,1,\ldots,n-1), \tag{2.4.2}$$

$$H(x_0) = 0 .$$

Given a random number R, use of the inversion method on (2.4.2) gives

$$R = 1 - e^{-[n-j][H(X_{(j+1)})-H(X_{(j)})]} ,$$

which leads to the recursion,

$$H(X_{(j+1)}) = H(X_{(j)}) + E_j/(n-j) \qquad (j=0,1,2,\ldots,n-1), \tag{2.4.3}$$

$$H(X_{(0)}) = 0 ,$$

where E_j is a standard Negative Exponential Variate. To generate from Beta (α,β) with $\alpha < \beta$, Newby (1981) uses (2.4.3) to generate $H(X_{(k)})$ where $k = \alpha$ and $n = \alpha + \beta - 1$. For a random variable uniformly distributed in $(0,1)$,

$$H(x) = \int_0^x \frac{du}{1-u} = -\ln(1-x) ,$$

and so

$$X_{(\alpha)} = 1 - e^{-H(X_{(\alpha)})} \tag{2.4.4}$$

is distributed as Beta (α,β). When $\alpha \geq \beta$, k is set to β and

$$X_{(\beta)} = 1 - e^{-H(X_{(\beta)})}$$

is distributed as Beta (β,α). Thus

$$1 - X_{(\beta)} = e^{-H(X_{(\beta)})} \qquad (2.4.5)$$

is returned as a Beta (α,β) variate. Setting $k = \text{Min}(\alpha,\beta)$ in this manner minimises the size of the ordered sample, and makes the algorithm more competitive with other methods when one of the shape parameters is large and the other small.

Methods based on order statistics fail if both shape parameters are not integers. In this case a method due to Johnk (1964) is to generate two random numbers R_1 and R_2 and to return $R_1^{1/\alpha}/(R_1^{1/\alpha}+R_2^{1/\beta})$, subject to $R_1^{1/\alpha} + R_2^{1/\beta} \leq 1$, as a Beta (α,β) variate. The sampling efficiency is $\Gamma(\alpha+1)\Gamma(\beta+1)/\Gamma(\alpha+\beta+1)$. The method is not recommended when α and $\beta > 1$, since the efficiency becomes very small as α and β increase. (e.g. 0.0500 for $\alpha = \beta = 3$, and 0.0040 for $\alpha = \beta = 5$). For α and $\beta < 1$ the sampling efficiency lies between 0.5 and 1 (e.g. 0.5 for $\alpha = \beta = 1$, $\pi/4$ for $\alpha = \beta = 0.5$, and 1 for $\alpha = 0$ or $\beta = 0$). For one parameter larger than 1 and the other smaller than 1, the efficiency decreases as the value of the larger parameter increases (e.g. 0.589 for $\alpha = 0.5$, $\beta = 1.5$, and 0.369 for $\alpha = 0.5$, $\beta = 5$).

An alternative method applicable for all values of α and β is to generate two Gamma variates X_1, X_2 with shape parameters α and β respectively. Then $X_1/(X_1+X_2)$ is distributed as Beta (α,β). Fishman (1973, pp 204-205) implies that this is suitable only for integral α and β, though clearly the current availability of Gamma routines for non-integral shape parameters means that in principle the method could be used for all real values of α and β.

Other available methods utilise the envelope rejection technique. When α and β are both greater than one, the p.d.f. is bounded over the entire domain, and an obvious approach is to employ a uniform target distribution, $g_Y(x) = 1$. Since the mode of Beta (α,β) is at $(\alpha-1)/(\alpha+\beta-2)$,

$$M = \max_x \left(f_X(x)/g_Y(x)\right) = \frac{1}{B(\alpha,\beta)} \left(\frac{\alpha-1}{\alpha+\beta-2}\right)^{\alpha-1} \left(\frac{\beta-1}{\alpha+\beta-2}\right)^{\beta-1}, \qquad (2.4.6)$$

and this becomes unacceptably high as either α or β becomes large.

An alternative method when both parameters are greater than 1 is algorithm BN due to Ahrens and Dieter (1974). They utilise a Normal target distribution with mean $\mu = (\alpha-1)/(\alpha+\beta-2)$, corresponding to the mode of the Beta distribution, and standard deviation $\sigma = 0.5/\sqrt{(\alpha+\beta-2)}$. The number of trials per generated Variate (including those rejected for not falling in (0,1)) is given by Ahrens and Dieter as

$$\left\{\frac{\pi}{2(\alpha+\beta-2)}\right\}^{\frac{1}{2}} \left\{ \frac{(\alpha-1)^{\alpha-1}(\beta-1)^{\beta-1}}{(\alpha+\beta-2)^{\alpha+\beta-2}\, B(\alpha,\beta)} \right\}. \qquad (2.4.7)$$

Comparison with (2.4.6) shows that (2.4.7) is larger when $\alpha + \beta < 2 + \pi/2$. Given the speed of generation from a uniform target distribution compared to that from a Normal distribution, the uniform target method is certainly preferable when $\alpha + \beta < 2 + \pi/2$, and probably for values of $\alpha + \beta$ exceeding $2 + \pi/2$ by some undetermined amount. Table 2.4 below shows some specimen values of the sampling efficiency for algorithm BN. The efficiency is rather poor when one of the parameters is small, particularly when the other is large. When $\alpha = \beta$, the sampling efficiency asymptotically approaches 1 as $\alpha \to \infty$, reflecting the symmetry and approximate Normality of such distributions.

Table 2.4 Sampling Efficiency for Ahrens and Dieter's algorithm BN

β \ α	$1+\varepsilon^{\dagger}$	1.5	2	5	10	20
1	$2\varepsilon/\pi$	0.376	0.399	0.319	0.239	0.174
1.5		0.627	0.677	0.601	0.471	0.351
2			0.752	0.726	0.592	0.450
5				0.917	0.878	0.742
10					0.961	0.912
20						0.981

($\dagger$ $\varepsilon \to +0$)

When α and β are both close to 1 the sampling efficiency approaches zero. The reason for this is that the Normal target distribution has a very large variance. Thus a very high proportion of observations fall outside (0,1) and have to be rejected. The deficiency of the algorithm in this respect is confirmed by the absence of any computer timings when α and β are both close to 1 in table B of Ahrens and Dieter's paper.

Schmeiser and Shalaby (1980) modify algorithm BN by incorporating a pre-test, using the "squeeze" technique. The speed of the resulting method (BNM) shows little improvement, particularly in the region of $\alpha \to 1$. In the same paper they consider target distributions consisting of a mixture of a uniform and two triangular distributions (algorithm 2P) and six uniform and four triangular distributions (algorithm 4P). These methods are analogous to the Gamma methods in Schmeiser and Lal (1978). Schmeiser and Shalaby compared speeds for these algorithms with other Beta methods (but excluding the efficient Forsythe implementation of Atkinson and Pearce, 1976). They found that for most values of α and β either 2P or 4P was the fastest. Schmeiser and Babu (1980) modify algorithms 2P and 4P by utilising Exponential

tails for the target distribution, and note a marginal improvement in speed.

When both parameters are less than one, Atkinson and Whittaker's (1976) switching algorithm employs a target distribution,

$$g_Y(x) = rg_1(x) + (1-r)g_2(x) \quad . \tag{2.4.8}$$

where

$$g_1(x) = \begin{cases} \alpha\, x^{\alpha-1}/t^{\alpha} & (x \le t) \\ 0 & (x > t)\ , \end{cases} \tag{2.4.9}$$

$$g_2(x) = \begin{cases} 0 & (x \le t) \\ \beta(1-x)^{\beta-1}/(1-t)^{\beta} & (x > t)\ , \end{cases} \tag{2.4.10}$$

t is a free parameter in the range $(0,1)$, and $0 < r < 1$. Thus

$$\frac{f_X(x)}{g_Y(x)} = \begin{cases} (1-x)^{\beta-1}t^{\alpha}/\{r\ \alpha\ B(\alpha,\beta)\} & (x \le t) \\ x^{\alpha-1}(1-t)^{\beta}/\{(1-r)\ \beta\ B(\alpha,\beta)\} & (x > t). \end{cases} \tag{2.4.11}$$

The maximum of $f_X(x)/g_Y(x)$ in the range $[0,t]$ occurs at $x = t$, and in $(t,1]$ at $t + \varepsilon$ $(\varepsilon \to 0+)$, the two maxima being equal when

$$\frac{(1-t)^{\beta-1}t^{\alpha}}{r\ \alpha} = \frac{t^{\alpha-1}(1-t)^{\beta}}{(1-r)\beta}\ ,$$

which leads to a choice of

$$r = Bt/\{(1-t)\alpha + Bt\}\ , \tag{2.4.12}$$

and a sampling efficiency of

$$M^{-1} = \frac{\alpha\beta\ B(\alpha,\beta)t^{1-\alpha}(1-t)^{1-\beta}}{[(1-t)\alpha + Bt]} \quad . \tag{2.4.13}$$

Atkinson and Whittaker show that for any t the sampling efficiency is lower than that of Johnk's, when $\alpha + \beta < 1$. The optimum value of t is that maximising M^{-1}, and is

$$t_{opt} = \frac{\{\alpha(1-\alpha)\}^{\frac{1}{2}}}{\{\alpha(1-\alpha)\}^{\frac{1}{2}} + \{\beta(1-\beta)\}^{\frac{1}{2}}} \quad . \tag{2.4.14}$$

Using this value of t, the sampling efficiency is superior to Johnk's when $\alpha + \beta > 1$, and equal to Johnk's when $\alpha + \beta = 1$. This theoretical analysis is supported by computational experience reported by Atkinson and Whittaker.

When one parameter, α say, is less than 1, and the other is larger than 1, Atkinson and Whittaker suggest a target distribution as before. The maximum of $f_X(x)/g_Y(x)$ in the range $[0,t]$ is now at $x = 0$, and in $(t,1]$ it occurs at $x = t + \varepsilon$ $(\varepsilon \to 0)$, as before. Equating the two maxima leads to a choice of

$$r = Bt/\{\alpha(1-t)^{\beta} + Bt\} , \qquad (2.4.15)$$

which gives a sampling efficiency of

$$M^{-1} = \frac{\alpha\ \beta\ B(\alpha,\beta)}{\{Bt^{\alpha} + \alpha(1-t)^{\beta}t^{\alpha-1}\}} . \qquad (2.4.16)$$

The optimum value of t cannot be found analytically, and Atkinson and Whittaker used the method of false position. Their computational experience shows that, using the optimal value of t, the speed of the algorithm is fairly insensitive to values of α and β, and comparable to the speed of their algorithm when $\alpha < 1$ and $\beta < 1$ $(\alpha + \beta > 1)$, or Johnk's method when $\alpha < 1$ and $\beta < 1$ $(\alpha + \beta \leq 1)$.

Atkinson and Whittaker remark that the target distribution could in theory be used for the case $\alpha > 1$, $\beta > 1$. However limited computational experience showed that the time per variate generated increases dramatically with increasing α and β. Atkinson (1979a) gives an efficient algorithm for this range by replacing the target distribution in (2.4.9)-(2.4.10) with $g_1(x) = px^{p-1}/t^p$ and $g_2(x) = q(1-x)^{q-1}/(1-t)^q$ where $1 \leq p \leq \alpha$ and $1 \leq q \leq \beta$. Optimal values of p, q and t require numerical methods, but a non-optimised version yielded sampling efficiencies which were no more than 3% lower than those for the optimised version.

Cheng (1978) notes that if X has a Beta prime distribution with p.d.f.

$$f_X(x) = x^{\alpha-1}(1+x)^{-(\alpha+\beta)}/B(\alpha,\beta) \quad . \quad (x \geq 0), \tag{2.4.17}$$

then $U = X/(1+X)$ is distributed as Beta (α,β). To sample from (2.4.17) envelope rejection is used, with a target distribution,

$$g_Y(x) = \lambda \mu x^{\lambda-1}/(\mu+x^{\lambda})^2 \qquad (x \geq 0) \ , \tag{2.4.18}$$

where $\mu = (\alpha/\beta)^{\lambda}$ and

$$\lambda = \begin{cases} \text{Min } (\alpha,\beta) & \{\text{Min } (\alpha,\beta) \leq 1\} \\ \sqrt{\{(2\alpha\beta-\alpha-\beta)/(\alpha+\beta-2)\}} & \{\text{Min } (\alpha,\beta) > 1\}. \end{cases} \tag{2.4.19}$$

For this choice of μ and λ, $f_X(x)/g_Y(x)$ has a unique maximum at $x = \alpha/\beta$, leading to a sampling efficiency of

$$M^{-1} = \lambda B(\alpha,\beta)(\alpha+\beta)^{\alpha+\beta}/(4\alpha^{\alpha}\beta^{\beta}) \ . \tag{2.4.20}$$

For α and $\beta \geq 1$, $M^{-1} > (e/4) = 0.680$. When at least one of the parameters is less than 1, $M^{-1} > 0.25$, approaching this limit when either α or $\beta \to 0$. When $\alpha = \beta = 1$, $M^{-1} = 1$, in contrast to the poor performance of Ahrens and Dieter's BN algorithm. Modifications to the basic Cheng algorithm include pre-tests which reduce the number of logarithmic evaluations.

Finally we mention an implementation of Forsythe's method (Atkinson and Pearce, 1976). Computational experience showed that the timings were fairly insensitive to values of α and β, and in most cases outperformed the other methods considered (Johnk, Ahrens and Dieter BN, rejection using a uniform target distribution). It requires elaborate programming in order to set up the constants. Atkinson and Pearce report that the set up time is equivalent to the generation of approximately 1000 variates.

In assessing the suitability of the various methods, it should be noted that although Newby's order statistic method should be reasonably efficient when Min (α,β) is small, it is restricted to integer α and β, and is therefore limited in its application. As with most distributions, Forsythe's method is likely to be very competitive with other algorithms on a time per variate generated basis, but carries the overhead of more complicated programming and calculation of constants. Regarding the method involving two Gamma variates, Atkinson and Pearce indicate that generating two Gamma variates using Forsythe's method would be slower than direct application of Forsythe's method.

For the case $\alpha \leq 1$ and $\beta \leq 1$, Johnk's method is extremely easy to program, easily understood and relatively efficient. For $\alpha + \beta \gtrsim 1$, Atkinson and Whittaker's method proved faster. If, for convenience, only one algorithm were required for the range $\alpha \geq 1$, $\beta \geq 1$, then the simplicity of Johnk's method might justify its use.

For $\alpha > 1$ and $\beta > 1$, Ahrens and Dieter's algorithm BN is not recommended, principally because of its poor performance if both parameters are close to 1. In contrast Cheng's (1978) algorithm gives good sampling efficiencies for this range, with an asymptotic sampling efficiency of 1 as α and $\beta \to 1$. Considering the ease with which it may be implemented, it is recommended for α and β in this range. Similar behaviour can be expected for Atkinson's (1979a) switching algorithm, although no timing comparisons with Cheng's method are available.

For $\alpha \leq 1$ and $\beta \geq 1$, Johnk's and Cheng's methods do not provide a uniformly high sampling efficiency, as specimen values in table 2.5 show. Atkinson and Whittaker's switching algorithm requires a numerical procedure to determine the optimal value of t. However use of a non-optimal value, $t = (\alpha-1)/(\beta+1-\alpha)$, suggested by Atkinson (1979a) gives a

uniformly high sampling efficiency as shown in table 2.5. These results are consistent with Atkinson's timing experiments which showed that the non-optimised t never increased execution time by more than 10%. In view of this the method is recommended for this range of α and β.

Table 2.5 Sampling efficiencies for $\alpha \leq 1$ and $\beta \geq 1$, using the methods of Atkinson and Whittaker with $t = (\alpha-1)/(\beta+1-\alpha)$ (AW), Cheng (C) and Johnk (J).

α		β = 1	2	5	10
0.01	AW	0.997	0.994	0.992	0.992
	C	0.264	0.264	0.263	0.263
	J	0.990	0.985	0.977	0.971
0.1	AW	0.970	0.949	0.933	0.927
	C	0.350	0.340	0.334	0.333
	J	0.909	0.866	0.801	0.752
0.5	AW	0.866	0.828	0.801	0.791
	C	0.650	0.582	0.543	0.530
	J	0.667	0.533	0.369	0.270
0.9	AW	0.865	0.852	0.842	0.839
	C	0.931	0.793	0.708	0.679
	J	0.526	0.363	0.193	0.112
1.0	AW	1.000	1.000	1.000	1.000
	C	1.000	0.844	0.746	0.713
	J	0.500	0.333	0.167	0.091

2.5 Some other continuous distributions

In this section we outline briefly, approaches to generation from some other continuous distributions. Neither the distributions covered, nor the methods mentioned are intended to be exhaustive, mainly due to the diverse origins and structural properties of many distributions.

A random variable has the Lognormal distribution if its p.d.f. is of the form,

$$f_X(x) = \frac{e^{-(\ell nx-\mu)^2/2\sigma^2}}{\sqrt{2\pi}\,\sigma\,x} \qquad (x \geq 0), \qquad (2.5.1)$$

where μ and σ (>0) are constants. Structurally, $X = e^Y$ where $Y \sim N(\mu,\sigma^2)$, and this provides an obvious method of generation.

The standard Cauchy distribution has p.d.f.,

$$f_X(x) = \{\pi(1+x^2)\}^{-1} \quad , \qquad (2.5.2)$$

and generation using inversion has implicitly been dealt with in algorithm 1.3. An alternative is to take the ratio of two standard Normal deviates (see for example Kendall and Stuart, (1963), pp 268).

Generation from the three parameter Weibull is conveniently managed using inversion, as described in Section 1.1 .

A variate X has the Laplace distribution if its p.d.f. is

$$f_X(x) = \tfrac{1}{2}\,\lambda e^{-\lambda|x-\beta|} \qquad (\lambda > 0) \quad . \qquad (2.5.3)$$

In addition to being useful in its own right, it has also been used as a target distribution in the Gamma rejection based method of Tadikamalla, (1978b). The c.d.f. is

$$F_X(x) = \begin{cases} 1 - \tfrac{1}{2}\, e^{-\lambda(x-\beta)} & (x \geq \beta) \\ \tfrac{1}{2}\, e^{-\lambda(\beta-x)} & (x < \beta) \end{cases} , \qquad (2.5.4)$$

and given a random number R, inversion leads to the generation scheme,

$$X = \begin{cases} \beta - \lambda^{-1}\ell n\{2(1-R)\} & (R \geq \tfrac{1}{2}) \\ \beta + \lambda^{-1}\ell n\{2R\} & (R < \tfrac{1}{2}) \ . \end{cases} \qquad (2.5.5)$$

The logistic distribution has p.d.f.

$$f_X(x) = \frac{e^{-(x-\alpha)/\beta}}{\beta\{1 + e^{-(x-\alpha)/\beta}\}^2} \qquad (\beta > 0) \ , \qquad (2.5.6)$$

and c.d.f.,

$$F_X(x) = \{1 + e^{-(x-\alpha)/\beta}\}^{-1} ,$$

which can be inverted to give the generation scheme,

$$X = \alpha + \beta \ \ell n\{R/(1-R)\} \ . \qquad (2.5.7)$$

A variate χ^2_n having p.d.f.,

$$f_{\chi^2_n}(x) = \frac{e^{-x/2}\ x^{(n/2)-1}}{2^{n/2}\ \Gamma(n/2)} \quad (x \geq 0 \ ; \ n = 1,2,\ldots), \qquad (2.5.8)$$

is distributed as chi-squared with n degrees of freedom. Equivalently this is a Gamma distribution with shape parameter $\alpha = n/2$ and scale parameter $\lambda = 1/2$. Thus one method is to generate a standard Gamma variate X and set $\chi^2_n = X/\lambda = 2X$. The current availablity of efficient routines for integer and non-integer α $(\geq \tfrac{1}{2})$ allows χ^2_n variates to be conveniently obtained, for both n odd and even-valued. Alternatively,

$$\chi^2_n = \sum_{i=1}^{n} Z_i^2 \ , \qquad (2.5.9)$$

where $Z_i \sim N(0,1)$. For large n however, the time required to generate n standard Normal deviates is likely to be prohibitive. A third method when n is odd valued, is to set $\chi^2_n = Z_1^2 + G_\alpha$, where G_α is a Gamma variate with (integral) shape parameter $\alpha = (n-1)/2$ and scale parameter $\lambda = 1/2$. G_α may be generated using the sum of α

Exponential variates, but this is likely to be inefficient when n is large.

Non-central chi-squared variates S_n, with non-centrality parameter μ (>0) and n degrees of freedom may be conviently generated by noting that

$$S_n = \sum_{i=1}^{n-1} Z_i^2 + (Z_n - \mu^{\frac{1}{2}})^2$$

$$= \chi^2_{n-1} + (Z_n - \mu^{\frac{1}{2}})^2 . \qquad (2.5.10)$$

Variates having p.d.f.

$$f_{T_n}(x) = \frac{(1 + \frac{x^2}{n})^{-(n+1)/2}}{\sqrt{n} \cdot B(\frac{1}{2},\frac{n}{2})} \qquad (n = 1,2,\ldots), \qquad (2.5.11)$$

follow a t-distribution with n degrees of freedom. One method of generation is to set

$$T_n = Z/\sqrt{\{\chi^2_n \, n^{-1}\}} . \qquad (2.5.12)$$

Kinderman, Monahan and Ramage (1977) propose several alternatives to (2.5.12), based on envelope rejection methods incorporating both acceptance and rejection pre-tests. Kinderman and Monahan (1980) derive a ratio of uniforms method with sampling efficiency between $\pi/4$ $(n = 1)$ and $\sqrt{(\pi e)}/4$ $(n \to \infty)$. A further alternative is to deliver T_n as the square root of an F-variate with 1 and n degrees of freedom.

A variate F_{n_1,n_2} having p.d.f.

$$f_{F_{n_1,n_2}}(x) = \frac{n_1^{(n_1/2)} \; n_2^{(n_2/2)} \; x^{(n_1/2)-1}}{B(n_1/2,n_2/2)(n_2+n_1x)^{(n_1+n_2)/2}} , \qquad (2.5.13)$$

where $x \geq 0$ and $n_1, n_2 = 1, 2, \ldots$, follows an F-distribution with n_1 and n_2 degrees of freedom. One method of generation is to set $F_{n_1,n_2} = (n_2\chi^2_{n_1})/(n_1\chi^2_{n_2})$. Kinderman, Monahan and Ramage mention generation via transformation of the associated Beta variate, but we are not aware of any method exploiting the relationship between F and Beta-prime variates, X. This takes the form, $X = n_1 F_{n_1,n_2}/n_2$ where

$$f_X(x) = \frac{x^{(n_1/2)-1}}{B(n_1/2, n_2/2)(1+x)^{(n_1+n_2)/2}} \qquad (x \geq 0), \qquad (2.5.14)$$

Johnson and Kotz (1970b, pp 77). An efficient generation procedure for X has already been described in (2.4.17)-(2.4.19), where we identify $\alpha = n_1/2$ and $\beta = n_2/2$. From (2.4.20) we deduce that $M^{-1} > e/4 = 0.680$ if $n_1 \geq 2$ and $n_2 \geq 2$. If $n_1 = 1$, $M^{-1} = \pi/4$ at $n_2 = 1$, $(\sqrt{3}/2)^3 = 0.650$ at $n_2 = 2$, approaching an asymptotic value of $\sqrt{\{\pi e/32\}} = 0.517$ as $n_2 \to \infty$ (obtained using Stirling's formula). We conclude that Cheng's efficient and easily implemented algorithm for the Beta-prime distribution, although originally intended for Beta generation, would appear to be of wider applicability. In particular, F_{n_1,n_2} and T_n variates may be readily obtained without explicit Beta variate generation.

CHAPTER 3

METHODS OF GENERATION FROM SPECIFIC DISCRETE DISTRIBUTIONS

3.1 Binomial Distribution

X is Binomially distributed $\left(\text{Binomial}(n,p)\right)$ if its p.m.f. is

$$f_X(x) = \binom{n}{x} p^x (1-p)^{n-x} \qquad (x = 0,1,\ldots,n), \tag{3.1.1}$$

where $0 \le p \le 1$. A well known method is to simulate n Bernoulli trials, with p being the probability of success per trial. The method becomes quite inefficient for large n. An alternative is to invert the cumulative distribution function. This results in an execution time which is linear in np. Generation from Binomial $(n,1-p)$ if $p > \frac{1}{2}$, will bring this down to $n\,\text{Min}(p,1-p)$. Besides being unbounded in n, the method carries an overhead of initial calculation of the cumulative probabilities, and additionally, requires these values to be stored.

Fishman (1978, pp 447) describes an envelope rejection procedure (B2), based on a uniform target distribution,

$$g_Y(x) = (n+1)^{-1} \qquad (x = 0,1,2,\ldots,n)\ . \tag{3.1.2}$$

Given that the mode of $f_X(x)$ is at $x' = \langle (n+1)p \rangle$, the sampling efficiency is M^{-1}, where,

$$M = f_X(x')/g_Y(x') = \binom{n}{x'} p^{x'} (1-p)^{n-x'} (n+1)\ , \tag{3.1.3}$$

and not as shown by Fishman, where the mode is incorrectly taken as $\langle np \rangle$. For large n, the behaviour of M is easily obtained by noting that $X \to N(np, np[1-p])$, and so

$$M \doteq (n+1) \max_{x} \left[\frac{e^{-\frac{1}{2}(x-np)^2/[np(1-p)]}}{\sqrt{\{2\pi np(1-p)\}}} \right]$$

$$\doteq [n/\{2\pi p(1-p)\}]^{\frac{1}{2}} , \qquad (3.1.4)$$

which is again unbounded as $n \to \infty$.

Relles (1972) exploits a relationship between the Binomial and Beta distributions. Let $X_{n,p}$ and V be variates from Binomial (n,p) and Beta $(j,n-j+1)$ respectively for any $j \in \{1,2,\ldots,n\}$.

Then

$$X_{n,p} = \begin{cases} j + X_{n-j,\,\frac{p-V}{1-V}} & (V \leq p) \qquad (3.1.5) \\ X_{j-1,\,\frac{p}{V}} & (V > p) , \qquad (3.1.6) \end{cases}$$

and $$X_{0,\cdot} = 0. \qquad (3.1.7)$$

A proof of this decomposition is given in Ahrens and Dieter (1974), and we remark that Fishman (1978, pp 449 and 1979) incorrectly gives the right hand side of (3.1.5) as $j + X_{n-j+1,\,\frac{p-V}{1-V}}$. Equations (3.1.5)-(3.1.6) are applied recursively, until a variate $X_{0,\cdot}$ is called for, which terminates the recursion. In this way a Binomial(n,p) variate may be generated through a sequence of Beta variates. The choice of j (at a given stage in the recursion) is left to the implementer of the algorithm. In algorithms RBINOM (Relles) and BB (Ahrens and Dieter) j is set to $(n+1)/2$, if n is odd valued. If n is even valued, a single Bernoulli variate is generated and added to a Binomial$(n-1,p)$ variate.

An interesting problem, which has not previously been formulated, is to determine the optimal choice of j at any stage in the recursion. This will depend upon the parameter values n and p. Thus we may define $\lambda^*_{n,p}$ to be the minimum expected number of iterations (i.e. Beta variates) required to generate a Binomial(n,p) variate, the minimisation being over a sequence of decision variables $\{j = j(n,p)\}$. Using the

Principle of Optimality, we derive the recursion,

$$\lambda^*_{n,p} = 1 + \min_j \{ \int_0^p \lambda^*_{n-j,\frac{p-v}{1-v}} v^{j-1}(1-v)^{n-j}/B(j,n-j+1).dv$$

$$+ \int_p^1 \lambda^*_{j-1,\frac{p}{v}} v^{j-1}(1-v)^{n-j}/B(j,n-j+1).dv\} \qquad (3.1.8)$$

for $n = 1,2,\ldots$, with $\lambda^*_{0,\cdot} = 0$.

The solution to (3.1.8) appears to be non-trivial, and is not presented as part of this thesis, although it is intended to investigate the problem at some later time.

It is of interest to determine the expected number of iterations if j is set to 1. Dropping the minimisation in (3.1.8), putting $j = 1$, and defining $\lambda_{n,p}$ to be the expected number of Beta variates required under such a strategy, we have,

$$\lambda_{n,p} = 1 + \int_0^p \lambda_{n-1,\frac{p-v}{1-v}} (1-v)^{n-1}/B(1,n)dv \quad (n = 1,2,\ldots), \qquad (3.1.9)$$

with $\lambda_{0,\cdot} = 0$. It may be verified that

$$\lambda_{n,p} = np + 1 - p^n \qquad (3.1.10)$$

is a solution to these difference equations. This result is more conveniently obtained as follows. Let p_x denote the probability that a Binomial(n,p) variate takes the value x. Then for $x < n$ the number of Beta variates required is $x + 1$, while for $x = n$ it is n. Thus the expected number of Beta variates required is $\sum_{x<n} (1+x)p_x + np_n = np + 1 - p^n$. From (3.1.10), we conclude that the mean number of Beta variates required is always greater than the mean number of comparisons in the Inversion method (np), and therefore that the Beta method, under the $j = 1$ strategy is not competitive.

It has not been mentioned in the literature that the decomposition (3.1.5)-(3.1.7) provides an elegant method of generating truncated Binomial variates. To generate from Binomial(n,p) subject to $X_{n,p} \geq j$, sample the initial Beta variate subject to $V \leq p$. Conversely, generation subject to $X_{n,p} < j$, is via the generation of an initial Beta variate subject to $V > p$.

A disadvantage of methods described above is that in each case the execution time per variate generated is unbounded as $n \to \infty$. Fishman (1979) gives an envelope rejection procedure, with a bounded execution time. The target distribution is Poisson, mean μ. The proportion of Poisson variates which are accepted is maximised when

$$\mu = \begin{cases} n - \langle n(1-p)\rangle & \{n(1-p)-\langle n(1-p)\rangle \leq p\} \\ p[\langle n(1-p)\rangle+1]/(1-p) & \text{(otherwise)}. \end{cases} \qquad (3.1.11)$$

If $p > \frac{1}{2}$, sampling is from Binomial(n,1-p). The proportion of Poisson variates which are accepted is never less than $1/\sqrt{2}$. The speed of the algorithm will depend upon the nature of the Poisson generator. Fishman's implementation (BP) uses a Poisson generator PIF, with execution time varying as $\mu^{\frac{1}{2}}$. Thus for large μ, Fishman finds it necessary to modify PIF, by generating Poisson variates from the corresponding Normal approximation. Timing comparisons with the Beta method and the inversion method, showed that the Poisson method was fastest only when n is large ($\gtrsim 20$) and Min(p,1-p) not close to zero, most of these cases corresponding to the use of the Normal approximation. For the remaining cases the inversion method proved to be fastest except when n was small ($n \lesssim 5$) and Min(p,1-p) was not close to zero. In these cases the Beta method was fastest.

Ahrens and Dieter (1974) have developed a method (algorithm BC; "count ones") based on the representation of the success probability

as a sequence of random bits taking values zero or one. More recently Ahrens and Dieter (1980) have employed envelope rejection with a Laplace target distribution, tail values outside [0,n] being rejected. The Laplace envelope ensures that the sampling efficiency approaches $\sqrt{\{\pi/(2e)\}}$ as $\mu \to \infty$, corresponding to the Normal approximation to the Binomial. The method is exact and has a bounded execution time, thus satisfying a requirement which the previous generators do not. However it is not particularly fast, Ahrens and Dieter remarking that Inversion is faster if $\text{Min}(np,n-np) < 16$, while Fishman's method BP (exact version, without Normal approximation) was faster when $np < 100$.

3.2 Poisson Distribution

In this section we review methods available for sampling variates having p.m.f.

$$p_x = f_X(x) = \frac{\mu^x e^{-\mu}}{x!} \qquad (\mu > 0,\ x = 0,1,2,\ldots)\ , \tag{3.2.1}$$

and c.d.f.

$$q_x = \sum_{j=0}^{x} p_j\ . \tag{3.2.2}$$

We denote such variates as Poisson(μ). A recent study giving comparative timings for many of the methods to be described appears in Atkinson (1979b and 1979c).

Probably the most familiar procedure is the multiplicative method, described in section 1.2. This becomes inefficient for large μ, due to the increasing requirement of random numbers. An alternative is inversion. Two approaches are possible, one in which the cumulative probabilities are pre-calculated and stored (less suitable when μ varies between calls), the other where the cumulative

probabilities are calculated as required for each variate generated. When μ is not too large, the latter is not as inefficient as might first appear, since the individual probabilities are conveniently calculated through the recursion $p_x/p_{x-1} = \mu/x$. For large μ, both approaches become unattractive, since the number of comparisons varies as μ.

A better method is to start the search at the mode of the distribution $\langle\mu\rangle$, searching for the smallest x satisfying $q_x \geq R$, if $R \geq q_{\langle\mu\rangle}$, or the largest x satisfying $q_{x-1} < R$, if $R < q_{\langle\mu\rangle}$. Fishman (1976b) incorporates these ideas into algorithm PIF. The search from the mode is used for all integral μ, although in principle it could be used for non-integral values as well. For non-integral $\mu < 7$, Fishman uses conventional inversion, starting at $x = 0$. For non-integral $\mu \geq 7$, a Poisson ($\langle\mu\rangle$) variate is generated using the modal search, and added to a Poisson ($\mu - \langle\mu\rangle$) variate. Such an algorithm still has an unbounded execution time as $\mu \to \infty$, so in one version of the algorithm, Fishman incorporates a switch for large μ, whereby variates are sampled using the Normal approximation to the Poisson.

A natural extension of the modal search, is to divide the probability distribution into k equal parts, and use the indexed search method of Chen and Asau, described in section 1.6.5, As k increases, the amount of pre-calculation and storage increases, accompanied by a decrease in per-variate execution time. Another implementation requiring pre-calculation of the individual probabilities and substantial storage is the method of Norman and Cannon.

Ahrens and Dieter (1974) show that a Poisson variate may be decomposed into either a Poisson variate (having smaller mean) or a Binomial variate. Let X_μ, G_n, $X_{n,p}$ denote independent Poisson (μ).

Gamma(n) and Binomial (n,p) variates. Then Ahrens and Dieter show (pp. 239-240) that

$$X_\mu = \begin{cases} X_{n-1,\mu/G_n} & (G_n > \mu) \qquad (3.2.3) \\ n + X_{\mu-G_n} & (G_n \leq \mu) \; . \qquad (3.2.4) \end{cases}$$

Thus a Poisson variate may be generated recursively through a sequence of Gamma and Binomial variates. In algorithm PG, n is set to $\langle 0.875\mu \rangle$, and once the recursion requires a Poisson variate of sufficiently small mean, this variate is sampled directly using the multiplicative method. Ahrens and Dieter report that their choice of n necessitates the generation of few Binomial variates, and thus justify the use of the relatively inefficient Bernoulli method. Their computational experience for the whole algorithm shows that the method is slower than the multiplicative one if $\mu < 16$ (for Fortran) and the final version of their algorithm incorporates a switch to this effect.

We remark that the Binomial variate in (3.2.3) could itself be generated through a sequence of Beta variates, (3.1.5)-(3.1.7), making the algorithm dependent upon Gamma and Beta variates only. A problem analogous to the one formulated in (3.1.8) is to find the optimal decomposition of the Poisson distribution, so that the mean requirement of Gamma and Beta variates is minimised.

Ahrens and Dieter (1974) also describe the Centre-Triangle algorithm (PT). If μ is sufficiently small (<9), the multiplicative method is used. Otherwise sampling is via the generation of Poisson ($\langle\mu\rangle$) and Poisson ($\mu-\langle\mu\rangle$) variates. The former is via a probability mixture method, comprising a triangular p.d.f. and a (small) residual density. Poisson ($\mu-\langle\mu\rangle$) variates are generated through a pre-calculated c.d.f.. The method requires considerable setting up of constants and tables, and Atkinson's timing comparisons indicate that execution time is unbounded as $n \to \infty$.

One of the attractions of the Alias method, described in section 1.4 is that, apart from set-up time, the execution time is largely independent of the parameters of the distribution. Atkinson (1979c) has implemented this for the Poisson, truncating the distribution at a suitable point in the upper tail of the distribution. His experiments show the algorithm to be very fast, out performed (for $\mu \lesssim 20$) only, by the indexed search method with 100 intervals. In Chapter 6 a modification of the Alias method, which avoids artificial truncation of the distribution will be discussed.

Ahrens and Dieter (1980) have applied the Laplace envelope method, previously seen in Binomial generation, to the Poisson. The result is an exact algorithm with bounded execution times. They found inversion to be faster when $\mu \leq 12$. Another envelope rejection method is due to Atkinson (1979b), which again has the merit of being exact and possessing a bounded sampling efficiency on $\mu \to \infty$. The target distribution is the Logistic. Atkinson reports that the performance of this algorithm is disappointing and highlights two problems. One is that the method is slowed down by the calculation of the Logistic density, and the other that the algorithm ceases to become compact due to the necessity of storing a table of logarithms of factorials. In Chapter 6 we describe modifications to the method which eliminate these difficulties.

The choice of Poisson generator depends on the criteria set. If speed is all important, regardless of set-up time and preparation, Atkinson's experiments showed that the Alias, Indexed search (100 intervals), and method of Norman and Cannon are all very fast, the first two having the advantage of being exact (apart from the tail of the distribution). A disadvantage is the artificial truncation of the

probability distribution in the upper tail. The timings for algorithm PT of Ahrens and Dieter do not appear competitive, given the complexity of the algorithm and consequent set-up overhead. Algorithm PG using decomposition of the Poisson variate did not perform well in Ahrens and Dieter's tests, although other strategies for decomposition may yield better results. Finally for small μ the multiplicative method is relatively fast, easy to program and particularly suitable when μ changes from one call of the generator to the next.

3.3 Other discrete distributions

In this section, we review briefly methods available for sampling from some other discrete distributions.

The discrete analogue of the Negative Exponential distribution is the Geometric, having p.m.f.,

$$p_x = f_X(x) = p(1-p)^{x-1} \qquad (x = 1,2,\ldots) , \qquad (3.3.1)$$

where $0 < p < 1$. Since X represents the number of Bernoulli trials till the first success, with success probability p per trial, one method is to simulate the individual trials. The mean requirement of random numbers per Geometric variate is $E(X) = 1/p$, and thus the method becomes inefficient for small p. An alternative is to invert the c.d.f., leading to one logarithmic evaluation per Geometric Variate as described in section 1.1 .

A Variate X, having p.m.f.

$$p_x = f_X(x) = \binom{x + k - 1}{x} p^x (1-p)^k \qquad (x = 0,1,2,\ldots), \qquad (3.3.2)$$

where $0 < p < 1$ and $k > 0$, has the Negative Binomial Distribution,

which we denote as N.Binom(k,p). For integral k, X + k represents the sum of k independent Geometric Variates, each having mean $(1-p)^{-1}$. Thus one generation method is to simulate Bernoulli trials till the k^{th} failure occurs, as in Ahrens and Dieter (1974), algorithm NU. Such a method requires a mean of k/(1-p) random numbers, and becomes inefficient as this ratio becomes large. If k/(1-p) is large, but k small, a better method would be to generate k Geometric Variates by inversion, requiring k logarithmic evaluations per Variate generated.

Neither of these methods can be used directly if k is non-integral. In these cases a numerical search of the c.d.f. can be employed (as in Ahrens and Dieter (1974), algorithm NS), with the c.d.f. updated during the search, using the recursion $p_x/p_{x-1} = \{1 + (k-1)/x\}p$. The mean number of comparisons per generated variate is 1 + E(X) = 1 + kp/(1-p). An alternative is to sample Variates from N.Binom(<k>,p), using one of the integer methods, and then to add this to a N.Binom(k - <k>,p) variate, generated by numerical inversion of the c.d.f..

A relationship between Gamma, Poisson and Negative Binomial Variates provides another method. If $G_k \sim$ Gamma(k), then Poisson($pG_k/[1-p]$) is a N.Binom(k,p) variate. Ahrens and Dieter (1974) have implemented this in Fortran (algorithm NG). The method is attractive, since procedures now exist for efficient sampling from the Gamma and Poisson distributions (with parameter values varying between calls) and whose execution times are bounded for all values of their parameters. Thus execution time for the Negative Binomial would also be bounded for all k and p. Further the method is applicable to non-integer k.

A limiting form for the Negative Binomial distribution as $k \to 0$ is the Logarithmic distribution, having p.m.f.,

$$p_x = f_X(x) = -\alpha^x/\{x\ \ell n(1-\alpha)\} \qquad (x = 1,2,\ldots)\ , \qquad (3.3.3)$$

where $0 < \alpha < 1$. Kemp (1981) proposes two strategies for generation. One is based on numerical inversion of the c.d.f., (algorithm LS). The c.d.f. is effectively updated during each search, using the recursion $p_x/p_{x-1} = \alpha(1-1/x)$. Instead of adding p_x to the cumulative q_{x-1}, a variant known as the "chop-down search" is employed, in which the random number is reduced by p_x. In this way one less variable (q_x) is employed in the algorithm, and one less assignment required per variate generated. The second strategy uses the property that if Y has c.d.f.,

$$F_Y(y) = \frac{\ell n(1-y)}{\ell n(1-\alpha)} \qquad (0 \le y \le \alpha)\ , \qquad (3.3.4)$$

and X has p.m.f.,

$$f_X(x) = (1-Y)Y^{x-1} \qquad (x = 1,2,\ldots)\ , \qquad (3.3.5)$$

then the unconditional distribution of X is Logarithmic. Given random numbers R_1 and R_2, inversion on (3.3.4)-(3.3.5) yields,

$$Y = 1 - (1-\alpha)^{R_1}\ , \qquad (3.3.6)$$

and

$$X = \langle 1 + (\ell n\ R_2/\ell n\ Y)\rangle \qquad (3.3.7)$$

respectively. Kemp implements this as algorithm LB. However, for parameter values $0 < \alpha \le 0.8$, the probability that the delivered value is either 1 or 2 is at least 0.695. Thus in a modified implementation (algorithm LK), pretest are incorporated which avoid the two logarithmic evaluations on a high proportion of occasions. For distributions where α is not close to one, the mean number of comparisons in the inversion

method is small as $E(X)$ is small, but rises steeply as $\alpha \to 1$. This is confirmed by the results from Kemp's timing experiments, and she recommends use of LK for $\alpha \gtrsim 0.90$, and LS for $\alpha \lesssim 0.90$.

3.4 Zeta Distribution

The Geometric and Logarithmic distributions are both candidates for modelling a discrete Variate with decreasing probabilities. A third discrete distribution with monotonically decreasing probabilities is the Zeta, having probability mass function,

$$f_X(x) = A\, x^{-(\rho+1)} \qquad (\rho > 0;\ x = 1,2,\ldots)\ , \qquad (3.4.1)$$

where $A = [\zeta(1+\rho)]^{-1}$, and $\zeta(.)$ denotes the Riemann Zeta function. The distribution has been used in such diverse areas as modelling the frequency of occurrence of specific words in linguistic texts (Zipf, 1949), the distribution of number of Insurance Policies held by individuals (Seal, 1947) and estimating the likelihood of rare events (Bendell and Samson, 1981). There is no guidance in the literature on how to generate Variates from this distribution, and this section describes two new approaches.

Since $E(X) = \zeta(\rho)/\zeta(1+\rho)$, the mean is infinite for $\rho \leq 1$. For $\rho > 1$, and not very close to 1, $E(X)$ is not too large and so numerical inversion of the c.d.f. becomes a practicable possibility. The mean number of comparisons for specimen values of ρ is shown in Table 3.1.

Table 3.1 Mean number of comparisons (E[X]) per generated variate using inversion, sampling efficiency (M^{-1}) using a truncated Pareto envelope, and values of $\zeta(1+\rho)$[††]

ρ	ε[†]	0.5	0.7	0.9	1.0	1.1	1.5	2.0	3.0
E(X)	∞	∞	∞	∞	∞	4.757	1.944	1.368	1.111
M^{-1}	0.667	0.500	0.444	0.391	0.366	0.342	0.258	0.178	0.080
$\zeta(1+\rho)$	ε^{-1}	2.60	2.053	1.750	1.6449	1.5602	1.3415	1.2021	1.0823

4.0	5.0	∞
1.044	1.019	1
0.034	0.014	0
1.0369	1.0173	1

Algorithm 3.1 below gives an implementation of the inversion method, incorporating "chop-down search", with successive probabilities calculated through the recursion,

$$f_X(x)/f_X(x-1) = \{(x-1)/x\}^{\rho+1} .$$

Algorithm 3.1

1. $p = A$.
2. generate $R \sim U(0,1)$, $X = 1$.
3. If $R > p$ go to 4, else deliver X.
4. $R = R - p$, $X = X + 1$, $p = p\{(X-1)/X\}^{\rho+1}$, go to 2.

As $\rho \to 1+$, the distribution becomes extremely long-tailed with $E(X) \to \infty$, and so an alternative to inversion must be found as follows.

---

† $\varepsilon \to 0+$.

†† For integral values of ρ exceeding 0, $\zeta(1+\rho)$ is obtained from Abramowitz and Stegun (1965, pp 811). Other values were computed by Mr G C Collie, Dundee College of Technology.

In order to generate from

$$f_N(n) = A\, n^{-(\rho+1)} \qquad (n = 1,2,\ldots), \tag{3.4.2}$$

it is sufficient to generate variates X from the continuous distribution,

$$f_X(x) = A\langle x+\tfrac{1}{2}\rangle^{-(\rho+1)} \qquad (x > \tfrac{1}{2}) , \tag{3.4.3}$$

and set $N = \langle X+\frac{1}{2}\rangle$. Variates may be generated from (3.4.3) using envelope rejection with a Pareto target distribution, truncated at $x = \frac{1}{2}$ and having c.d.f.

$$G_Y(x) = 1 - (2x)^{-\rho} \qquad (x > \tfrac{1}{2}) , \tag{3.4.4}$$

and p.d.f.,

$$g_Y(x) = 2^{-\rho}\rho\, x^{-(\rho+1)} \qquad (x > \tfrac{1}{2}) . \tag{3.4.5}$$

Given a random number R_1, inverting $G_Y(\cdot)$ yields

$$Y = (2R_1^{\,1/\rho})^{-1} . \tag{3.4.6}$$

From (3.4.3) and (3.4.5) we have,

$$\frac{f_X(x)}{g_Y(x)} = \left(\frac{2^\rho A}{\rho}\right)\left(\frac{x}{\langle x+\frac{1}{2}\rangle}\right)^{\rho+1} \qquad (x > \tfrac{1}{2}) , \tag{3.4.7}$$

which attains its maximum at $x = 1.5 - \varepsilon$, where $\varepsilon \to 0+$, giving,

$$M = \left(\frac{2^\rho A}{\rho}\right) 1.5^{(\rho+1)} . \tag{3.4.8}$$

Thus, given a second random number R_2, the acceptance condition becomes

$$R_2\, 1.5^{(\rho+1)} \le (Y/\langle Y+\tfrac{1}{2}\rangle)^{\rho+1} , \tag{3.4.9}$$

or

$$E \ge (\rho+1)(\ell n\ 1.5 - \ell n\ W) , \tag{3.4.10}$$

where E is a standard Exponential variate, and $W = Y/\langle Y+\frac{1}{2}\rangle$. Since $0.5 < W < 1.5$, effective bounds may be obtained for $\ell n\ W$. These are

$$W - 1 \geq \ln W \geq \begin{cases} 2(W-1)\ln 1.5 & (1 \leq W < 1.5) \\ 2(1-W)\ln 0.5 & (0.5 < W < 1). \end{cases} \tag{3.4.11}$$

We remark that when $\rho < 1$, large values of Y will occur frequently, leading to values of W close to 1. In this case the bounds become very tight. Using (3.4.11) an acceptance pre-test,

$$E \geq \begin{cases} (\rho+1)(3-2W)\ln 1.5 & (1 \leq W < 1.5) \\ (\rho+1)(\ln 6 + W \ln 0.25) & (0.5 < W < 1), \end{cases} \tag{3.4.12}$$

may be derived. If this is not satisfied a rejectance pre-test,

$$E < (\rho+1)(1-W+\ln 1.5) \tag{3.4.13}$$

is applied. Only if (3.4.12) and (3.4.13) both fail does the full test (3.4.10) need to be used. Algorithm 3.2 below gives an implementation of this rejection method.

Algorithm 3.2

1. generate two random numbers R_1, R_2.
2. $Y = (2R_1^{1/\rho})^{-1}$, $N = \langle Y+0.5 \rangle$, $W = Y/N$, $E = -\ln R_2$.
3. If $W \geq 1$ go to 5.
4. If $E \geq (\rho+1)(\ln 6 + W \ln 0.25)$ deliver N, else go to 6.
5. If $E \geq (\rho+1)(3-2W)\ln 1.5$ deliver N.
6. If $E < (\rho+1)(1-W+\ln 1.5)$ go to 1.
7. If $E \geq (\rho+1)([\ln 1.5] - \ln W)$ deliver N, else go to 1.

Since $A = \{\zeta(1+\rho)\}^{-1}$, the sampling efficiency is

$$M^{-1} = \frac{2\rho\ \zeta(1+\rho)}{3^{\rho+1}}. \tag{3.4.14}$$

Specimen values of M^{-1} are shown in Table 3.1. To establish the limiting behaviour of (3.4.14) as $\rho \to 0$ and $\rho \to \infty$, we note that

$$\zeta(1+\rho) = \sum_{x=1}^{\infty} x^{-(\rho+1)} = 1 + \int_{1.5}^{\infty} \langle x+\tfrac{1}{2} \rangle^{-(\rho+1)} dx$$

$$\leq 1 + \int_{1.5}^{\infty} (x-\tfrac{1}{2})^{-(\rho+1)} dx$$

$$= 1 + \rho^{-1} . \qquad (3.4.15)$$

Similarly,

$$\zeta(1+\rho) \geq 1 + \int_{1.5}^{\infty} (x+\tfrac{1}{2})^{-(\rho+1)} dx = 1 + 2^{-\rho}\rho^{-1} . \qquad (3.4.16)$$

Hence,

$$\rho + 2^{-\rho} \leq \rho\ \zeta(1+\rho) \leq 1 + \rho . \qquad (3.4.17)$$

From (3.4.14) and (3.4.17) we deduce that

$$\lim_{\rho \to 0} \{M^{-1}\} = \frac{2}{3} , \qquad (3.4.18)$$

and

$$\lim_{\rho \to \infty} \{M^{-1}\} = \lim_{\rho \to \infty} \{2\rho\ 3^{-(\rho+1)}\} = 0 . \qquad (3.4.19)$$

Table 3.1 suggests that the two algorithms will be complimentary in their performance. Inversion cannot be used for $\rho \leq 1$. Fortunately in this region the rejection sampling efficiency is at its highest. Rejection becomes progressively less attractive as ρ increases above 1, due to the decreasing sampling efficiency. In this region however $E(X)$ reduces with increasing ρ, indicating that the inversion method is likely to be quite efficient.

Algorithms 3.1 and 3.2 were programmed as Fortran subroutines IZ2 and IZ1. These appear in Appendix 1 under program names ZIPINF.FOR and ZIPFRP.FOR respectively. For $\rho < .1$, it is quite possible for N or even Y to overflow in algorithm 3.2. Thus a check is made in IZ1 to determine whether R_1 is small enough to cause overflow of N. If

it is, then $\ell n\ Y$, rather than Y is computed. Fortunately, for such large values of Y, Y/N is extremely close to 1, so the acceptance condition (3.4.9) can be replaced by $R_2 < \left(\frac{2}{3}\right)^{\rho+1}$. If this condition is satisfied then $\ell n\ Y$ (being extremely close to $\ell n\ N$) is returned.

The programs were executed on a DEC-20 computer, using random number generator RAN(•). This takes approximately 13 μsecs to generate one random number. Its statistical adequacy is reported in Appendix 2. The mean execution time per generated Zeta variate was obtained by generating samples of size 1000 and 11000 and taking the difference in times. The timing experiments were conducted when demand by other users was low, so as to reduce variation due to multiprogramming. Under these conditions the maximum variation appeared to be of the order of 7%. Although not ideal, it is felt that if the speeds of two algorithms are comparable within these limits, other criteria such as ease of implementation and storage requirements would be more influential in determining which is the best algorithm for the specified machine.

The results are summarised in Table 3.2. Timings for the inversion method were not obtained for $\rho \leq 1$, as the analysis above shows that the mean number of comparisons required is infinite. The results obtained tend to confirm the conclusions based on the analysis of $E(X)$ and M^{-1}. For $\rho \leq 1$ rejection is the only method that can be used. The pre-tests made Algorithm 3.2 approximately 12% faster than a version with no pre-tests. For $\rho \gtrsim 1.1$, the inversion method is faster and its superiority increases rapidly with increasing ρ.

Table 3.2: Mean time (μsecs) to generate one Zeta variate on a DEC-20 computer.

ρ	0.001	0.1	0.3	0.5	0.7	0.9	1.0	1.01	1.05	1.1	1.5
Inversion	-	-	-	-	-	-	-	781	638	514	161
Rejection	112	330	394	446	500	567	605	603	621	646	855

2	3	4	5
77	41	31	25
1246	2736	6444	15080

It is of interest to consider how the methods would compare if a slower random number generator were used. Consider for example a generator taking 65 μsecs per random number. Table 3.1 indicates that at $\rho = 1.1$, a mean of (2/0.342) and 1 random number per variate are required for rejection and inversion respectively. Thus the rejection and inversion execution times would rise to approximately 950 and 566 μsecs respectively.

A conservative recommendation for the current implementation would be to use rejection for $\rho \leq 1.1$, otherwise to use inversion. However, one minor advantage of the rejection method is that there is no necessity to calculate the constant A, and so this might justify its use for certain values of ρ in excess of 1.1 .

CHAPTER 4

Generation from multivariate continuous distributions

Most of the developments in variate generation have been concerned with univariate distributions, but little guidance exists for the experimenter who wishes to generate variates from multivariate distributions. Exceptions include the Multivariate Normal which is well documented since there is a convenient transformation to independent form as described for example in Fishman (1978, pp 464-466). Deak (1980) has developed an efficient method for the Multivariate Normal, but at the expense of generating vectors which are not independent. Kemp and Loucas (1978) describe methods for Bivariate Poisson and Hermite distributions, employing inversion of the distribution function and also exploitation of the stochastic structure of specific distributions. Hull (1977) employs a method based on an approximate transformation to bivariate Normal form. In this chapter three general methods for generating variates from multivariate continuous distributions are presented, and then illustrated with reference to three specific distributions.

4.1 General Methods

4.1.1 Method of Conditional Distributions

Suppose the p.d.f. of a random vector $\underset{\sim}{X}$ is $f_{\underset{\sim}{X}}(.)$ where $\underset{\sim}{X}' = (X_1, X_2, \ldots, X_n)$. Then this method generates the following random variables in sequence:

$$X_1; X_2|X_1; X_3|X_2, X_1; \ldots\ldots; X_n|X_{n-1}, X_{n-2}, \ldots, X_1 \quad .$$

The success of this method depends on the ease with which expressions for the conditional distributions may be obtained, and the speed at which variates may be sampled from them.

4.1.2 Transformation to Independent Form

In certain cases it may be possible to obtain a transformation $\underset{\sim}{Y} = \underset{\sim}{h}(\underset{\sim}{X})$ such that the components of $\underset{\sim}{Y}$ are independent random variables. The practicability of such a method depends on whether variates can easily be sampled from the marginal distributions of $\underset{\sim}{Y}$ and whether the inverse $\underset{\sim}{X} = \underset{\sim}{h}^{-1}(\underset{\sim}{Y})$ is readily obtainable. For the Multivariate Normal the method works well since the inverse transformation takes the convenient form

$$\underset{\sim}{X} = \underset{\sim}{\mu} + \underset{\sim}{C}\,\underset{\sim}{Y} \qquad (4.1.1)$$

where the components Y_i $(i = 1,2,\ldots,n)$ of $\underset{\sim}{Y}$ are independent standard Normal deviates, $E(\underset{\sim}{X}) = \underset{\sim}{\mu}$, $\underset{\sim}{C}\underset{\sim}{C}' = \underset{\sim}{V}$, $\underset{\sim}{V}$ being the variance-covariance matrix for $\underset{\sim}{X}$. It is evident that a method due to Ronning (1977) for generation from a Multivariate Gamma distribution also falls into this category, the correlated Gamma variates being constructed from sums of independent Gamma variates.

4.1.3 Rejection Method

Generalisation of the envelope rejection method from univariate to multivariate application does not appear to have been exploited in the past. However the following algorithm will generate random vectors $\underset{\sim}{X}$ from a multivariate p.d.f. $f_{\underset{\sim}{X}}(.)$ using a target distribution $g_{\underset{\sim}{Y}}(.)$.

Algorithm 4.1

0. Select M, such that $M \geq \text{Max}[f_{\underset{\sim}{X}}(\underset{\sim}{x})/g_{\underset{\sim}{Y}}(\underset{\sim}{x})]$.

1. generate $\underset{\sim}{Y}$ from $g_{\underset{\sim}{Y}}(.)$ and $R \sim U(0,1)$.

2. If $RM \geq f_{\underset{\sim}{X}}(\underset{\sim}{Y})/g_{\underset{\sim}{Y}}(\underset{\sim}{Y})$ go to 1, else deliver $\underset{\sim}{Y}$.

To demonstrate the validity of the algorithm it is only necessary to show that the c.d.f. of $\underset{\sim}{Y}$ conditional upon $RM < f_{\underset{\sim}{X}}(\underset{\sim}{Y})/g_{\underset{\sim}{Y}}(\underset{\sim}{Y})$ is $F_{\underset{\sim}{X}}(.)$:

$$P\left\{\underset{\sim}{Y} \leq \underset{\sim}{x} \middle| RM < f_{\underset{\sim}{X}}(\underset{\sim}{Y})/g_{\underset{\sim}{Y}}(\underset{\sim}{Y})\right\} = \frac{P\left\{RM<f_{\underset{\sim}{X}}(\underset{\sim}{Y})/g_{\underset{\sim}{Y}}(\underset{\sim}{Y});\ \underset{\sim}{Y} \leq \underset{\sim}{x}\right\}}{P\left\{RM<f_{\underset{\sim}{X}}(\underset{\sim}{Y})/g_{\underset{\sim}{Y}}(\underset{\sim}{Y})\right\}} . \qquad (4.1.2)$$

Since $f_{\underset{\sim}{X}}(\underset{\sim}{Y})/Mg_{\underset{\sim}{Y}}(\underset{\sim}{Y}) \leq 1$, and $R \sim U(0,1)$, this becomes

$$\frac{\int_{\underset{\sim}{y}\leq\underset{\sim}{x}} g_{\underset{\sim}{Y}}(\underset{\sim}{y})d\underset{\sim}{y} \int_0^{f_{\underset{\sim}{X}}(\underset{\sim}{y})/Mg_{\underset{\sim}{Y}}(\underset{\sim}{y})} dr}{\int_{\underset{\sim}{y}\leq\underset{\sim}{\infty}} g_{\underset{\sim}{Y}}(\underset{\sim}{y})d\underset{\sim}{y} \int_0^{f_{\underset{\sim}{X}}(\underset{\sim}{y})/Mg_{\underset{\sim}{Y}}(\underset{\sim}{y})} dr}$$

$$= \frac{M^{-1} \int_{\underset{\sim}{y}\leq\underset{\sim}{x}} f_{\underset{\sim}{X}}(\underset{\sim}{y})d\underset{\sim}{y}}{M^{-1} \int_{\underset{\sim}{y}\leq\underset{\sim}{\infty}} f_{\underset{\sim}{X}}(\underset{\sim}{y})d\underset{\sim}{y}}$$

$$= \frac{M^{-1} F_{\underset{\sim}{X}}(\underset{\sim}{x})}{M^{-1}} \qquad (4.1.3)$$

$$= F_{\underset{\sim}{X}}(\underset{\sim}{x}) , \qquad (4.1.4)$$

as required. The denominator of (4.1.3) shows that the sampling efficiency is M^{-1}.

The choice of distribution $g_{\underset{\sim}{Y}}(\cdot)$ for first-stage sampling is governed by several considerations. Firstly, it should be easy to sample from $g_{\underset{\sim}{Y}}(\cdot)$ and since it is the dependence between the components of $\underset{\sim}{X}$ which usually precludes direct generation, it seems reasonable to select $\underset{\sim}{Y}$ in such a way that the components are independently distributed. Secondly, it is preferable if $g_{\underset{\sim}{Y}}(\cdot)$ "imitates" the function $f_{\underset{\sim}{X}}(\cdot)$ as far as is practicable, in order that M is as close to unity as possible. Finally the sampling efficiency M^{-1} will be improved if the maximum of $f_{\underset{\sim}{X}}(\underset{\sim}{x})/g_{\underset{\sim}{Y}}(\underset{\sim}{x})$ can be found rather than just an upper bound as implied in step 0 of the algorithm. The first two requirements suggest that in many cases a suitable choice for $g_{\underset{\sim}{Y}}(\cdot)$ is one obtained by considering the product of the marginals of $\underset{\sim}{X}$:

$$g_{\underset{\sim}{Y}}(\cdot) = \prod_{i=1}^{n} f_{X_i}(\cdot) \quad . \tag{4.1.5}$$

In the limiting case of independent components for $\underset{\sim}{Y}$, this choice ensures that the proportion of accepted variates is one.

4.2 Multivariate Normal Distribution

Using the notation of section 4.1.2, a row vector $\underset{\sim}{X}'$ is distributed as Multivariate Normal if its p.d.f. is

$$f_{\underset{\sim}{X}}(\underset{\sim}{x}) = (2\pi)^{-\frac{1}{2}n} |\underset{\sim}{V}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{\mu})'\underset{\sim}{V}^{-1}(\underset{\sim}{x}-\underset{\sim}{\mu})} \quad , \tag{4.2.1}$$

where $\underset{\sim}{V}$ is any positive definite real symmetric matrix.

To motivate a method based on conditional distributions denote

$\underset{\sim}{X}' = (\underset{\sim}{X}_1', \underset{\sim}{X}_2')$ where $\underset{\sim}{X}_1' = (X_1, \ldots, X_j)$ and $\underset{\sim}{X}_2' = (X_{j+1}, \ldots, X_n)$,

$\underset{\sim}{\mu}' = (\underset{\sim}{\mu}_1', \underset{\sim}{\mu}_2')$ where $\underset{\sim}{\mu}_1' = (\mu_1, \ldots, \mu_j)$ and $\underset{\sim}{\mu}_2' = (\mu_{j+1}, \ldots, \mu_n)$ and

$$\underset{\sim}{V}^{-1} = \begin{pmatrix} \underset{\sim}{A}_{11} & \underset{\sim}{A}_{12} \\ \underset{\sim}{A}_{21} & \underset{\sim}{A}_{22} \end{pmatrix} ,$$

where partitioning is at the j^{th} row and column. The distribution of $\underset{\sim}{X}_2$ given $\underset{\sim}{X}_1$ is itself multivariate Normal with mean $\underset{\sim}{\mu}_2' - (\underset{\sim}{X}_1-\underset{\sim}{\mu}_1)'\underset{\sim}{A}_{12}\underset{\sim}{A}_{22}^{-1}$ and variance-covariance matrix $\underset{\sim}{A}_{22}^{-1}$, (see for example Johnson & Kotz, 1972, pp 41). Hence, given $X_1,\ldots,X_j$ the next variate X_{j+1} may be generated via

$$X_{j+1} = \mu_{j+1} - \sum_{i=1}^{j} (X_i-\mu_i)(\underset{\sim}{A}_{12}\underset{\sim}{A}_{22}^{-1})_{i1} + (\underset{\sim}{A}_{22}^{-1})_{11} Z_{j+1}, \qquad (4.2.2)$$

where Z_{j+1} is a standard Normal deviate. In principle (4.2.2) could be used recursively to generate the vector $\underset{\sim}{X}$.

A more convenient method is to employ a transformation to independent form as specified in (4.1.1). The product of the lower triangular matrix $\underset{\sim}{C}$ and the vector $\underset{\sim}{Y}$ is very much simpler than use of the recursion (4.2.2), while the only preliminary work required is calculation of the matrix $\underset{\sim}{C}$.

A third possibility is use of the rejection method with a target distribution attempting to satisfy the requirements described in section 4.1.3 . Such a distribution is

$$g_{\underset{\sim}{Y}}(\underset{\sim}{x}) = (2\pi)^{-\frac{1}{2}n} |\underset{\sim}{\beta}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{\mu})'\underset{\sim}{\beta}^{-1}(\underset{\sim}{x}-\underset{\sim}{\mu})} , \qquad (4.2.3)$$

where

$$\underset{\sim}{\beta} = \begin{pmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_n \end{pmatrix} . \qquad (4.2.4)$$

In this case,

$$\frac{f_{\underset{\sim}{X}}(\underset{\sim}{x})}{g_{\underset{\sim}{Y}}(\underset{\sim}{x})} = \frac{|\underset{\sim}{\beta}|^{\frac{1}{2}}}{|\underset{\sim}{V}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{\mu})'(\underset{\sim}{V}^{-1}-\underset{\sim}{\beta}^{-1})(\underset{\sim}{x}-\underset{\sim}{\mu})} \quad . \tag{4.2.5}$$

We select $\underset{\sim}{\beta}$, such that $(\underset{\sim}{V}^{-1}-\underset{\sim}{\beta}^{-1})$ is positive definite. This is achieved by ensuring that every diagonal element of $\underset{\sim}{\beta}^{-1}$ is no larger than the smallest eigenvalue of $\underset{\sim}{V}^{-1}$, or equivalently, that every diagonal element of $\underset{\sim}{\beta}$ is no smaller than the largest eigenvalue of $\underset{\sim}{V}$. Representing the eigenvalues of $\underset{\sim}{V}$ in ascending order of values as $\lambda_1,\lambda_2,\ldots,\lambda_n$, a choice of $\beta_i \geq \lambda_n$ for all i, gives

$$M = \underset{\underset{\sim}{x}}{\text{Max}} \left[f_{\underset{\sim}{X}}(\underset{\sim}{x})/g_{\underset{\sim}{Y}}(\underset{\sim}{x})\right] = |\underset{\sim}{\beta}|^{\frac{1}{2}}/|\underset{\sim}{V}|^{\frac{1}{2}} \quad . \tag{4.2.6}$$

The acceptance condition becomes

$$R < e^{-\frac{1}{2}(\underset{\sim}{Y}-\underset{\sim}{\mu})'(\underset{\sim}{V}^{-1}-\underset{\sim}{\beta}^{-1})(\underset{\sim}{Y}-\underset{\sim}{\mu})}$$

or

$$E > \tfrac{1}{2}(\underset{\sim}{Y}-\underset{\sim}{\mu})'(\underset{\sim}{V}^{-1}-\underset{\sim}{\beta}^{-1})(\underset{\sim}{Y}-\underset{\sim}{\mu}) \quad , \tag{4.2.7}$$

where E is a standard Exponential Variate. From (4.2.6) we note that sampling efficiency is maximised by choosing $\beta_i = \lambda_n$, for all i.

The method may be illustrated with reference to a Bivariate Normal distribution with

$$\underset{\sim}{V} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \quad .$$

The eigenvalues are $\lambda_1 = 1 - |\rho|$ and $\lambda_2 = 1 + |\rho|$. The ideal choice is $\beta_1 = \beta_2 = 1 + |\rho|$, giving

$$M = \frac{1 + |\rho|}{(1-|\rho|^2)^{\frac{1}{2}}} = \left[\frac{1 + |\rho|}{1 - |\rho|}\right]^{\frac{1}{2}} \quad , \tag{4.2.8}$$

indicating that the sampling efficiency $\to 0$ as $|\rho| \to 1$.

For higher dimensional cases, finding the largest eigenvalue λ_n routinely poses some problem.. A suitable bound however is

$$\lambda_n \leq \underset{i}{\text{Max}} \{ \sum_j |V_{ij}| \} , \qquad (4.2.9)$$

(Gerchgorin's first theorem). To illustrate, consider generation from a tri-variate distribution with

$$\underset{\sim}{V} = \begin{pmatrix} 1 & 0.433 & -0.385 \\ 0.433 & 1 & 0.083 \\ -0.385 & 0.083 & 1 \end{pmatrix} . \qquad (4.2.10)$$

Using (4.2.9), the largest eigenvalue,

$$\lambda_3 \leq 1.818 . \qquad (4.2.11)$$

Selecting $\beta_i = 1.818$ $(i = 1,2,3)$, gives

$$M = \frac{(1.818)^{1.5}}{|\underset{\sim}{V}|^{\frac{1}{2}}} = \frac{(1.818)^{1.5}}{0.794} = 3.09 , \qquad (4.2.12)$$

or a sampling efficiency of 0.324 .

It is clear that the rejection method applied to the Multivariate Normal is in no way competitive to method 2 based on the well-known transformation to independent form. Calculation of the right hand side of acceptance condition (4.2.7) is significantly more involved than calculation of $\underset{\sim\sim}{CY}$ in (4.1.1) and several such calculations may be required to generate 1 variate. Further the rejection rate becomes unacceptably high if the correlation structure is such that $|\underset{\sim}{V}| \doteq 0$.

4.3 A Bivariate Exponential Distribution

Consider a bivariate Exponential distribution, Gumbel (1960), with p.d.f.,

$$f_{\underset{\sim}{X}}(\underset{\sim}{x}) = e^{-(x_1+x_2+\theta x_1 x_2)} \{(1+\theta x_1)(1+\theta x_2) - \theta\} \quad (\underset{\sim}{x} \geq 0;\ 0 \leq \theta \leq 1). \qquad (4.3.1)$$

The marginal densities of X_1 and X_2 are standard Exponential and therefore the conditional density $X_2|X_1$ is

$$f_{X_2|X_1}(x_2|x_1) = e^{-x_2(1+\theta x_1)}\{(1+\theta x_1)(1+\theta x_2) - \theta\}$$

$$= (1+\theta x_1-\theta)e^{-x_2(1+\theta x_1)} + \theta x_2(1+\theta x_1)e^{-x_2(1+\theta x_1)}$$

$$= \left(\frac{1+\theta x_1-\theta}{1+\theta x_1}\right) f_{Z_1|X_1}(x_2|x_1) + \left(\frac{\theta}{1+\theta x_1}\right) f_{Z_2|X_1}(x_2|x_1), \qquad (4.3.2)$$

where

$$f_{Z_1|X_1}(x_2|x_1) = (1+\theta x_1)e^{-x_2(1+\theta x_1)}, \qquad (4.3.3)$$

and

$$f_{Z_2|X_1}(x_2|x_1) = (1+\theta x_1)^2 x_2 e^{-x_2(1+\theta x_1)}, \qquad (4.3.4)$$

which are Exponential, mean $(1+\theta x_1)^{-1}$, and Gamma, with shape parameter 2 and mean $2(1+\theta x_1)^{-1}$ respectively. Noting that the conditional density is the weighted sum of two other conditional densities, the method of conditional distributions gives the following algorithm.

Algorithm 4.2

1. Generate $R \sim U(0,1)$ and E_1, E_2, independent standard Exponential variates.
2. $X_1 = E_1$.
3. If $R > \frac{\theta}{1+\theta X_1}$ then $X_2 = \frac{E_2}{1+\theta X_1}$, else generate E_3 a standard Exponential variate and

$$X_2 = \frac{E_2+E_3}{1+\theta X_1}.$$

The transformation to independent form appears to fail for this distribution, since no convenient transformation is apparent. A rejection method would use a target distribution,

$$g_{\underset{\sim}{Y}}(\underset{\sim}{y}) = e^{-(y_1+y_2)} , \qquad (4.3.5)$$

whence

$$\frac{f_{\underset{\sim}{X}}(\underset{\sim}{x})}{g_{\underset{\sim}{Y}}(\underset{\sim}{x})} = e^{-\theta x_1 x_2} \{(1+\theta x_1)(1+\theta x_2) - \theta\} . \qquad (4.3.6)$$

Unfortunately (4.3.6) is unbounded as $x_1 \to 0$ and $x_2 \to \infty$, giving $M = \infty$, a 100% rejection rate. We conclude therefore that the method of conditional distributions is the only one of the three methods that can be used in this case.

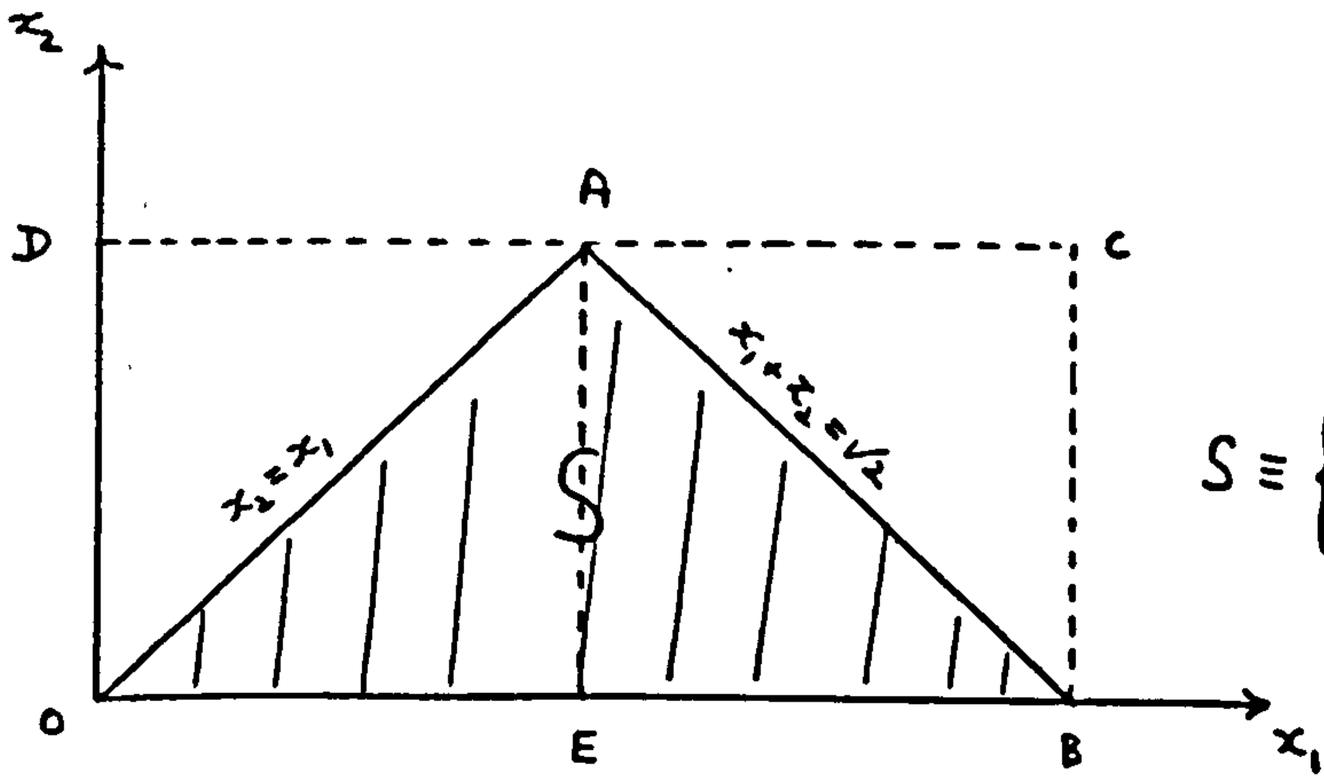
4.4 A Non-standard Bivariate Distribution

Consider a bivariate distribution with p.d.f.,

$$f_{\underset{\sim}{X}}(x_1,x_2) = \begin{cases} 8x_1/\{3(x_1+x_2)\} & (\underset{\sim}{x} \in S) \\ 0 & (\underset{\sim}{x} \notin S), \end{cases} \qquad (4.4.1)$$

where S is the region shown in Figure 4.1 .

Figure 4.1 Region S for p.d.f. (4.4.1)



$$S \equiv \left\{ \begin{array}{l} 0 \le x_2 \le x_1 \text{ when } 0 < x_1 \le \frac{1}{\sqrt{2}} \\ 0 \le x_2 \le \sqrt{2} - x_1 \text{ when } \frac{1}{\sqrt{2}} < x_1 \le \sqrt{2} \end{array} \right\}$$

The method of conditional distributions fails in this case, because although expressions for the marginal distribution functions can be found, it is not possible to invert these analytically.

However transformation to independent form is readily obtained by considering $\underset{\sim}{U}' = (U_1, U_2)$, where $U_1 = X_2/X_1$ and $U_2 = X_1 + X_2$. The p.d.f. of $\underset{\sim}{U}$ is

$$g_{\underset{\sim}{U}}(u_1, u_2) = \frac{8u_2}{3(1+u_1)^3} \qquad (0 \le u_1 \le 1;\ 0 \le u_2 \le \sqrt{2}). \tag{4.4.2}$$

Thus U_1 and U_2 are independently distributed with marginal distribution functions

$$G_{U_1}(u_1) = \frac{4}{3}\{1 - (1+u_1)^{-2}\} \qquad (0 \le u_1 \le 1) \tag{4.4.3}$$

and

$$G_{U_2}(u_2) = \frac{1}{2}u_2^2 \qquad (0 \le u_2 \le \sqrt{2})\ . \tag{4.4.4}$$

Using the inversion method for generation from univariate distributions we set,

$$R_1 = G_{U_1}(U_1) \qquad \text{and} \qquad R_2 = G_{U_2}(U_2)\ , \tag{4.4.5}$$

where R_1 and R_2 are independently $U(0,1)$. Solving for U_1 and U_2 we obtain,

$$U_1 = (1 - \tfrac{3}{4}R_1)^{-\frac{1}{2}} - 1 \tag{4.4.6}$$

and

$$U_2 = (2R_2)^{\frac{1}{2}}\ . \tag{4.4.7}$$

On inverting the transformation, we obtain

$$X_1 = \frac{U_2}{1 + U_1} = \{2R_2(1 - \tfrac{3}{4}R_1)\}^{\frac{1}{2}} \tag{4.4.8}$$

and

$$X_2 = U_1X_1 = \{(1 - \tfrac{3}{4}R_1)^{-\frac{1}{2}} - 1\}X_1\ , \tag{4.4.9}$$

which provide a direct method for generation of random vectors $\underset{\sim}{X}$.

A rejection-based procedure is also possible, but difficulty in generation from the marginal distributions suggests that a more suitable form for the target distribution is

$$g_{\underset{\sim}{Y}}(y_1,y_2) = \begin{cases} 2 & (y \in S) \\ 0 & (y \notin S) \end{cases}, \qquad (4.4.10)$$

giving

$$M = \max_{\underset{\sim}{x}} \left\{ \frac{f_{\underset{\sim}{X}}(\underset{\sim}{x})}{g_{\underset{\sim}{Y}}(\underset{\sim}{x})} \right\} = \frac{4.}{3},$$

which results in algorithm 4.3 below.

Algorithm 4.3

1. generate 3 random numbers R_1, R_2, R_3.

2. Sample $\underset{\sim}{Y}$ from $g_{\underset{\sim}{Y}}(\cdot,\cdot)$:

   $Y_1 = \sqrt{2}\, R_1$, $Y_2 = R_2/\sqrt{2}$.

   If $Y_1 + Y_2 > \sqrt{2}$ then $Y_1 = (3/\sqrt{2}) - Y_1$, $Y_2 = (1/\sqrt{2}) - Y_2$

   and go to 3.

   If $Y_2 - Y_1 > 0$ then $Y_1 = (1/\sqrt{2}) - Y_1$ and $Y_2 = (1/\sqrt{2}) - Y_2$.

3. If $R_3 > Y_1/(Y_1+Y_2)$ go to 1, else deliver $\underset{\sim}{X}' = (Y_1,Y_2)$.

Step 2 of the algorithm generates points uniformly within the rectangle OBCD. Points lying within OAB are used directly in step 3, as prospective sample variates. Points lying in OAD or BCA are transformed to points in OEA and EBA respectively, before being used as prospective variates. In this way all points within the rectangle are used as prospective variates.

The methods based on (4.4.8)-(4.4.9) and Algorithm 4.3 were programmed as Fortran subroutines BVAR1 and BVAR2, and appear in Appendix 3 under program names MUL3.FOR and MUL1.FOR respectively. Timing comparisons were made on a DEC 20 computer in the manner described in section 3.4 . Transformation to independent form and the rejection method took a mean of 106 and 92 μsecs, respectively. A contributory factor to the slightly faster speed of the rejection method is that no square roots are involved.

It is apparent from these illustrative examples that some general methods for multivariate generation can be identified. In particular, we are not aware of any reference in the literature to the use of a rejection method with target distribution constructed from the product of marginals. It is not claimed that such a method is efficient in all cases, rather that the availability of methods for a particular distribution may be limited as to make the rejection method worthy of consideration.

CHAPTER 5

Simulating First Passage Times in a Wiener Process

5.1 Introduction

Consider a Wiener (Gaussian or diffusion) process with drift μ (>0), and variance of displacement σ^2 per unit time. Let $X(t)$ denote the displacement at time t, with $X(0) = 0$. Then $X(t) \sim N(\mu t, \sigma^2 t)$. Suppose there is an absorbing barrier at $x = a$ (>0). The problem considered in this chapter is how to generate the time to absorption (first passage time) in such a process. The p.d.f. of first passage time T is

$$f_T(t) = \frac{a\, e^{-(a-\mu t)^2/2\sigma^2 t}}{\sigma\sqrt{(2\pi t^3)}} \qquad (t > 0)\ , \qquad (5.1.1)$$

the Inverse Gaussian distribution, with mean (a/μ) and variance $(a\,\sigma^2/\mu^3)$.

The distribution arises in first passage time problems in Brownian motion. Wald (1947, pp 191-194) derives the p.d.f.

$$f_X(x) = \left(\frac{\phi}{2\pi}\right)^{\frac{1}{2}} e^{\phi}\, x^{-\frac{3}{2}}\, e^{-\{\frac{1}{2}\phi(x+x^{-1})\}} \qquad (x > 0), \qquad (5.1.2)$$

in the context of sample size in certain sequential probability ratio tests. (5.1.2) is the Standardised Wald distribution which is obtained from (5.1.1) when $a = \mu = 1$ and $\phi = \sigma^{-2}$. In an Operational Research context, the author has previously found it necessary to simulate Inventory depletion times, starting from a fixed level of inventory a, where the net rate of depletion is Normally distributed with mean μ and variance σ^2.

(5.1.1) is a three parameter distribution. From a simulation viewpoint, we note that we may sample from the one parameter distribution, (5.1.2), where $\phi = a\mu/\sigma^2$, and then set $T = aX/\mu$. This transformation simplifies the construction of any generator, and also aids our understanding of its performance which will be related to the value of the parameter ϕ only. Henceforth the problem will be to generate variates from the standardised Wald distribution, (5.1.2).

One obvious approach to generation is to use a stochastic model method and simulate the process directly. Thus, given a suitably small time increment δt,

$$X(t+\delta t) \doteq X(t) + \delta t + Z(\delta t/\phi)^{\frac{1}{2}} , \qquad (5.1.3)$$

where $Z \sim N(0,1)$. (5.1.3) is then applied recursively until $X(t) \geq 1$ for the first time. This idea was not pursued. The principle objections to it are that it yields approximate variates only (although the accuracy will improve as δt is made smaller), and that a large number of Normal deviates per generated variate will be required, particularly if high accuracy is desired. The low efficiency is to be expected, since information about the entire realisation $\{X(t) : X(t) \leq 1\}$ is being provided, but most of it is not used.

An alternative is to invert the c.d.f.. Johnson and Kotz (1970a, pp 141) give the c.d.f. as

$$P(X \leq x) = \Phi\{(x-1)\sqrt{(\phi/x)}\} + e^{2\phi}\Phi\{-(x+1)\sqrt{(\phi/x)}\} , \qquad (5.1.4)$$

where $\Phi(\cdot)$ is the Standard Normal c.d.f.. Thus, given a random number R, it is necessary to solve the equation,

$$\Phi\{(X-1)\sqrt{(\phi/X)}\} + e^{2\phi}\Phi\{-(X+1)\sqrt{(\phi/X)}\} = R . \qquad (5.1.5)$$

Analytical solution of (5.1.5) does not seem feasible, and so this method was not explored further.

Shuster (1968) notes that variates $Y\cdot$ having p.d.f.,

$$f_Y(x) = \frac{\lambda^{\frac{1}{2}} e^{-\lambda(x-d)^2/2d^2x}}{\sqrt{(2\pi x^3)}} \qquad (x > 0) \ , \qquad (5.1.6)$$

are related to chi-squared variates with one degree of freedom via

$$\lambda(Y-d)^2/(d^2Y) \sim \chi^2_{(1)} = \nu_0 \quad \text{say} \ . \qquad (5.1.7)$$

Given roots,

$$\begin{aligned} Y_1 &= d + \frac{d^2\nu_0}{2\lambda} - \frac{d}{2\lambda}(4d\lambda\nu_0 + d^2\nu_0^2)^{\frac{1}{2}} \ , \\ Y_2 &= d^2/Y_1 \ , \end{aligned} \qquad (5.1.8)$$

of (5.1.7), Michael, Schucany and Haas (1976) show that by selecting Y_1 with probability $d/(d+Y_1)$ and Y_2 with probability $Y_1/(d+Y_1)$, exact Inverse Gaussian variates may be generated. The method appears attractive, in that 1 chi-squared variate and 1 random number only are required per variate generated, with no rejection step involved. In view of the comment made regarding the relative merits of sampling from the three parameter and one parameter distributions, it would seem best to apply this method to the standardised Wald distribution, and then to use the necessary transformation. Setting $\lambda = \phi$ and $d = 1$ in (5.1.6) gives the Standardised Wald distribution. Thus a convenient generation method is to return

$$X_1 = 1 + \{\nu_0/(2\phi)\} - (4\phi\nu_0+\nu_0^2)^{\frac{1}{2}}/2\phi \qquad (5.1.9)$$

with probability $1/(1+X_1)$ and

$$X_2 = X_1^{-1} \qquad (5.1.10)$$

with probability $X_1/(1+X_1)$.

The fourth method considered is based on an application of the ratio of uniforms method described in section 1.3.3 . This forms the major part of this chapter and its performance is compared to the chi-squared transformation method. A secondary aim of the chapter is to draw attention to the role of the reciprocal distribution in any ratio of uniforms method. We are not aware of any previous recognition of this relationship within the literature.

5.2 A theorem facilitating the determination of an enclosing region in the ratio of uniforms method.

To develop an algorithm based on the ratio of uniform deviates, we note from p.d.f. (5.1.2) and the notation of section 1.3.3, that the acceptance region is given by

$$C = \{(u,v) : 0 \le \mu \le \left(\frac{\phi}{2\pi}\right)^{\frac{1}{4}} e^{\phi/2} \left(\frac{v}{u}\right)^{-\frac{3}{4}} e^{-\frac{1}{4}\left(\frac{v}{u}+\frac{u}{v}\right)} ; 0 \le \frac{v}{u} \}$$

$$= \{(u,v) : 0 \le u \le \left(\frac{\phi}{2\pi v^3}\right) e^{2\phi} e^{-\phi\left(\frac{v}{u}+\frac{u}{v}\right)} ; 0 \le \frac{v}{u} \}. \qquad (5.2.1)$$

The first step is to find an enclosing rectangle,

$$D = \{(u,v) : 0 \le u \le u_+ ; -v_- \le v \le v_+ ; 0 \le \frac{v}{u} \} . \qquad (5.2.2)$$

To facilitate the determination of u_+, v_+ and v_-, we use the following theorem:

Theorem

Let $f_X(\cdot)$ and $f_Y(\cdot)$ be the p.d.f.'s of random variables X (having domain S) and $Y = X^{-1}$ respectively. Let C be the region,

$$\{(u,v) : 0 \le u \le f_X^{\frac{1}{2}}(v/u) \; ; \; \frac{v}{u} \in S\} \; .$$

Let $u_+ = \underset{x}{\text{Max}} \{f_X^{\frac{1}{2}}(x)\}$, $v_+ = \underset{y \ge 0}{\text{Max}} [f_Y^{\frac{1}{2}}(y)]$ and $v_- = \underset{y<0}{\text{Max}} \{f_Y^{\frac{1}{2}}(y)\}$.

Then C is enclosed by the region

$$D \equiv \{(u,v) : 0 \le u \le u_+ \; ; \; -v_- \le v \le v_+ \; ; \; v/u \in S\} \; .$$

Proof

The c.d.f. of Y is

$$F_Y(y) = \begin{cases} P(X \ge y^{-1}) + P(X < 0) & (y \ge 0) \\ P(y^{-1} \le X < 0) & (y < 0) \; , \end{cases} \qquad (5.2.3)$$

giving p.d.f.

$$f_Y(y) = \begin{cases} y^{-2} \; f_X(y^{-1}) & (y^{-1} \in S) \\ 0 & (\text{elsewhere}) \; . \end{cases} \qquad (5.2.4)$$

Thus any point (u,v) satisfying

$$0 \le u \le f_X^{\frac{1}{2}}(v/u) \quad ; \quad v/u \in S \; , \qquad (5.2.5)$$

also satisfies

$$0 \le u \le \{(\tfrac{u}{v})^2 \; f_Y(\tfrac{u}{v})\}^{\frac{1}{2}} \quad ; \quad v/u \in S \; , \qquad (5.2.6)$$

or

$$|v| \le f_Y^{\frac{1}{2}}(u/v) \quad ; \quad v/u \in S \; . \qquad (5.2.7)$$

Since

$$v_+ \ge f_Y^{\frac{1}{2}}(\tfrac{u}{v}) \quad \text{for all} \quad u \ge 0 \quad \text{and} \quad v \ge 0 \; , \qquad (5.2.8)$$

and

$$v_- \ge f_Y^{\frac{1}{2}}(\tfrac{u}{v}) \quad \text{for all} \quad u \ge 0 \quad \text{and} \quad v < 0 \; , \qquad (5.2.9)$$

it follows that the point (u,v) also satisfies

$$-v_- \le v \le v_+ \quad ; \quad \frac{v}{u} \in S \quad , \qquad (5.2.10)$$

showing that every point in C is contained in D.

The theorem provides a convenient method of enclosing C, providing the modal values of $f_X(\cdot)$ and $f_Y(\cdot)$ (the distribution of the reciprocal variate) are known. To illustrate use of the theorem we consider the Normal and a shifted Gamma distribution:

(a) Normal distribution

In this case

$$f_X(x) = (2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}x^2} \quad , \tag{5.2.11}$$

and

$$f_Y(y) = (2\pi)^{-\frac{1}{2}} y^{-2} e^{-1/[2y^2]} \tag{5.2.12}$$

giving modes at $m_X = 0$ and $m_Y = \pm 1/\sqrt{2}$ respectively. Hence,

$$u_+ = f_X^{\frac{1}{2}}(0) = (2\pi)^{-\frac{1}{4}} \quad , \tag{5.2.13}$$

$$v_+ = f_Y^{\frac{1}{2}}(1/\sqrt{2}) = (2\pi)^{-\frac{1}{4}}(2/e)^{\frac{1}{2}} \quad , \tag{5.2.14}$$

$$v_- = f_Y^{\frac{1}{2}}(-1/\sqrt{2}) = (2\pi)^{-\frac{1}{4}}(2/e)^{\frac{1}{2}} \quad . \tag{5.2.15}$$

Figure 5.1 shows plots of the regions,

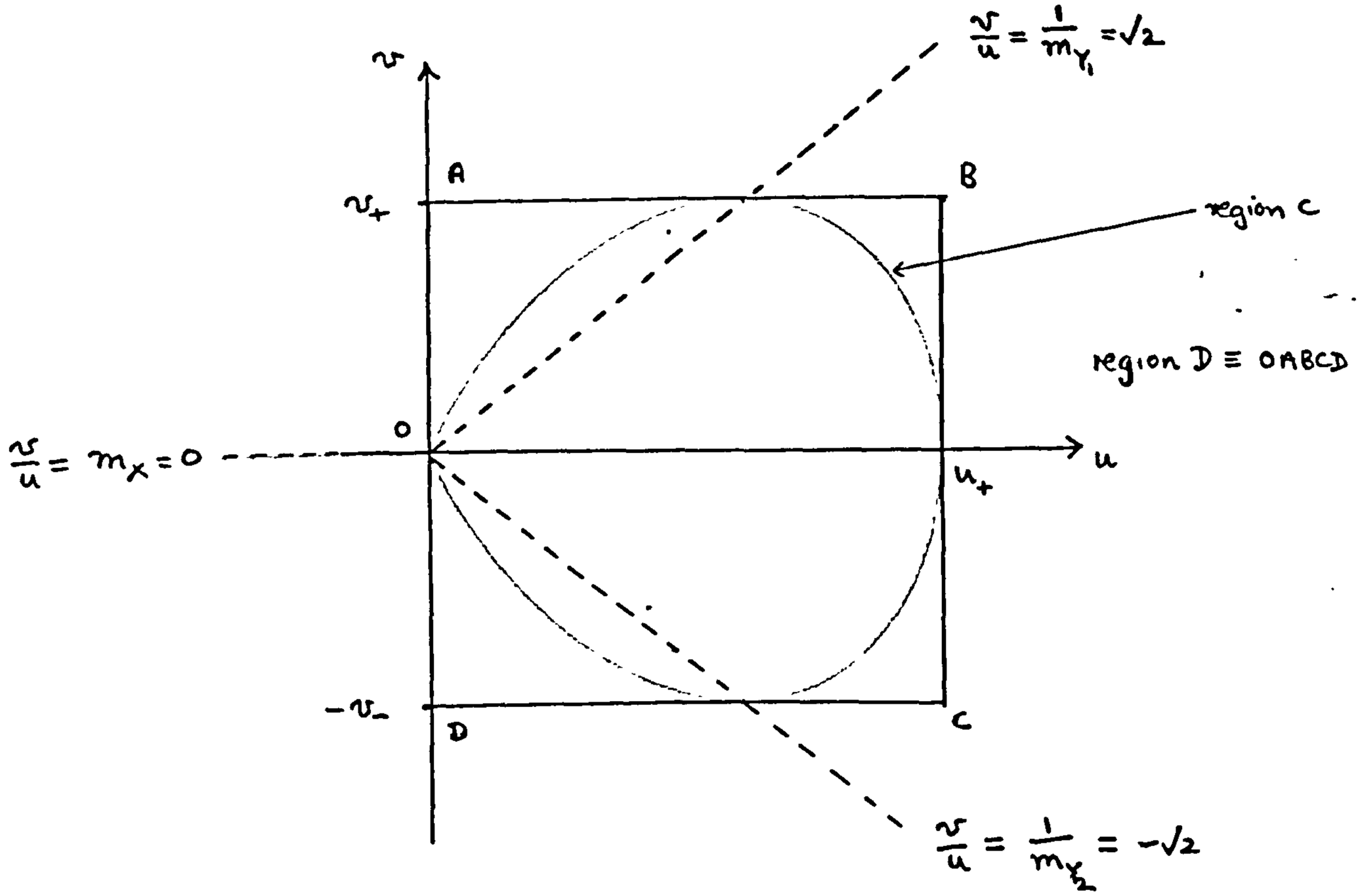
$$C = \{(u,v) : 0 \le u \le (2\pi)^{-\frac{1}{4}} e^{-v^2/4u^2} \; ; \; \left|\frac{v}{u}\right| \le \infty\}$$

$$= \{(u,v) : 0 \le u \le (2\pi)^{-\frac{1}{4}}; \; |v| \le u\{-\ell n(2\pi u^4)\}^{\frac{1}{2}}; \; \left|\frac{v}{u}\right| \le \infty\}, \tag{5.2.16}$$

and

$$D = \{(u,v) : 0 \le u \le u_+ \; ; \; -v_- \le v \le v_+ \; ; \; \left|\frac{v}{u}\right| \le \infty\} \quad . \tag{5.2.17}$$

Figure 5.1 Relationship between regions C and D for the Normal Distribution



(b) A Shifted Gamma distribution ($\alpha \geq 1$)

In this case

$$f_X(x) = \frac{(x+\alpha-1)^{\alpha-1} e^{-(x+\alpha-1)}}{\Gamma(\alpha)} \qquad (x \geq 1-\alpha), \qquad (5.2.18)$$

and

$$f_Y(y) = \begin{cases} \dfrac{(y^{-1}+\alpha-1)^{\alpha-1} e^{-(y^{-1}+\alpha-1)}}{y^2\Gamma(\alpha)} & \begin{pmatrix} y \geq 0 \\ y \leq -1/[\alpha-1] \end{pmatrix} \\ 0 & \text{(elsewhere)}, \end{cases} \qquad (5.2.19)$$

giving modes at $m_X = 0$ and $m_Y = 1/\{1 \pm \sqrt{(2\alpha-1)}\}$ respectively.

Thus

$$u_+ = f_X^{\frac{1}{2}}(0) \ , \tag{5.2.20}$$

$$v_+ = f_Y^{\frac{1}{2}}(1/\{1+\sqrt{2\alpha-1}\}) \ , \tag{5.2.21}$$

$$v_- = f_Y^{\frac{1}{2}}(1/\{1-\sqrt{2\alpha-1}\}) \ . \tag{5.2.22}$$

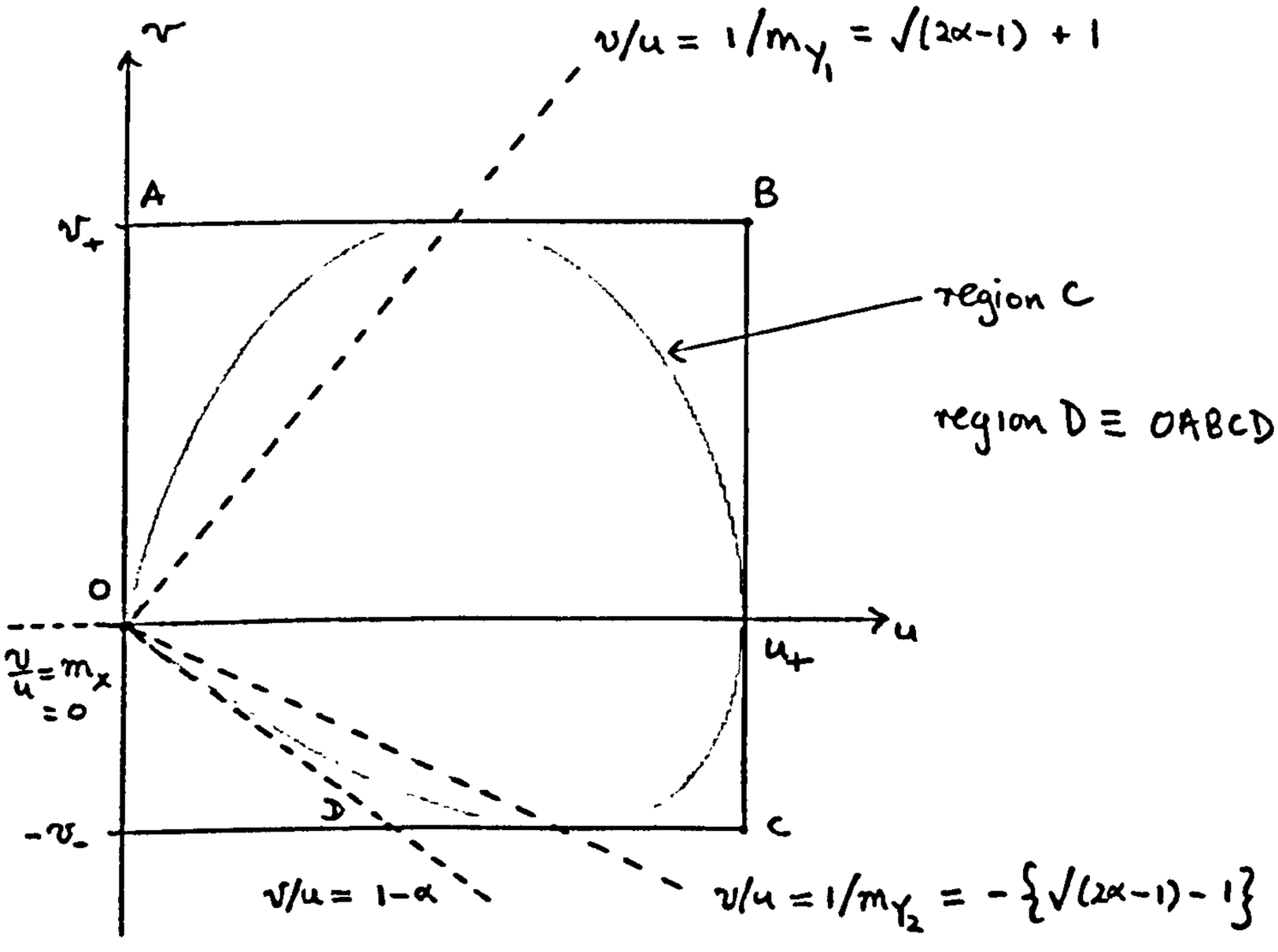
Figure 5.2 shows (for $\alpha = 3$) plots of the regions,

$$C = \left\{(u,v) : \frac{0 \le u \le \{(\frac{v}{u}) + \alpha - 1\}^{\frac{\alpha-1}{2}} \ e^{-\{\frac{v}{u} + \alpha - 1\}/2}}{\sqrt{\{\Gamma(\alpha)\}}} \ ; \ \frac{v}{u} \ge 1-\alpha\right\} \tag{5.2.23}$$

and

$$D = \{(u,v) : 0 \le u \le u_+ \ ; \ -v_- \le v \le v_+ \ ; \ \frac{v}{u} \ge 1-\alpha\} \ . \tag{5.2.24}$$

Figure 5.2 Relationship between regions C and D for a shifted Gamma distribution.



5.3 An algorithm based on the ratio of uniforms method

Applying the theorem proved in Section 5.2 to generation of standardised Wald variates, the mode of X is

$$m_X = (1 + \frac{9}{4\phi^2})^{\frac{1}{2}} - \frac{3}{2\phi} \quad , \qquad (5.3.1)$$

giving

$$u_+ = (\frac{\phi}{2\pi})^{\frac{1}{4}} e^{\phi/2} \; m_X^{-\frac{3}{4}} \; e^{-\phi(m_X+m_X^{-1})/4} \quad . \qquad (5.3.2)$$

The reciprocal variate has p.d.f.

$$f_Y(y) = (\frac{\phi}{2\pi y})^{\frac{1}{2}} \; e^{\phi} \; e^{-\phi(y+y^{-1})/2} \qquad (y \geq 0) \; , \qquad (5.3.3)$$

with a single mode at

$$m_Y = (1 + \frac{1}{4\phi^2})^{\frac{1}{2}} - \frac{1}{2\phi} \quad , \qquad (5.3.4)$$

giving

$$v_+ = (\frac{\phi}{2\pi})^{\frac{1}{4}} \; e^{\phi/2} \; m_Y^{-\frac{1}{4}} \; e^{-\phi(m_Y+m_Y^{-1})/4} \qquad (5.3.5)$$

and

$$v_- = 0.$$

Thus initial sampling is from the region D (5.2.2), confined to the first quadrant with u_+, v_+ and v_- given in (5.3.2) and (5.3.5).

At this stage it is convenient to derive an expression for the sampling efficiency, E. This is the ratio of the areas of regions C and D, and so is given by

$$E = \tfrac{1}{2}\{u_+(v_+ + v_-)\}^{-1} \; ,$$

which from (5.3.2) and (5.3.5) becomes

$$E = (\frac{\pi}{2\phi})^{\frac{1}{2}} (m_X^3 m_Y)^{\frac{1}{4}} \; e^{\phi(m_X+m_X^{-1}+m_Y+m_Y^{-1}-4)/4} \quad . \qquad (5.3.6)$$

The limiting performance of the algorithm is obtained as follows.

From (5.3.1) and (5.3.4), as $\phi \to 0$

$$m_X \to \frac{\phi}{3} \qquad \text{and} \qquad m_Y \to \phi \quad , \tag{5.3.7}$$

from which we deduce that

$$E \to \left(\frac{\pi\phi}{2}\right)^{\frac{1}{2}} \frac{e}{3^{\frac{3}{4}}} \quad . \tag{5.3.8}$$

Similarly as $\phi \to \infty$,

$$m_X \to 1 \quad \text{and} \quad m_Y \to 1 \ , \tag{5.3.9}$$

from which we obtain the limiting efficiency,

$$E \to \left(\frac{\pi}{2\phi}\right)^{\frac{1}{2}} \ . \tag{5.3.10}$$

Table 5.1 shows specimen values of the sampling efficiency, together with approximate values for small and large ϕ, using (5.3.8) and (5.3.10) respectively.

Table 5.1 Sampling efficiency (E) for generation from a standardised Wald distribution with parameter ϕ, via the ratio of uniform deviates.

ϕ	100	25	5	1	0.87	0.50	0.25	0.1	0.01
E (exact)	0.125	0.244	0.495	0.719	0.721	0.692	0.594	0.429	0.148
E (asymptotic approximation for small ϕ)	-	-	-	-	-	-	0.747	0.430	0.149
E (asymptotic approximation for large ϕ)	0.125	0.251	0.560	-	-	-	-	-	-

Given u_+ and v_+, points (U,V) may be generated uniformly within D using

$$U = u_+R_1 \quad ; \quad V = v_+R_2 \ ,$$

where R_1 and R_2 are random numbers. Thus a prospective standardised variate X is given by

$$X = \frac{V}{U} = \left(\frac{R_2}{R_1}\right)\left(\frac{m_X^3}{m_Y}\right)^{\frac{1}{4}} e^{\phi(m_X+m_X^{-1}-m_Y-m_Y^{-1})/4} \ . \qquad (5.3.11)$$

The acceptance condition is

$$U \le \left(\frac{\phi}{2\pi V^3}\right) e^{2\phi}\, e^{-\phi\left(\frac{V}{U}+\frac{U}{V}\right)} \ ,$$

which in terms of R_1, R_2 and X becomes

$$R_1R_2^3 e^{\phi(X+X^{-1})} \le (m_Xm_Y)^{\frac{3}{4}} e^{\phi(3m_Y+3m_Y^{-1}+m_X+m_X^{-1})/4} \ , \qquad (5.3.12)$$

or

$$\ell n(R_1R_2^3) + \phi(X+X^{-1}) \le \tfrac{3}{4}\ell n(m_Xm_Y) + \frac{\phi(3m_Y+3m_Y^{-1}+m_X+m_X^{-1})}{4} \ . \qquad (5.3.13)$$

Thus a procedure for generating standardised first passage times is given in algorithm 5.1 below.

Algorithm 5.1

0. $m_X = (1+\frac{9}{4\phi^2})^{\frac{1}{2}} - \frac{3}{2\phi}$,

$m_Y = (1+\frac{1}{4\phi^2})^{\frac{1}{2}} - \frac{1}{2\phi}$,

$A = (m_X^3/m_Y)^{\frac{1}{4}} e^{\phi(m_X+m_X^{-1}-m_Y-m_Y^{-1})/4}$,

$B = \tfrac{3}{4}\ell n(m_Xm_Y) + \frac{\phi(3m_Y+3m_Y^{-1}+m_X+m_X^{-1})}{4}$.

1. generate two random numbers R_1, R_2 . $X = AR_2/R_1$.

2. If $\ell n(R_1R_2^3) + \phi(X+X^{-1}) \le B$ deliver X, else go to 1.

5.4 Computational Experience

The chi-squared method (5.1.9)-(5.1.10) was programmed as Fortran subroutine WALDCH appearing in Appendix 4 under the program name WALDCH.FOR. Chi-squared variates were generated in pairs, using a pair of Normal deviates obtained via the Box-Muller method.

Algorithm 5.1 was programmed as Fortran subroutine WALD, which appears in Appendix 4 under the program name INGAUS.FOR. In step 2 of the algorithm a logarithmic evaluation can sometimes be avoided using a pre-test based on the bound

$$\ell n\ w \le w - 1\ . \qquad (5.4.1)$$

Since $w = R_1 R_2^3 \le 1$, a tighter fitting bound is

$$\ell n\ w \le 2(w-1)/(w+1). \qquad (5.4.2)$$

Pre-tests based on (5.4.1) and (5.4.2) were incorporated into programs INLOG.FOR and INLOG1.FOR, which are variants of INGAUS.FOR, and are also listed in Appendix 4.

Table 5.2 gives mean execution times for the four routines. Timings were carried out in the manner described in section 3.4, based on the generation of 10000 variates.

Table 5.2 Mean time (μsecs) to generate one standardised Wald variate on a DEC-20 computer.

Routine \ φ	100	25	5	1	.25	.01
WALDCH (Chi-squared)	167	175	172	169	180	176
WALD (Ratio of uniforms)	937	481	239	174	205	805
WALD (pre-test 5.4.1)	917	480	221	164	214	853
WALD (pre-test 5.4.2)	998	504	233	154	229	903

Execution time for the chi-squared method appears to be independent of ϕ, and in the majority of cases is less than the other methods. This may be explained by the fact that a pair of Wald variates requires one logarithmic evaluation, one cosine evaluation, plus four random numbers. In contrast routine WALD requires a minimum of one logarithmic evaluation plus two random numbers per generated variate. The pre-tests (5.4.1)-(5.4.2) are of marginal benefit for ϕ values close to one, but for some other values appear to be of negative benefit, due to the increasing rejection rate. Our conclusion is that, despite the fact that the ratio of uniforms method may be slightly more efficient for selected values of ϕ, the chi-squared method is to be preferred as a general purpose generator, suitable for all values of ϕ.

It is nevertheless instructive to interpret the timings of the ratio of uniforms routine WALD, in the context of the nature of the acceptance region. Figures 5.3-5.8 show the acceptance region,

$$C' = \{(R_1,R_2): \ell n(R_1R_2^3)+\phi(X+X^{-1})\leq B; X=AR_2/R_1; 0\leq R_1\leq 1,\ 0\leq R_2\leq 1\}, \qquad (5.4.3)$$

for ϕ = 100, 25, 4, 1, 0.25, 0.01 respectively. For large ϕ (figures 5.3-5.4) the distribution of X is highly concentrated about the mean value 1, and hence there is a high density of points about the line $X = V/U = AR_2/R_1 = 1$, resulting in a low sampling efficiency. For moderate values of ϕ (figures 5.5-5.6) a wider variation in X values is now possible, as exemplified by the wider variation in the slopes of lines constructed by joining any point in the acceptance region to the origin. This results in higher efficiency. For small ϕ (figures 5.7-5.8), the efficiency starts to fall again. The explanation is that the distribution of X is now moderately/highly positively skewed, with relatively fewer variates taking intermediate values, and relatively more variates taking very small or very large

Figures 5.3-5.8 Plots of the acceptance region for Standardised Wald generation using a ratio of uniform deviates.

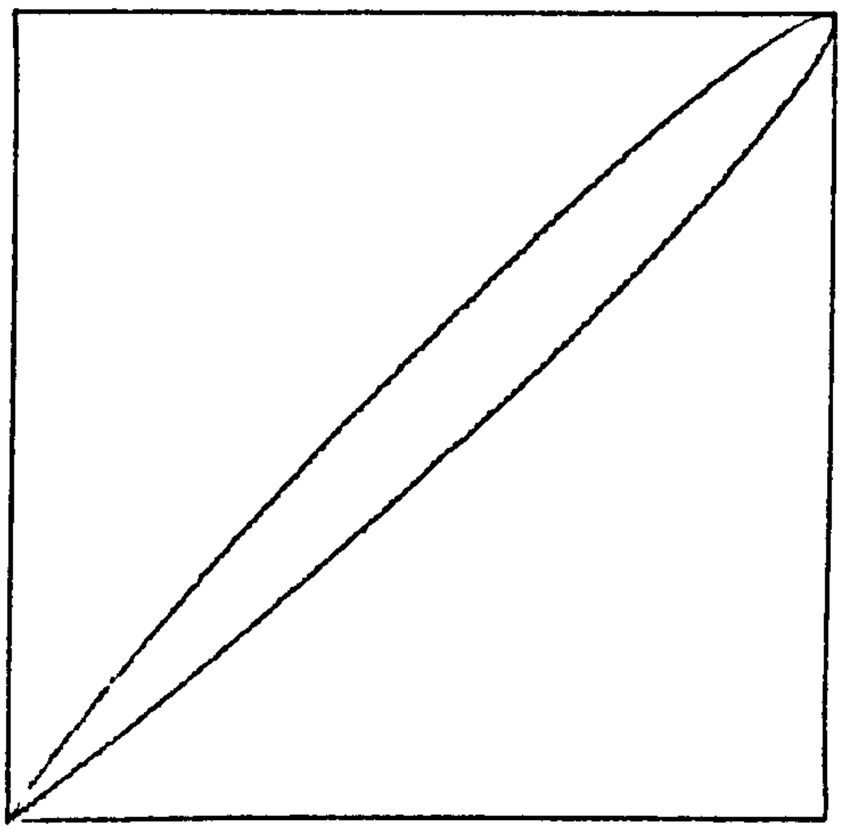


Fig 5.3 (ϕ = 100)

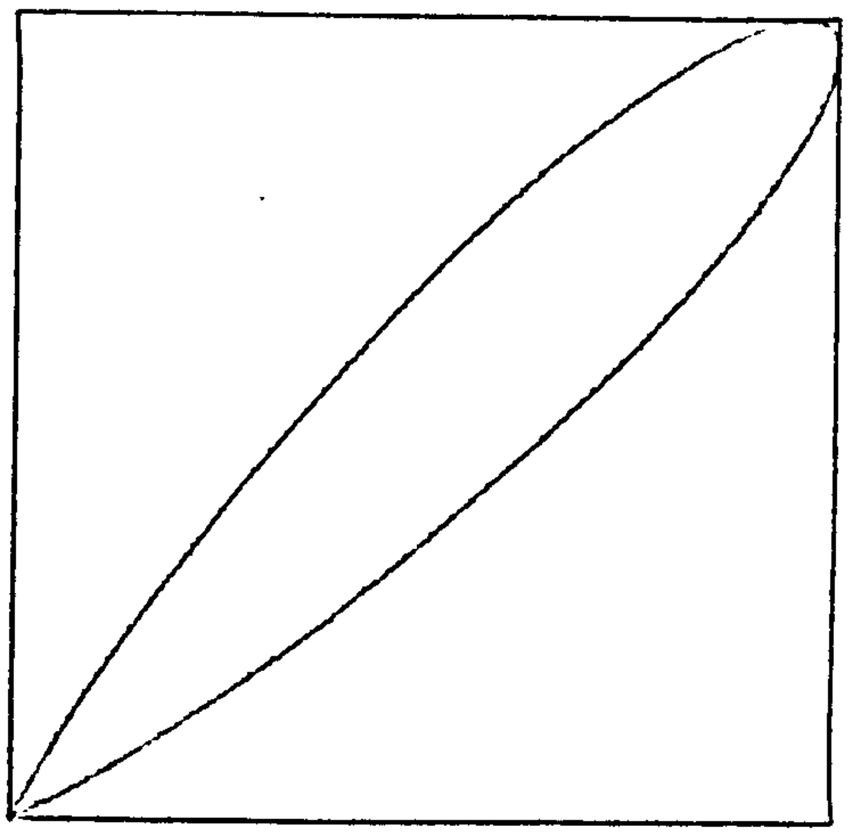


Fig 5.4 (ϕ = 25)

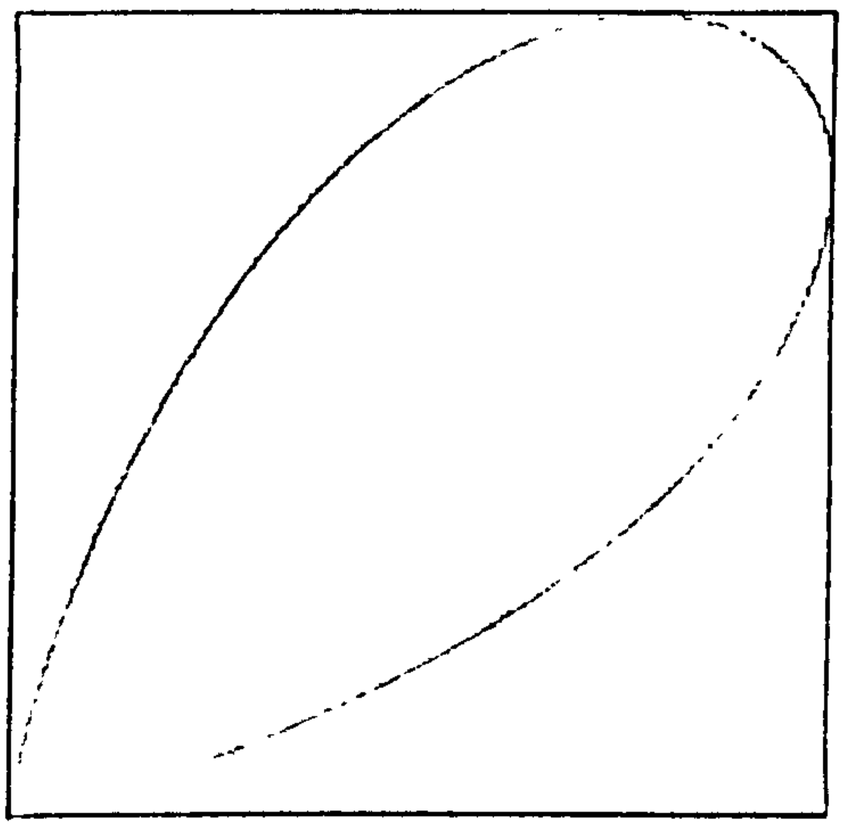


Fig 5.5 (ϕ = 4)

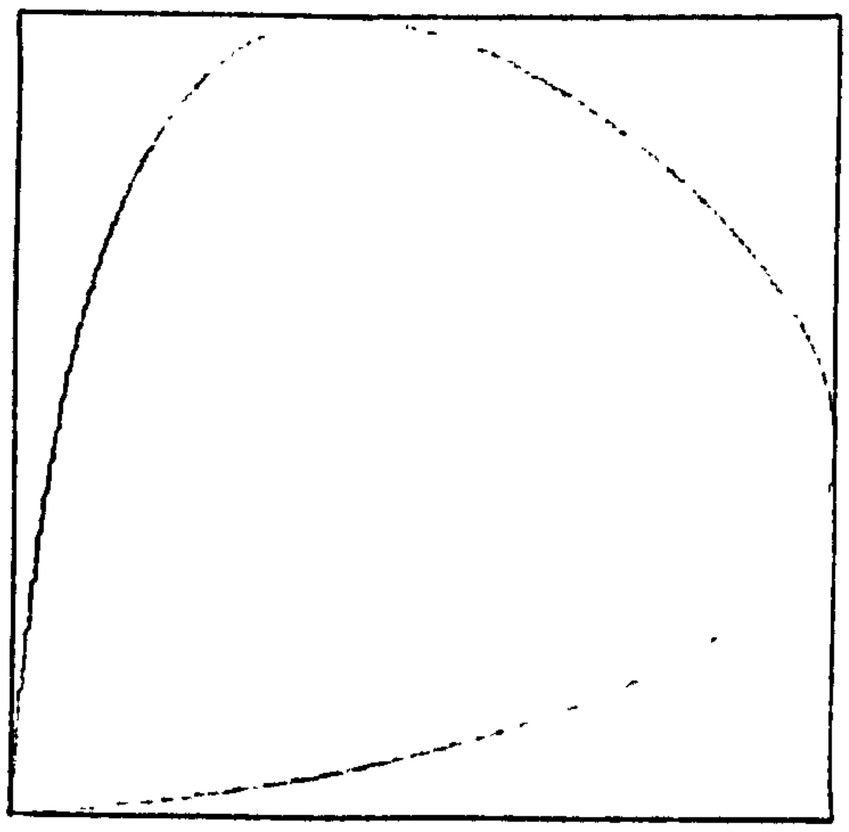


Fig 5.6 (ϕ = 1)

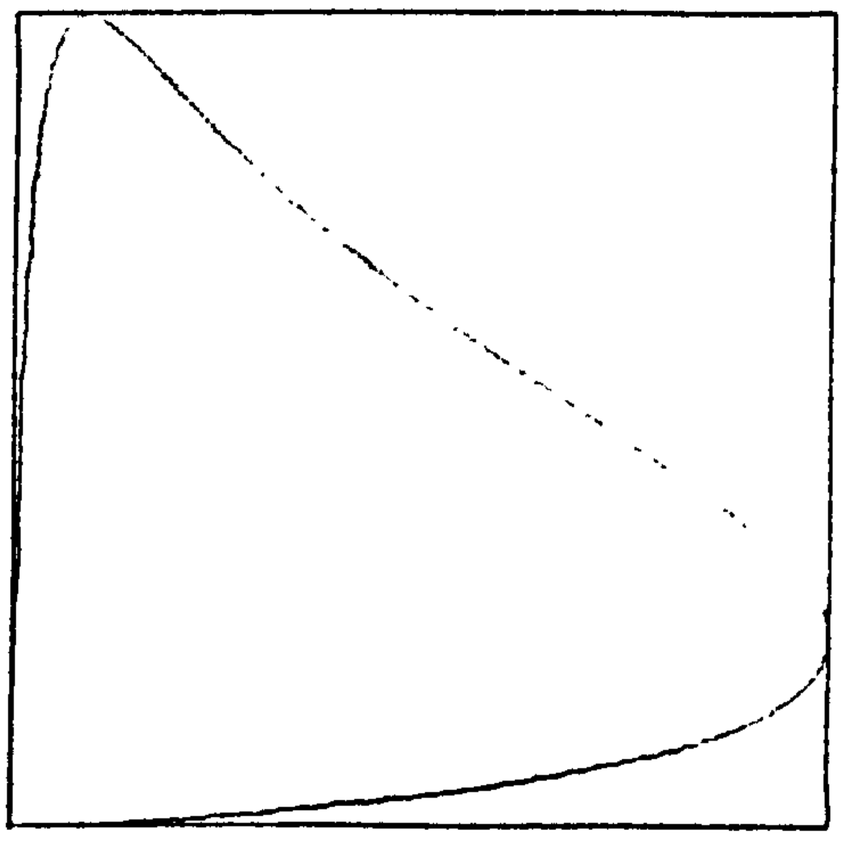


Fig 5.7 (ϕ = 0.25)

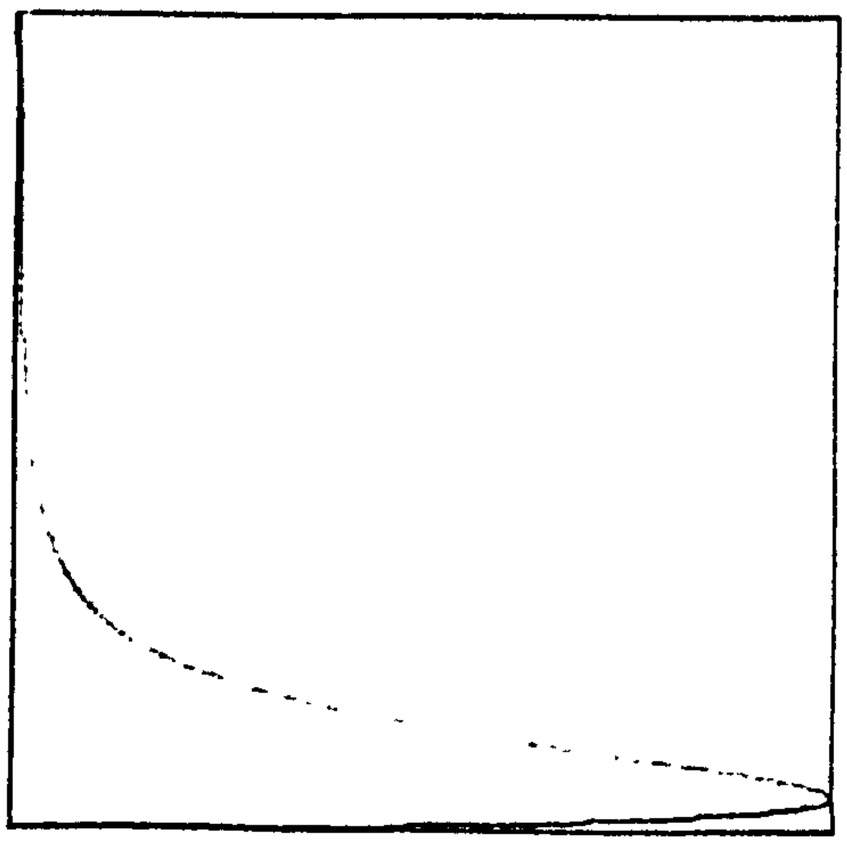


Fig 5.8 (ϕ = 0.01)

values. Thus in figure (5.8) we note that the acceptable points are beginning to concentrate in the bottom right and top left hand corners of the diagram, corresponding to low and high variate values respectively.

The performance of the ratio of uniforms method is only comparable to that of the chi-squared method when ϕ takes its most favourable values, corresponding to high sampling efficiency. In view of this, techniques to improve the sampling efficiency for other values of ϕ would probably not result in an algorithm superior to the chi-squared method. For completeness however, we remark that the efficiency could be improved for cases such as figures 5.3-5.5 by generating variates from a shifted Wald distribution with mode at zero, in a similar manner to the Kinderman and Monahan (1980) method for generating Gamma variates. Preliminary investigation in the case of the Wald distribution showed the modes of the reciprocal distribution to be given by the roots of a cubic equation, which cannot be solved analytically. In the case of a ϕ value such as that leading to figure 5.8, relocation of the mode would not help, since the mode is already close to zero.

CHAPTER 6

SOME GENERATORS FOR THE POISSON AND NEGATIVE BINOMIAL DISTRIBUTIONS

6.1 A hybrid Alias/Envelope Rejection generator for the Poisson distribution.

6.1.1 Introduction

In section 3.2 several Poisson generators were described and reference made to the timing experiments carried out by Atkinson. These indicate that of the "exact" algorithms, the Alias and Indexed search procedures give the fastest per-variate execution speeds, with little to choose between the two. For the Alias method, the timings in table 1 of Atkinson (1979c), and the fact that variates are generated by direct access of a particular element in an array, suggest that the speed is virtually independent of the mean μ. In the case of the Indexed Search procedure, providing the number of intervals is sufficiently large, the speed will again be almost independent of μ. This is confirmed by the results in table 1 of Atkinson (1979c), where for 100 intervals, the time is fairly insensitive to μ (up to 20), but rises gradually as μ increases to 200.

A remaining unsatisfactory feature of the Alias procedure (for any distribution having an infinite domain) is the arbitrary truncation of the distribution. This criticism also extends to the Indexed Search method, unless one is prepared to calculate individual tail probabilities when they are (infrequently) required.

The first aim therefore of this section is to incorporate the positive features of the Alias method (principally speed and reliance on a standard Alias table generator) into a hybrid algorithm, which generates exact variates without truncation.

A secondary aim concerns the robustness of the algorithm for large μ. While Atkinson quotes timings for μ up to 200, it was felt that it would be useful to develop a generator handling values of μ up to at least 1000. This raises numerical problems regarding the accurate and efficient evaluation of the probability masses.

These two features could also be incorporated into the Standard Indexed Search Method. For 100 intervals, Atkinson's timings suggest that there is little to choose between the two methods. It was felt therefore that little extra could be gained by applying these modifications to the Indexed Search Method.

6.1.2 A hybrid algorithm

The problem of generation from

$$f_X(x) = \frac{\mu^x e^{-\mu}}{x!} \qquad (x = 0,1,\ldots) , \qquad (6.1.1)$$

having infinite domain, is resolved by representing it as a probability mixture,

$$f_X(x) = wf_A(x) + (1-w)f_T(x) , \qquad (6.1.2)$$

where

$$f_A(x) = \frac{\mu^x e^{-\mu}}{wx!} \qquad (x = 0,1,\ldots,m-1), \qquad (6.1.3)$$

$$f_T(x) = \frac{\mu^x e^{-\mu}}{(1-w)x!} \qquad (x = m,m+1,\ldots), \qquad (6.1.4)$$

$$w = \sum_{x=0}^{m-1} \frac{\mu^x e^{-\mu}}{x!} , \qquad (6.1.5)$$

and m is any positive integer. The method is then to sample A with probability w, using the Alias method, and to sample T with probability (1-w), using envelope rejection with a truncated Geometric distribution, having probability mass function,

$$g_Z(x) = (1-[\mu/m])(\mu/m)^{x-m} \qquad (m > \mu;\ x = m, m+1, \ldots) \ . \tag{6.1.6}$$

A prospective tail variate is conveniently obtained by setting

$$Z = m + \langle E/\ell n(m/\mu) \rangle \ , \tag{6.1.7}$$

where E is a standard Negative Exponential Variate. To obtain the acceptance condition, we note that

$$f_T(x)/g_Z(x) = \mu^x e^{-\mu}/\{(1-w)x!\,(1-[\mu/m])(\mu/m)^{x-m}\} \tag{6.1.8}$$

$$(x = m, m+1, \ldots) \ ,$$

is maximised at x = m, giving

$$M = f_T(m)/g_Z(m) = \frac{\mu^m e^{-\mu}}{(1-w)m!\{1-[\mu/m]\}} \ . \tag{6.1.9}$$

Thus, give a random number R_2, the acceptance condition

$$R_2 M \le f_T(Z)/g_Z(Z) \tag{6.1.10}$$

becomes

$$R_2 \le m!\ m^{Z-m}/Z! \ . \tag{6.1.11}$$

m is chosen so that the proportion of variates sampled from $f_T(x)$ is small. In this way the efficiency (M^{-1}) of sampling from $f_T(x)$ is not crucial to the efficiency of the whole routine. Clearly m should not be too large, otherwise an extra overhead arises from the generation of additional alias and cut-off values. In practice a value of $m = 1 + \langle \mu + 2.5\sqrt{\mu} \rangle$ proved to be a satisfactory compromise. The Hybrid algorithm becomes:

Algorithm 6.1

1. $m = 1 + \langle \mu + 2.5\mu^{\frac{1}{2}} \rangle$.

2. $f_X(x) = \mu^x e^{-\mu}/x!$ $\qquad (x = 0,1,\ldots,m-1)$.

3. $w = \sum_{x=0}^{m-1} f_X(x)$, $\quad f_A(x) = f_X(x)/w$ $\qquad (x = 0,1,\ldots,m-1)$.

4. Call Alias subroutine to compute cut-off and alias values, c_j and a_j $\quad (j = 0,1,\ldots,m-1)$.

5. generate $R_1 \sim U(0,1)$.

6. If $R_1 > w$ go to 10.

7. $R_1 = R_1/w$.

8. $A = R_1 m$.

9. If $A - \langle A \rangle \leq C_{\langle A \rangle}$ deliver $\langle A \rangle$, else deliver $a_{\langle A \rangle}$.

10. generate $R_2 \sim U(0,1)$, E a standard negative Exponential variate.

11. $Z = m + \langle E/\ell n(m/\mu) \rangle$.

12. If $R_2 > m!\, m^{Z-m}/Z!$ go to 10, else deliver Z.

Steps 1-4 set up the probability distribution which is input into the Alias routine, and compute the cut-off and alias values. These are saved between calls. Steps 5-7 determine whether to sample A or Z. In the former case a random number is re-constituted by dividing by w, which poses no problem as w is close to 1. Step 8 produces a continuous variate uniformly distributed in (0,m). Noting that $A - \langle A \rangle \sim U(0,1)$, step 9 determines whether the primary value $\langle A \rangle$, or Alias value $a_{\langle A \rangle}$, is to be delivered. If sampling is from the tail, step 11 gives a prospective variate, which is either accepted or rejected in step 12.

To the best of the author's knowledge the Geometric tail procedure has not been used elsewhere. The work was first described in Dagpunar (1979) and at that time the author was also under the impression that it represented the first application of the Alias method to the Poisson distribution. Independently Atkinson (1979c) reported results for an Alias method with truncation of the distribution. However it is felt that the method described above represents an improvement, in that tail values are not lost.

6.1.3 Programming and Numerical Aspects

If the algorithm is to be robust over a wide range of μ, then care is needed in programming certain steps. In step 2 $e^{-\mu}$ and $\mu^x/x!$ can easily under and overflow respectively, causing great inaccuracies in $f_X(x)$. Evaluation using logarithms would make the set-up time unduly high, particularly for large μ. The approach used is to compute accurately the modal probability $f_X(<\mu>)$, then to compute $f_X(x)$ $(x \neq <\mu>)$ recursively, starting at the mode. In this way no premature underflow will occur, since the probabilities are decreasing either side of the mode. If a probability genuinely approaches an underflow value, then subsequent probabilities further from the mode are set to zero.

The problem of computing $f_X(<\mu>)$ accurately was tackled in the following manner. This modal probability may be rewritten as

$$\begin{aligned} f_X(<\mu>) &= e^{-\mu}\mu^{<\mu>}/<\mu>! \\ &= e^{-(\mu-<\mu>)}(\mu/e)^{<\mu>}/<\mu>! \\ &= e^{-(\mu-<\mu>)}\prod_{j=0}^{<\mu>} B(j)\ , \end{aligned} \qquad (6.1.12)$$

where

$$B(j) = \begin{cases} 1 & (j = 0) \\ \mu/(ej) & (j \geq 1). \end{cases} \qquad (6.1.13)$$

We note that with the possible exception of $j = 0$, $B(j)$ is a decreasing function of j. Direct evaluation of the product of factors $B(j)$ may lead to premature overflow when μ is large, since, for small j, $B(j)$ is very much larger than 1, while for large j, it is smaller than 1.

A subroutine, REMULT, due to Dr N H Scott, Plymouth Polytechnic, evaluates the product by starting at $j^* = < \mu/e >$ (factor value closest to 1), then moves to $j^*-1,\ldots,j'$ until the cumulative product exceeds $B(j^*)$. When this happens factors corresponding to $j^*+1,\ldots,j''$ are brought in until the product falls below $B(j^*)$. At this point factors to the left of j' are brought in. This process continues, so that, initially at least, the cumulative product never deviates far from $B(j^*)$. By containing the magnitude of the cumulative product in this way, premature under or overflow is avoided.

In step 12, Z is delivered immediately if $Z = m$. Otherwise the right hand side of the inequality is evaluated as

$$\left(\frac{m}{Z}\right)\left(\frac{m}{Z-1}\right) \cdots \left(\frac{m}{m+1}\right).$$

Since these factors are all less than 1, no premature underflow occurs.

Algorithm 6.1 was programmed with these features using the Standard Alias table generating subroutine PTAB, due to Kronmal and Peterson (1978). A Fortran subroutine POISAL appears in Appendix 5 under the program name ALIAS.FOR .

6.1.4 Verification and Validation

Because of the relative complexity of the algorithm, and the numerical problems associated with large μ, the following steps were taken to verify and validate the routine:

(i) The product of factors B(j), as calculated by the routine REMULT was compared with known values using tables of $\log_{10}\mu!$. The results shown in table 6.1, indicate that the routine is correct to at least 6 significant figures, and is therefore adequate for our purposes. Also shown in table 6.1 are the bounds on $(\mu/e)^{<\mu>}/<\mu>!$, using:

$$\sqrt{(2\pi)}\ e^{-<\mu>}<\mu>^{<\mu>+\frac{1}{2}} \leq <\mu>! \leq \sqrt{(2\pi)}\ e^{-<\mu>}<\mu>^{<\mu>+\frac{1}{2}}\ e^{1/(12<\mu>)}.$$

When μ is 10 or larger, the lower bound so obtained is correct to at least 6 significant figures. It is of interest to note therefore, that for these values of μ, this bound could have been substituted for the routine REMULT, thereby reducing the set-up time. Although this is a useful practical feature, it was not programmed into the final version of POISAL, principally because it was desired to keep the routine exact in a mathematical sense (providing there was no significant numerical degradation) but also because the routine REMULT (or simplified version of it) would still be required for smaller values of μ.

(ii) A test of the adequacy of the random number generator. Random numbers were obtained using the Fortran function RAN(•) available on the DEC-20. A wide variety of empirical tests is available for pseudo random number generators. Since our main concern is with the uniformityand apparent statistical independence of the numbers, two tests were applied. These tests are described together with the results in Appendix 2.

The conclusions reached are that there is little evidence to suggest departure from uniformity, either locally or globally, and similarly that there appears to be little autocorrelation for lags up to 250.

(iii) Comparison of the sample mean over 10,000 generated variates with the population mean. Sample means are shown in table 6.1 . Agreement is good, and even with such large sample sizes, there is no evidence to suggest that observations are not being drawn from a parent population of mean μ.

(iv) Comparison of the population probability mass function with corresponding sample probabilities over $n = 10,000$ generated variates. A test of fit of the simple hypothesis that the variates are drawn from a Poisson distribution with known mean was carried out, by dividing the variate range into k mutually exclusive and exhaustive classes. The central $k-2$ classes each correspond to one variate value, while the remaining two were formed by grouping tail values. Under the null hypothesis, the statistic $\sum_{i=1}^{k} n(\hat{p}_i - p_i)^2/p_i$ is asymptotically distributed as chi-squared with $k-1$ degrees of freedom. Values of this statistic, together with degrees of freedom are shown in table 6.1 . These show that there is no evidence to reject the null hypothesis at the 5% level. These results were obtained using a program VERIFY.FOR . This together with output listings appears in the supporting material.

TABLE 6.1 Summary of results of Verification/Validation experiments on routine POISAL

μ	1	10	20	50	100	200	1000
Product of factors as calculated using REMULT.	0.3678794	0.1251100	0.0888353	0.0563250	0.0398610	0.0281977	0.0126146
Known value of above.	0.3678794	0.1251100	0.0888353	0.0563250	0.0398610	0.0281977	0.0126146
Bounds on above.	0.3670446 0.3989423	0.1251097 0.1261563	0.0888353 0.0892062	0.0563250 0.0564190	0.0398610 0.0398942	0.0281977 0.0282095	0.0126146 0.0126157
Sample Mean (10,000 variates).	0.9997	10.0355	19.9898	49.9645	99.8464	200.0215	999.8674
Value of chi-squared statistic (degrees of freedom).	3.83(6)	11.05(21)	11.08(28)	32.91(43)	48.26(59)	80.03(78)	163.91(156)

6.2 Modification to an envelope rejection procedure for the Poisson distribution.

6.2.1 Introduction

In section 3.2 reference was made to the envelope rejection procedure (PA), due to Atkinson. In this section the following improvements are made:

(i) Incorporation of a Logistic p.d.f., truncated at $x = -0.5$, to eliminate generation of prospective variates having value less than -0.5.

(ii) Elimination in the rejection step, of the direct evaluation of the Logistic density, which involves an exponentiation.

(iii) Elimination of storage of logarithms of factorials, thereby making the algorithm compact and portable.

(iv) The algorithm is made robust with respect to large μ.

6.2.2 Development of Algorithm

Poisson(μ) variates, N, may be obtained from continuous variates X, having p.d.f.

$$f_X(x) = \frac{\mu^{<x+0.5>} e^{-\mu}}{<x+0.5>!} \qquad (x > -0.5) , \qquad (6.2.1)$$

by setting $N = <X+0.5>$. Given (6.2.1), a suitable target distribution is Logistic, truncated at $x = -0.5$, and having c.d.f.

$$G_Y(x) = \frac{1 - e^{-\beta(x+0.5)}}{1 + e^{-(\beta x-\alpha)}} \qquad (x > -0.5) . \qquad (6.2.2)$$

We follow Atkinson in choosing $\beta = \pi/\sqrt{(3\mu)}$ and $\alpha = \beta\mu$, thereby equating mean and variance of the Poisson and <u>untruncated</u> Logistic. Given a random number R_1, inversion yields,

$$R_1 = \frac{1 - e^{-\beta(X+0.5)}}{1 + e^{-(\beta X-\alpha)}} \quad ,$$

or

$$X = -0.5 + \beta^{-1} \ell n \left\{ \frac{1 + R_1 e^{\alpha+0.5\beta}}{1 - R_1} \right\} \quad . \qquad (6.2.3)$$

Given a random number R_2, the acceptance condition is

$$R_2 M g_Y(X) \leq \frac{\mu^{<X+0.5>} e^{-\mu}}{<X+0.5>!} \quad , \qquad (6.2.4)$$

where

$$M \geq \text{Max}[\, f_X(x)/g_Y(x)\,] \quad .$$

Since,

$$g_Y(X) = \frac{dG}{dR_1} \cdot \frac{dR_1}{dX}$$

$$= \frac{dR_1}{dX}$$

$$= \frac{\beta(1-R_1)(1+R_1 e^{\alpha+0.5\beta})}{(1 + e^{\alpha+0.5\beta})} \quad , \qquad (6.2.5)$$

(6.2.4) becomes

$$\frac{\beta R_2 (1-R_1)(1+R_1 e^{\alpha+0.5\beta}) M}{1 + e^{\alpha+0.5\beta}} \leq \frac{\mu^N e^{-\mu}}{N!} \quad . \qquad (6.2.6)$$

Since $N!$ can easily overflow, it is necessary to take logarithms giving,

$$\ell n\{R_2(1-R_1)(1+R_1 e^{\alpha+0.5\beta})\} + \ell n(N!/\mu^N) \leq -\mu + \ell n\{(1+e^{\alpha+0.5\beta})/M\beta\}. \qquad (6.2.7)$$

We note that direct evaluation of the Logistic density has been avoided by expressing $g_Y(X)$ in terms of the random number R_1. This saves an exponentiation and is to be compared with step 4 of Atkinson's algorithm. In step 2 of Atkinson's algorithm, prospective variates

smaller than -0.5 have to be discarded, whereas (6.2.3) never gives such values. Since the c.d.f. of the untruncated Logistic is

$$H(x) = [1 + e^{-(\beta x-\alpha)}]^{-1} \quad , \tag{6.2.8}$$

the proportion of variates rejected in this way is

$$P = (1 + e^{\alpha+0.5\beta})^{-1} \quad , \tag{6.2.9}$$

and specimen values are shown in table 6.2 .

Table 6.2 Probability that prospective variate is smaller than -0.5 in Atkinson's method.

μ	0.5	1	2	5	10	50
P	0.0714	0.062	0.039	0.011	0.002	0.000024

We conclude that the proportion of variates saved by this modification is small (even for small μ). However the modification is retained since the program statements involved are not significantly more complex than in the original method.

A major problem of the acceptance condition (6.2.7) is the evaluation of $S_N = \ell n(N!/\mu^N)$. One possibility which is suggested in the original version is to provide a table of log factorials. This immediately puts the algorithm into the non-compact class. Additionally for large N, the log factorial would have to be calculated recursively, using the last entry in the table provided. A second possibility is to use the recursion $S_N = \ell n(N/\mu) + S_{N-1}$. For all but small N, the

number of logarithmic evaluations involved will slow down the procedure beyond the point of usefulness.

The method finally chosen, which retains compactness and speed is to use some effective pre-tests, based on the Stirling bounds,

$$0.5\ell n(2\pi)+(N+0.5)\ell n\ N-N \le \ell n\ N! \le 0.5\ell n(2\pi)+(N+0.5)\ell n\ N-N+\{1/(12N)\}. \tag{6.2.10}$$

Using these bounds in (6.2.7), we find that sufficient conditions for a prospective variate N to be accepted and rejected are

$$\ell n\{R_2(1-R_1)(1+R_1e^{\alpha+0.5\beta})\} + 0.5\ell n(2\pi\mu) + (N+0.5)\ell n(N/\mu)$$

$$- N + (12N)^{-1} \le -\mu + \ell n\{(1+e^{\alpha+0.5\beta})/M\beta\}\ , \tag{6.2.11}$$

and

$$\ell n\{R_2(1-R_1)(1+R_1e^{\alpha+0.5\beta})\} + 0.5\ell n(2\pi\mu) + (N+0.5)\ell n(N/\mu)$$

$$- N > -\mu + \ell n\{(1+e^{\alpha+0.5\beta})/M\beta\}\ , \tag{6.2.12}$$

respectively. Due to the known tightness of the bounds (6.2.10) (even for small N), and the variability in

$$\ell n\{R_2(1-R_1)(1+R_1e^{\alpha+0.5\beta})\}$$

as R_2 varies, we expect that there is only a small probability that <u>neither</u> condition (6.2.11) nor (6.2.12) is satisfied. Thus only on a small proportion of occasions will S_N have to be evaluated (expensively) via $S_N = \ell n(N/\mu) + S_{N-1}$.

From the discussion above, a compact algorithm for Poisson generation appears below.

Algorithm 6.2

0. $\beta = \pi/\sqrt{(3\mu)}$, $\alpha = \beta\mu$, $\gamma = e^{\alpha+0.5\beta}$,

$\Delta = -\mu + \ell n\{(1+e^{\alpha+0.5\beta})/M\beta\} - 0.5\ell n(2\pi\mu)$.

1. generate $R_1, R_2 \sim U(0,1)$.

2. $N = < \beta^{-1} \ell n\{(1+\gamma R_1)/(1-R_1)\} >$, $W = \ell n\{R_2(1-R_1)(1+\gamma R_1)\}$,

$V = 0$.

3. If $N = 0$ go to 6.

4. $Z = W + (N+0.5)\ell n(N/\mu) - N$.

If $Z + (12N)^{-1} \leq \Delta$ deliver N, else, if $Z > \Delta$ go to 1.

5. $V = \ell n(N!/\mu^N)$.

6. If $W + V \leq \Delta + 0.5\ell n(2\pi\mu)$ deliver N, else go to 1.

6.2.3 Determination of Sampling Efficiency, M^{-1}.

An explicit expression for M could not be found. M was estimated using the following Monte-Carlo procedure. Referring to (6.2.6) it is clear that the maximum value (M) of $f_X(x)/g_Y(x)$ is obtained by setting

$$M = e^{-(\mu+T)}(1 + e^{\alpha+0.5\beta})/\beta \quad , \tag{6.2.13}$$

where

$$T = \min_{R_1}[\ell n\{(1-R_1)(1+R_1 e^{\alpha+0.5\beta})\} + \ell n(N!/\mu^N)] \, . \tag{6.2.14}$$

An estimate of T is obtained by generating a lengthy realisation of random numbers R_1, and identifying the trial which gives the minimum value of the expression under the minimisation sign in (6.2.14). By using the Stirling lower bound on $N!$, we obtain an upper bound (M_0) on the largest value of $f_X(x)/g_Y(x)$ achieved over the realisation. For each value of μ, M was estimated using 10,000 random numbers. If required, further accuracy could have been obtained by estimating the minimising value of R_1, then generating a second realisation with R_1 conditioned to fall within a small interval surrounding the initial estimate. However this was not thought necessary in view of the number of random numbers used, and the margin provided by using a Stirling bound rather than the exact value.

The relevant Fortran program appears in Appendix 5 under the program name MVALUE.FOR . Estimated values of M are summarised in table 6.3 below.

Table 6.3 Estimated Values of M for envelope rejection method

μ	0.5	1.0	2.0	5.0	10.0	20.0	30.0	50.0	100	200	1000
M	17.48	4.854	2.914	2.049	1.700	1.546	1.484	1.421	1.360	1.323	1.276

In implementing algorithm 6.2, the value of M used is the one shown in table 6.3 . For values of μ not shown in the table, the value used is the one corresponding to the next smallest μ shown in the table. This incurs a small loss in efficiency.

6.2.4 Verification, Validation and Analysis of Algorithm

Algorithm 6.2 was programmed and a Fortran function NREJ appears in Appendix 5, under the program name POLOG.FOR . To be consistent with the procedure adopted in 6.1, the following steps were taken to verify and validate the function:

(i) Comparison of population and sample mean over 10,000 generated variates.

(ii) Comparison of the sample mean number of prospective variates required to generate 1 Poisson variate (sample M-value) with the M-value used in the program. This sample mean is based on the generation of 10,000 Poisson variates.

(iii) Comparison of the population probability mass function with corresponding sample probabilities over 10,000 generated variates.

Results are summarised in table 6.4 . The sample M-values and means are in good agreement with the population values, while the chi-squared statistic values, show that at the 5% level, there is no evidence to suppose that observations are not being drawn from the required Poisson distribution. These results were obtained using a program VERLOG.FOR . This together with output listings appears in the supporting material.

A critical feature of the generator is the effectiveness or otherwise of the pre-tests. Using the generation of 10,000 variates, the program VERLOG.FOR also records the number of times $\ell n(N!/\mu^N)$ has to be evaluated directly. The estimated mean rate (per generated variate) of direct log factorial evaluations is shown in table 6.4 . For large μ, where a log factorial evaluation is extremely expensive, the rate of such evaluations is very small. For small μ, the rate is still low (lower than 5%), while individual log factorial evaluations are less expensive. We conclude therefore that the pre-tests are very effective.

Table 6.4 Summary of results of Verification/Validation experiments on function NREJ, together with analysis of use of direct log.factorial evaluations.

μ	1	10	20	50	100	200	1000
Sample mean (10,000 variates)	1.0068	10.0415	20.0622	50.0761	100.1116	200.0917	1000.0327
M-value used	4.854358	1.699009	1.545193	1.420260	1.359644	1.322996	1.275129
Sample M-value (10,000 variates)	4.955	1.692	1.535	1.409	1.350	1.313	1.270
Value of chi-squared statistic. (Degrees of freedom)	4.39(6)	23.78(21)	27.00(28)	38.84(43)	62.21(59)	92.10(78)	156.61(156)
Sample rate of log. factorial evaluations. (10,000 variates)	0.0411	0.0105	0.0047	0.0023	0.0010	0.0005	0.0002

6.3 Timing Comparisons for the Poisson Generators

The two generators described previously are intended to be complementary in performance. Algorithm 6.1 requires extensive set-up and storage, but should result in low per-variate execution times, performance being dominated by the Alias part of the generator. Algorithm 6.2 requires very little set-up, is easy to implement, but is expected to result in higher per-variate execution times, because of dependence upon three logarithmic evaluations plus a rejection step.

In carrying out the timing experiments two variants of Algorithm 6.2 were used with and without the Stirling pre-tests. Tests were carried out both for μ fixed and varying between calls, and the results are summarised in tables 6.5 and 6.6 respectively. The experiments were carried out under the conditions described in section 3.4 .

When μ remains fixed between calls, routine POISAL is uniformily fastest, taking approximately 55 μsecs independently of μ. The incorporation of Stirling pre-tests is clearly justified in NREJ, the ratio of times without and with pre-tests, rising from approximately 3 at $\mu = 10$ to approximately 200 at $\mu = 1000$. Clearly use of NREJ without the pre-test is not recommended, although it is of interest to note that for such a version, the lowest execution time is at $\mu \simeq 2$, reflecting a compromise between improving sample efficiency and expensive log factorial evaluations.

Table 6.5 Mean time (μsecs) to generate 1 Poisson variate on a DEC-20, when μ remains fixed between calls.

μ	0.5	1	2	5	10	20	50	100	200	1000
POISAL	53	54	53	55	57	55	53	57	56	56
NREJ (with pre-test)	3570	999	643	482	407	365	344	323	324	316
NREJ (without pre-test)	4140	1375	910	1010	1350	2140	4580	8280	15590	70900

Table 6.6 Mean time (μsecs) to generate 1 Poisson variate on a DEC-20, when μ re-specified between calls.

μ	0.5	1	2	5	10	20	50	100	200	1000
POISAL	389	620	750	1080	1540	2500	5250	9160	16990	78500
NREJ (with pre-test)	3830	1201	865	692	623	586	555	548	532	531
NREJ (without pre-test)	4480	1340	1090	1230	1590	2380	4760	8470	15680	75100

When μ is re-specified between calls, POISAL becomes very much less competitive, the set-up time being equivalent to the generation of approximately 12 variates at $\mu = 1$, 100 at $\mu = 50$ and 1500 at $\mu = 1000$. This set-up time is approximately linear in $m = < 1 + \mu + 2.5\sqrt{\mu} >$. The set-up time is not so crucial in NREJ (with pre-test), accounting for approximately 200 μsecs of generation time. From an execution time viewpoint, the optimal point at which to switch from NREJ to POISAL appears to be $\mu \lesssim 2$.

The conclusion is that if ease of implementation and low storage requirements are not crucial factors, and if a sufficient number of variates is to be generated, then POISAL should be used. In other cases the function NREJ (with pre-test) should be used. In the latter case, the method degrades for $\mu \lesssim 2$, and for these values the simple multiplicative method should match the characteristics of NREJ for $\mu \gtrsim 2$. Regarding the ease of implementation and storage requirements of NREJ, it is appropriate to compare with other algorithms that give low execution times when μ is respecified. In this category, Atkinson (1979c) draws attention to algorithm PIF of Fishman and a composite indexed search algorithm, PQ100. Both require extensive storage of constants, while the former is inexact, in the sense that the Normal approximation is used for large μ. In contrast NREJ is exact, requires very little code and storage, and is robust for μ lying between 2 and 1000.

6.4 A hybrid Alias/Envelope Rejection generator for the Negative Binomial Distribution.

6.4.1 Development of Algorithm

In this section the tail generation procedure is applied to another discrete distribution, having infinite domain, the Negative Binomial, with p.m.f.

$$f_X(x) = \binom{x + k - 1}{x} p^x q^k \qquad (x = 0,1,\ldots) \ , \tag{6.4.1}$$

where $q = 1 - p$, and k is assumed in this case to be a positive integer. As in the Poisson case, the method was first described in Dagpunar (1979).

The p.d.f. is expressed as a probability mixture,

$$f_X(x) = wf_A(x) + (1-w)f_T(x) \ , \tag{6.4.2}$$

where

$$f_A(x) = \binom{x + k - 1}{x} p^x q^k / w \qquad (x = 0,1,\ldots,m-1) \ , \tag{6.4.3}$$

$$f_T(x) = \binom{x + k - 1}{x} p^x q^k / (1-w) \quad (x = m,m+1,\ldots) \ , \tag{6.4.4}$$

$$w = \sum_{x=0}^{m-1} \binom{x + k - 1}{x} p^x q^k \ , \tag{6.4.5}$$

and m is any positive integer.

The procedure is analogous to the Poisson case, with T generated via envelope rejection, using a Geometric target distribution, having probability mass function,

$$g_Z(x) = [1 - \{p(m+k-1)/m\}]\{p(m+k-1)/m\}^{x-m} \ , \tag{6.4.6}$$

for $x = m,m+1,\ldots$ where $m > (k-1)p/q$ and $m > 1 - k$. Z is obtained by setting

$$Z = m + < E/\ell n[\, m/\{p(m+k-1)\}] > \quad , \tag{6.4.7}$$

where E is a standard Negative Exponential variate. To obtain the acceptance condition, we note that

$$f_T(x)/g_Z(x) = \frac{\binom{x+k-1}{x} p^x q^k}{(1-w)[\,1 - \{p(m+k-1)/m\}]\{p(m+k-1)/m\}^{x-m}} \tag{6.4.8}$$

is maximised at $x = m$, giving

$$M = \frac{\binom{m+k-1}{m} p^m q^k}{(1-w)\,(1-p[\,m+k-1\,]/m)} \quad . \tag{6.4.9}$$

Given a random number R_2, the acceptance condition becomes,

$$R_2 \le (Z+k-1)^{\dagger}_{Z-m} \cdot \frac{m!}{Z!} \left(\frac{m}{m+k-1}\right)^{Z-m} \quad . \tag{6.4.10}$$

As before m is set to approximately 2.5 standard deviations above the mean, giving

$$m = < (kp/q) + (2.5\sqrt{kp}/q) + 1 > \quad . \tag{6.4.11}$$

The resulting algorithm is identical to Algorithm 6.1, apart from steps 1, 2, 11 and 12, and these are given in Algorithm 6.3 below.

Algorithm 6.3

1. $m = < (kp/q) + (2.5\sqrt{kp}/q) + 1 >$.

2. $f_X(x) = \binom{x+k-1}{x} p^x q^k \qquad (x = 0,1,\ldots,m-1)$.

3.-10. As in algorithm 6.1 .

11. $Z = m + < E/\ell n[\, m/\{p(m+k-1)\}] >$.

12. If $R_2 > (Z+k-1)_{Z-m} \frac{m!}{Z!} \left(\frac{m}{m+k-1}\right)^{Z-m}$ go to 10, else deliver Z.

---

† For all real x and non-negative integers r, define $(x)_r = x(x-1) \ldots (x-r+1) \quad (r \ge 1)$, $(x)_0 = 1$.

We remark that in this form the algorithm is suitable for all real $k \geq 1$.

6.4.2 Programming and Numerical Aspects

In order to make the algorithm robust for distributions having a large mean (i.e. large kp/q), precautions are necessary to prevent premature under or overflow in calculating the probabilities $f_X(x)$. Since, for $k \geq 1$, the mode is at $s = \langle (k-1)p/q \rangle$, the modal probability may be written as

$$f_X(s) = \binom{s+k-1}{1} \ldots \binom{s+1}{k-1} p^s q^k$$

$$= \prod_{j=0}^{k-1} c(j) , \qquad (6.4.12)$$

where

$$c(j) = \begin{cases} p^{s/k} q & (j = 0) \\ \binom{s+k-j}{j} p^{s/k} q & (j \geq 1) . \end{cases} \qquad (6.4.13)$$

With the possible exception of $c(0)$, $c(j)$ is a decreasing function of j, and the factor which is closest to one (from below) is $c(j^*)$, where j^* is the smallest positive integer satisfying

$$c(j^*) < 1$$

or

$$\binom{s+k-j^*}{j^*} p^{s/k} q < 1$$

or

$$j^* > \frac{(s+k)}{(1 + p^{-s/k}/q)} . \qquad (6.4.14)$$

Thus the modal probability is evaluated using the product of factors routine REMULT, starting at

$$j^* = \text{Min}[1 + \langle (s+k)/\{1 + (p^{-s/k}/q)\} \rangle , k - 1]. \qquad (6.4.15)$$

The representation (6.4.12) seems preferable to an alternative,

$$f_X(s) = \begin{cases} q^k & (s = 0) \\ \prod_{j=1}^{s} \left\{ \left(\frac{s+k-j}{j} \right) pq^{k/s} \right\} & (s \geq 1), \end{cases} \qquad (6.4.16)$$

which for large modes involves more products than (6.4.12). If the program was required for all real $k > 0$, then (6.4.16) would have to be used rather than (6.4.12).

In step 12, Z is delivered immediately if $Z = m$. Otherwise the right hand side of the inequality is evaluated as $\prod_{j=1}^{Z-m} D(j)$, where

$$D(j) = \left(\frac{Z + k - j}{Z + 1 - j} \right) \left(\frac{m}{m + k - 1} \right) . \qquad (6.4.17)$$

Now, $D(j) \leq 1$, for all $j \in (1,\ldots,Z-m)$ providing,

$$(Z+k-j)m \leq (Z+1-j)(m+k-1) ,$$

or

$$Z \geq j + m - 1 , \qquad (6.4.18)$$

which is evidently true since $Z \geq m$. Since all the factors $D(j) \leq 1$, no premature underflow will occur by multiplying the product in its existing order. Further, since the cumulative product is non-increasing, control may be directed to step 10, if at any stage the cumulative product drops below the value of R_2.

Algorithm 6.3 was programmed with these features and a Fortran subroutine NBINOM appears in Appendix 5 under the program name NBINOM.FOR .

6.4.3 Verification and Validation

This follows closely the steps taken in 6.1.4. The results are summarised in table 6.7, and were obtained from a program VENBIN.FOR . This together with output listings appears in the supporting material.

The modal probability as calculated using REMULT is correct to at least 6 significant figures with the exception of two cases, where it is correct to 5 and 4 significant figures respectively. Comparison between sample and population means is effected by noting that the standard error on a sample mean of n observations is $\sqrt{(kp/n)}/q$. Agreement is good and there is little evidence to suggest that observations are not being drawn from a parent population of mean kp/q. The test of fit shows that in all cases there is no evidence to reject (at the 5% level) the null hypothesis that these observations are drawn from the specified Negative Binomial distribution.

Table 6.7 Summary of Results of Verification/Validation experiments on routine NBINOM.

(k,p)	(1,0.1)	(1,0.5)	(1,0.9)	(10,0.1)	(10,0.5)	(10,0.9)	(80,0.1)	(80,0.5)	(80.0.9)
Product of factors as calculated using REMULT	0.9000000	0.5000000	0.1000000	0.3486785	0.0927353	0.0138868	0.1276624	0.0316880	0.0047260
Known value of above	0.9000000	0.5000000	0.1000000	0.3486784	0.0927353	0.0138868	0.1276624	0.0316884	0.0047261
Population mean	0.1111	1.0000	9.0000	1.1111	10.0000	90.0000	8.8889	80.0000	720.0000
Sample mean (10000 variates)	0.1106	1.0044	8.8886	1.1106	10.0323	90.1610	8.9223	79.9541	719.6823
Value of chi-squared statistic (degrees of freedom)	0.66(2)	3.71(7)	22.09(47)	3.51(6)	13.21(28)	126.42(147)	21.24(21)	50.13(70)	339.78(357)

6.4.4 Timing results for routine NBINOM and conclusions.

Timing experiments were performed for parameter values k and p, fixed and respecified respectively between calls. Results are summarised in table 6.8 .

As expected, execution times in the first case are comparable to those for the routine POISAL, and are fairly insensitive to values of k and p. When parameter values are respecified between calls, the execution time increases substantially, due to the high set-up time.

We conclude that when ease of implementation and low storage requirements are not critical, and when many variates for given parameter values are required, that NBINOM has the advantage of being fast, and will outperform inversion or Bernoulli methods, whose execution times vary as kp/q and k/q respectively. For this sort of application the generation method involving Poisson and Gamma variates is unsuitable, because the parameters of the Poisson distribution vary between calls (even though k and p are fixed).

When any one of the conditions mentioned above is not satisfied, then NBINOM is unsuitable. Generation via Poisson and Gamma variates would be preferred, providing the routines have low storage requirements and are reasonably efficient for changing parameter values. A suitable choice for the Poisson would be the rejection routine NREJ. For the Gamma, generation via the envelope rejection method of Cheng (1977) and algorithm 2.1 could be used for $k \geq 1$ and $k < 1$ respectively.

Finally we remark that although NBINOM has been written for integer $k \geq 1$, it may be extended to deal with all real $k \geq 1$, by using the representation (6.4.15) in place of (6.4.12) for the modal probability.

Table 6.8 Mean time (μsecs) to generate 1 Negative Binomial Variate on a DEC-20.

(k,p)	(1,0.1)	(1,0.5)	(1,0.9)	(10,0.1)	(10,0.5)	(10,0.9)	(80,0.1)	(80,0.5)	(80.0.9)
Parameters fixed between calls.	68	54	57	58	56	63	55	57	62
Parameters re-specified between calls.	210	440	1830	820	1680	8960	3600	8320	50020

CHAPTER 7

Generation from the von Mises Distribution via a comparison of Random Numbers

7.1 Introduction

An angular random variable has the von Mises distribution if its p.d.f. is

$$h(\theta) = \{2\pi I_0(k)\}^{-1} \exp[k \cos(\theta-\theta_0)] , \qquad (7.1.1)$$

for $|\theta| \leq \pi$, $|\theta_0| \leq \pi$ and $k > 0$, where $I_0(k)$ is the modified Bessel function of the first kind and order zero. Without loss of generality we may take $\theta_0 = 0$, and consider a folded von Mises distribution with p.d.f.,

$$f(\theta) = \{\pi I_0(k)\}^{-1} \exp(k \cos \theta) , \qquad (7.1.2)$$

where $0 \leq \theta \leq \pi$ and $k > 0$. Since the distribution function cannot be inverted analytically previous sampling methods have concentrated on envelope rejection procedures. Seigerstetter (1974) proposed a uniform target distribution. This becomes a progressively poorer fit as k increases, with the result that the rejection rate becomes unacceptably high. Additionally the speed of computation is reduced through a logarithmic evaluation. Best and Fisher (1979) consider envelope rejection procedures with Wrapped Normal, Piecewise linear, Polynomial and Cardiod envelopes, but reject these because of deteriorating sampling efficiency or difficulty in generating prospective variates or difficulty in evaluating the target distribution. Instead they advocate the use of a Wrapped Cauchy distribution. This provides a better fit then Seigerstetter, with the acceptance rate lying between $\sqrt{(2\pi/e)}$ and unity, depending on k.

The main aim of this Chapter is to develop a new generator based on a comparison of random numbers. This led to the observation that an envelope rejection method employing a piecewise uniform target distribution could be written with minor changes to the comparison method. Performance measures for these two generators are derived theoretically. Timing experiments are conducted for these two generators and two versions of the Best and Fisher Wrapped Cauchy method. Comparisons are made and conclusions drawn.

7.2 A Random Number Comparison Method

Using the notation of section 1.3.4, the range of θ may be divided into n_k sub-intervals (θ_{j-1},θ_j), $(1 \le j \le n_k)$, with $\theta_0 = 0$ and $\theta_{n_k} = \pi$. n_k is controlled by the interval widths. If $p_j = \text{Prob}(\theta_{j-1} \le \theta \le \theta_j)$, then the von Mises density (7.1.2) may be expressed as the probability mixture,

$$f(\theta) = \sum_{j=1}^{n_k} p_j \, \psi_j(\theta) \quad , \tag{7.2.1}$$

where

$$\psi_j(\theta) = \begin{cases} \exp\{-h_j(\theta)\} \Big/ \displaystyle\int_{\theta_{j-1}}^{\theta_j} \exp\{-h_j(\theta)\}d\theta & (\theta_{j-1} \le \theta \le \theta_j) \\ 0 & \text{(elsewhere)}, \end{cases} \tag{7.2.2}$$

$$h_j(\theta) = k(\cos\theta_{j-1} - \cos\theta) \qquad (\theta_{j-1} \le \theta \le \theta_j) \ , \tag{7.2.3}$$

and

$$p_j = \int_{\theta_{j-1}}^{\theta_j} \exp(k\cos\theta)d\theta \Big/ \int_0^{\pi} \exp(k\cos\theta)d\theta. \tag{7.2.4}$$

The boundary values $\{\theta_j\}$ must be chosen so that $0 \le h_j(\theta) \le 1$, for all j. Interval widths are maximised by $\{\theta_j\}$ satisfying the difference equations:

$$\theta_0 = 0$$
$$\theta_{n_k} = \pi \qquad (7.2.5)$$
$$1 = k(\cos\theta_{j-1} - \cos\theta_j) \quad (1 \le j < n_k).$$

A solution is

$$\theta_j = \cos^{-1}\{1 - (j/k)\} \qquad (0 \le j < n_k)\ , \qquad (7.2.6)$$

which then gives the number of intervals as

$$n_k = \begin{cases} 2k & \text{(2k integer-valued)} \\ < 2k+1 > & \text{(otherwise)}\ . \end{cases} \qquad (7.2.7)$$

The algorithm below generates variates by first selecting the interval (θ_{j-1},θ_j) with probability p_j, and then uses the comparison method, based on (1.3.24) to obtain a variate from $\psi_j(\theta)$.

Algorithm 7.1

0. $n_k = \begin{cases} 2k & \text{(2k integer)} \\ < 2k+1 > & \text{(otherwise)}\ . \end{cases}$

If $n_k > 10$, $n_k = 10$.

$\theta_i = \cos^{-1}\{1-(i/k)\} \qquad (0 \le i < n_k)$.

$p_i = \int_{\theta_{i-1}}^{\theta_i} \exp(k\cos\theta) \,/\, \int_0^{\pi} \exp(k\cos\theta)d\theta \qquad (1 \le i \le n_k)$.

1. Generate $R \sim U(0,1)$. Find smallest j such that $\sum_{i=1}^{j} p_i > R$.

2. $R = (\sum_{i=1}^{j} p_i - R)/p_j$.

3. $\theta = \theta_{j-1} + R(\theta_j - \theta_{j-1})$.

4. Given $\{R_i\}$:

   $N = 1$ if $h_j(\theta) = k - j + 1 - k \cos\theta < R_1$, else

   $N = n$ if $h_j(\theta) \geq R_1 \ldots \geq R_{n-1} < R_n$.

5. If N even, generate $R \sim U(0,1)$ and go to 3.

6. Assign random sign to θ according to the sign of

$$\{R_1 - h_j(\theta)\}/\{1 - h_j(\theta)\} - 0.5 \quad (N = 1)$$

or

$$(R_n - R_{n-1}) \,/\, (1 - R_{n-1}) - 0.5 \quad (N > 1) \ .$$

Deliver θ.

Step 0 is performed just once, and for $k > 0.5$ requires a series of numerical integrations to find the p_i's. A maximum of 10 intervals is employed. For $k > 5$, n_k is set to 10, as the subsequent intervals may safely be dispensed with since numerical evaluation showed that $\text{Prob}(\theta > \theta_{10})$ never exceeded 7.4×10^{-5}. Step 1 selects the correct interval, while steps 2 and 3 generate a uniform variate within that interval, without requiring an extra random number. Step 4 is the comparison phase, the uniform variate being rejected if termination occurs after an even number of comparisons. In this case a further random number R is generated, to sample a new uniform variate from the (same) interval. If the variate is accepted, a random sign is applied in step 6, without requiring an additional random number.

Surprisingly few random numbers are required to generate one variate. Using (1.3.32) (and noting that the random number used for the first prospective uniform variate is also used for interval selection), the mean requirement of random numbers per generated variate is

$$N_1 = \sum_{j=1}^{n_k} \left[\frac{(\theta_j - \theta_{j-1}) + \int_{\theta_{j-1}}^{\theta_j} \exp\{k(\cos\theta_{j-1} - \cos\theta)\}d\theta}{\int_{\theta_{j-1}}^{\theta_j} \exp\{-k(\cos\theta_{j-1} - \cos\theta)\}d\theta} \right] p_j \qquad (7.2.8)$$

$$= \sum_{j=1}^{n_k} \left\{ \frac{(\theta_j - \theta_{j-1})\exp(k\cos\theta_{j-1}) + \exp(2k\cos\theta_{j-1}) \int_{\theta_{j-1}}^{\theta_j} \exp(-k\cos\theta)d\theta}{\int_{\theta_{j-1}}^{\theta_j} \exp(k\cos\theta)d\theta} \right\} p_j$$

$$= \sum_{j=1}^{n_k} \left\{ \frac{(\theta_j - \theta_{j-1})\exp(k-j+1) + \exp[2(k-j+1)] \int_{\pi-\theta_j}^{\pi-\theta_{j-1}} \exp(k\cos\phi)d\phi}{\int_{0}^{\pi} \exp(k\cos\theta)d\theta} \right\}$$

$$= \sum_{j=1}^{n_k} \left\{ \frac{(\theta_j - \theta_{j-1})\exp(k-j+1) + \exp[2(k-j+1)] \int_{\pi-\theta_j}^{\pi-\theta_{j-1}} \exp(k\cos\phi)d\phi}{\pi I_0(k)} \right\} . \qquad (7.2.9)$$

For $k \le 0.5$, $n_k = 1$, $\theta_0 \doteq 0$ and $\theta_1 = \pi$, giving

$$N_1 = [e^{-k} I_0(k)]^{-1} + e^{2k} . \qquad (7.2.10)$$

For $k > 0.5$, with $2k$ integer-valued (7.2.9) is conveniently evaluated since $\pi - \theta_{j-1} = \theta_{2k-j+1}$, giving

$$N_1 = \sum_{j=1}^{n_k} \left\{ \frac{(\theta_j - \theta_{j-1})\exp(k-j+1)}{\pi I_0(k)} + \exp[2(k-j+1)]p_{2k-j+1} \right\} . \qquad (7.2.11)$$

Specimen values of N_1 are displayed in Table 7.1, indicating that the maximum is just in excess of 4, corresponding to $k = 0.5$, dropping to a minimum of 2 when $k = 0$.

Algorithm 7.1 was programmed as a Fortran subroutine VON, which appears in Appendix 6, under the program name VMISES.FOR . For $k > 0.5$, the routine computes $\{p_i\}$ numerically using Simpson's rule. Experiments indicated that the optimum number of ordinates was of the order of 65, and this gives accuracy to at least 6 significant figures.

7.3 A Probability mixture method using envelope rejection with a piecewise uniform target distribution.

Having generated a uniform variate within the selected interval in step 3 of Algorithm 7.1, an alternative method of deciding whether to accept or reject the prospective variate from $\psi_j(\theta)$ is to employ envelope rejection using a uniform target distribution $g_j(\theta) = (\theta_j - \theta_{j-1})^{-1}$, $(\theta_{j-1} \le \theta \le \theta_j)$. The acceptance condition in such a test is

$$RM_j \le \psi_j(\theta)/g_j(\theta) \qquad (7.3.1)$$

where

$$M_j = \max_{\theta_{j-1}\le\theta\le\theta_j} [\psi_j(\theta)/g_j(\theta)] \qquad (7.3.2)$$

$$= \max_{\theta_{j-1}\le\theta\le\theta_j} [\exp\{-h_j(\theta)\}(\theta_j-\theta_{j-1})/\int_{\theta_{j-1}}^{\theta_j} \exp\{-h_j(\theta)\}d\theta\ ]$$

$$= \max_{\theta_{j-1}\le\theta\le\theta_j} [\exp(k\cos\theta)(\theta_j-\theta_{j-1})/\int_{\theta_{j-1}}^{\theta_j} \exp(k\cos\theta)d\theta]$$

$$= \frac{(\theta_j-\theta_{j-1})}{p_j\,\pi\,I_0(k)} \max_{\theta_{j-1}\le\theta\le\theta_j} [\exp(k\cos\theta)]$$

$$= \frac{(\theta_j-\theta_{j-1})\,\exp(k\cos\theta_{j-1})}{p_j\,\pi\,I_0(k)}$$

$$= \frac{(\theta_j-\theta_{j-1})\,\exp(k-j+1)}{p_j\,\pi\,I_0(k)}\ . \qquad (7.3.3)$$

Then (7.3.1) becomes

$$R\,e^{k-j+1} \le e^{k\cos\theta}$$

or

$$E \ge -k\cos\theta + k - j + 1\ , \qquad (7.3.4)$$

where E is a standard Exponential variate.

Steps 4, 5 and 6 in Algorithm 7.1 may therefore be replaced by new steps shown in Algorithm 7.2 below, to give an alternative generation procedure.

Algorithm 7.2

4. generate E, a standard Exponential variate.

5. If $E < -k \cos\theta + k - j + 1$, generate $R \sim U(0,1)$ and go to 3.

6. Generate $R_1 \sim U(0,1)$. If $R_1 > 0.5$, $\theta = -\theta$. Deliver θ.

In this instance an extra random number R_1 is required to assign a random sign to the variate. In addition two random numbers are required for each prospective variate and therefore the (unconditional) mean requirement of random numbers in this method is

$$N_2 = 1 + \sum_{j=1}^{n_k} 2M_j p_j ,$$

which from (7.3.3) becomes,

$$N_2 = 1 + 2[\pi e^{-k} I_0(k)]^{-1} \sum_{j=1}^{n_k} (\theta_j - \theta_{j-1}) e^{1-j} . \qquad (7.3.5)$$

Specimen values of N_2 are shown in Table 7.1 . For most values of k there is a slight increase (over the comparison method) in the mean requirement rate of random numbers, due to the necessity of employing an extra one for assigning the sign to the variate. The method requires at least one logarithmic evaluation per generated variate. Given that the set up procedure is exactly the same as in the comparison method, the theoretical analysis above suggests that the comparison method will probably be somewhat faster.

Algorithm 7.2 was programmed as a Fortran subroutine VON, which appears in Appendix 6, under the program name UNIVON.FOR .

7.4 Best and Fisher's Wrapped Cauchy Method

Best and Fisher's method uses a Wrapped Cauchy Target distribution with p.d.f.

$$g(\theta) = \frac{1 - \rho^2}{\pi(1+\rho^2-2\rho \cos \theta)} \quad , \qquad (7.4.1)$$

where $0 \leq \theta \leq \pi$ and $0 \leq \rho \leq 1$. They show that the acceptance probability is maximised by selecting

$$\rho = \rho^* = \{\tau - (2\tau)^{\frac{1}{2}}\}/2k \quad , \qquad (7.4.2)$$

where

$$\tau = 1 + (1+4k^2)^{\frac{1}{2}} \quad ,$$

and that the acceptance probability is

$$M^{-1} = (1-\rho^{*2}) I_0(k)/\{(2\rho/k)\exp[\{k(1+\rho^{*2})/2\rho^*\} - 1]\} \quad . \qquad (7.4.3)$$

The algorithm requires two random numbers for each prospective variate, plus one random number for assigning a random sign in the final step. Thus the mean requirement rate of random numbers is

$$N_3 = 2M + 1 \quad . \qquad (7.4.4)$$

Specimen values of N_3 are shown in Table 7.1, indicating that apart from $k = 0.5$ and $k = 1.0$, slightly more are required than in the comparison method. This is due in part to the random number required for assigning a random sign. The algorithm, described by Best and Fisher, incorporates a pre-test to reduce the number of logarithmic evaluations. The algorithm was programmed as Fortran subroutine BFISH and appears in Appendix 6 under the program name BFISH.FOR .

For each prospective variate, the cosine of a variate uniformly distributed in $(0,\pi)$ is required. Best and Fisher suggest that on some computers it may be quicker to evaluate the cosine indirectly using the polar method of Marsaglia and Bray (1964). Although the same number of random numbers (two) is required for each prospective

variate, the acceptance probability is reduced by the ratio $\pi/4$, and so the mean requirement rate of random numbers is now

$$N_4 = 2M(4/\pi) + 1 , \qquad (7.4.5)$$

where M is evaluated using (7.4.3). Specimen values of N_4 are shown in table 7.1 . This version was programmed as Fortran subroutine BFISHC and appears in Appendix 6 under the program name BFISHC.FOR .

7.5 Verification and Validation of Algorithms

Experiments were conducted to verify and validate the four routines for values of k in the range 0-20. One method of verification used was to generate samples of size 10,000 and to record the sample mean requirement of random numbers per generated variate, this value to be compared with the theoretical mean. The results are shown in table 7.1 . In the case of VON (mixture method with uniform rejection), BFISH and BFISHC, where the number of trials per generated variate is Geometrically distributed, the standard error is known to be $\sqrt{\{4p/(10000q^2)\}}$, where p and q are the probabilities of rejection and acceptance respectively in any trial. No significant differences between these sample and population means were detected. For the comparison method, where the process leading to acceptance is more complicated, expressions for the standard error become cumbersome. However inspection of differences between sample and theoretical means suggest good agreement.

The generators were validated by performing tests of fit on the output observations. A sample size of 10,000 was chosen since simulations using such generators could conceivably require that number (or even more) of variates. With such sample sizes, the

Kolmogorov-Smirnoff test is difficult to implement, due to the problems of storing and ordering the observations, and the large number of numerical integrations involved. For this reason a chi-squared goodness of fit test was employed. Results are shown in table 7.2 . With the exception of $k = 5$ for the routine BFISH, there is no evidence to suggest that these samples are not drawn from the parent distributions. A further sample for BFISH when $k = 5$ gave a chi-squared test statistic value of 53.7 with 42 degrees of freedom.

Results of these verification and validation experiments were obtained using program VONVER.FOR . This, together with output listings (which include sample and theoretical c.d.f.s) appears in the supporting material.

Table 7.1 Theoretical (sample†) Mean Rate of Random Numbers per generated Variate.

Routine \ k	0*	0.2	0.5	1	2	5	10	20
VON (Comparison of Random Nos.)	2.000 (2.000)	2.701 (2.697)	4.269 (4.249)	3.871 (3.880)	3.581 (3.573)	3.466 (3.438)	3.466 (3.447)	3.437 (3.433)
VON (mixture method with envelope rejection)	3.000 (3.000)	3.419 (3.409)	4.101 (4.105)	3.937 (3.927)	3.812 (3.800)	3.760 (3.769)	3.752 (3.761)	3.748 (3.781)
BFISH (Best and Fisher)	3.000 (3.001)	3.019 (3.021)	3.106 (3.104)	3.304 (3.294)	3.613 (3.606)	3.883 (3.870)	3.964 (3.957)	4.002 (3.993)
BFISHC (Best and Fisher with polar cosine evaluation).	3.547 (3.542)	3.571 (3.568)	3.681 (3.674)	3.934 (3.923)	4.327 (4.306)	4.671 (4.642)	4.773 (4.762)	4.823 (4.813)

† sample size is 10,000 in each experiment.

* A value of k = 0.001 was used for BFISH and BFISHC.

Table 7.2 Tests of Fit for von Mises Generators

Chi-squared statistic
(Degrees of freedom)

Routine \ k	0*	0.2	0.5	1	2	5	10	20
VON (Comparison of Random Nos.).	49.1 (49)	112.0 (99)	71.8 (99)	67.3 (99)	81.2 (99)	37.2 (42)	48.0 (32)	27.2 (20)
VON (mixture method with envelope rejection).	52.2 (49)	81.4 (99)	110.2 (99)	95.1 (99)	102.2 (99)	26.1 (42)	31.5 (32)	16.4 (20)
BFISH (Best and Fisher).	90.4 (99)	89.7 (99)	107.4 (99)	108.4 (99)	94.5 (99)	73.2 (42)	42.4 (32)	25.6 (20)
BFISHC (Best and Fisher with polar cosine evaluation).	87.4 (99)	74.2 (99)	87.7 (99)	105.0 (99)	72.2 (99)	34.6 (42)	37.5 (32)	18.9 (20)

* A value of 0.001 was used for BFISH and BFISHC.

7.6 Timing Experiments and Conclusions

Timing experiments were conducted for all four routines, for values of k between 0 and 20. Tables 7.3 and 7.4 give mean execution times per variate, when k is not respecified and k respecified respectively between calls. In all cases timings are based on the generation of 10000 variates, with the exception of VON (comparison) and VON (mixture method with envelope rejection) in table 7.4, for which 500 variates were generated when $0.5 < k < 5$ and 200 variates when $k \geq 5$.

When k is not respecified between calls, the comparison method VON is uniformly fastest for all values of k. For most values of k the savings over the best of BFISH and BFISHC are significant. This behaviour can be explained by the lower requirement of random numbers (except for $k = 0.5$) and because no logarithmic or inverse cosine evaluations are required. Random number generation takes approximately 13 μsecs and the effect of using a slower random number generator would be to increase the difference in times (except for $k = 0.5$). On a machine where log and inverse cosine evaluations are faster, the effect would be to reduce this difference.

Again, with reference to table 7.3, the polar cosine variant BFISHC appears to be marginally faster than the conventional method BFISH on this machine. The mixture method with envelope rejection is uniformly slower than the comparison method, and so the latter is certainly preferred, in view of the similar set-up characteristics for these two algorithms.

When k is respecified between calls (table 7.4), the comparison method is fastest when $k \leq 0.5$. The difference in time between this and BFISHC is more marked than in table 7.3, since some significant set-up is required in the latter case. Since only one interval is required in the comparison method, no numerical integrations are required, and consequently the set-up time is small. When $k > 0.5$, numerical integrations are required in the comparison method, involving large set-up times, table 7.3 indicating that BFISHC is very much faster.

We conclude that when many variates are required and portability is not critical, then the comparison method is preferred. When either of these conditions is not satisfied a hybrid algorithm using VON (comparison) and BFISHC for $k \leq 0.5$ and $k > 0.5$ respectively is indicated. Such an algorithm is easy to program, particularly for $k \leq 0.5$, since the comparison method requires no numerical integrations.

Finally, we note that the comparison method offers the advantage that interval selection can be controlled by reserving a stream for that purpose. This would apparently aid the generation of high quality antithetic variates. However it should be noted, that for most values of k, most of the probability is concentrated in the first interval, and so the induced negative correlation is unlikely to be as large as one might hope for.

Table 7.3 Time (μsecs) to generate 1 Von Mises Variate on a DEC-20, when parameter k not re-specified between calls.

Routine \ k	0*	0.2	0.5	1	2	5	10	20
VON (Comparison of Random Nos.).	132	160	204	196	186	180	181	182
VON (mixture method with envelope rejection).	179	208	260	249	238	241	232	238
BFISH (Best and Fisher).	199	205	213	236	261	285	295	298
BFISHC (Best and Fisher with polar cosine evaluation).	198	193	207	224	249	275	277	282

* A value of k = 0.001 was used for BFISH and BFISHC.

Table 7.4 Time (m secs) to generate 1 Von Mises Variate on a DEC-20, when parameter k is respecified between calls.

Routine \ k	0*	0.2	0.5	1	2	5	10	20
VON (Comparison of Random Nos.).	0.145	0.167	0.223	15.5	31.0	79.4	77.8	79.5
VON (Mixture method with envelope rejection).	0.196	0.224	0.282	15.3	30.9	77.5	77.5	77.7
BFISH (Best and Fisher).	0.274	0.278	0.286	0.312	0.340	0.359	0.369	0.374
BFISHC (Best and Fisher with polar cosine evaluation).	0.270	0.276	0.290	0.312	0.326	0.351	0.359	0.376

* A value of k = 0.001 was used for BFISH and BFISHC.

CHAPTER 8

Sampling Variates from the tail of Gamma and Normal distributions

8.1 Introduction

In this chapter a method is described for generating variates from the tail of a Gamma distribution (shape parameter > 1). The approach used was first described in Dagpunar (1977, 1978). Similar ideas may be applied to the Normal distribution and these are developed in section 8.6 .

Tail generation from the Gamma is equivalent to sampling from the truncated Gamma distribution,

$$h_Z(z) = \frac{(\lambda z)^{\alpha-1}\lambda e^{-\lambda z}}{\int_s^\infty (\lambda u)^{\alpha-1}\lambda e^{-\lambda u}du} \qquad (z > s), \qquad (8.1.1)$$

where $\alpha > 1$ and $s(>0)$ is a specified constant. The problem reduces to sampling from

$$f_X(x) = x^{\alpha-1}e^{-x} / \int_t^\infty v^{\alpha-1}e^{-v}dv \qquad (x > t), \qquad (8.1.2)$$

where $t = \lambda s$, and setting $Z = X/\lambda$. Methods for sampling from a complete Gamma distribution were reviewed in section 2.3 . The requirement that the sampled variate be greater than t, may be met simply by using one of these existing methods and rejecting all variates having values not exceeding t. Unfortunately this naive approach will be highly inefficient if t is large. The method described in the next section is envelope rejection-based, and eliminates the need to sample from the complete Gamma distribution. The efficiency of the approach relative to the naive approach will be shown to increase markedly with increasing t.

8.2 Method

First stage sampling is from the distribution

$$g_Y(x) = \mu e^{-\mu(x-t)} \qquad (x > t,\ \mu > 0) \ . \tag{8.2.1}$$

This is performed by putting $Y = t + (E_1/\mu)$, where E_1 is a standard negative Exponential variate. Y is then accepted subject to

$$RM < f_X(Y)/g_Y(Y) \ ,$$

where $R \sim U(0,1)$ and

$$M = \max_{x>t} [\, f_X(x)/g_Y(x) \,] \ .$$

The efficiency of the method is M^{-1}, so it is desirable to choose μ such that M is minimised. The best value of μ is therefore given by the saddle point of the function $f_X(x)/g_Y(x)$, giving a maximum with respect to x and a minimum with respect to μ.

In this case,

$$\frac{f_X(x)}{g_Y(x)} = \frac{x^{\alpha-1} e^{-[x(1-\mu)+\mu t]}}{\mu \int_v^\infty v^{\alpha-1} e^{-v} dv} \ . \tag{8.2.2}$$

It is easily shown that such a saddle point occurs at

$$x = (\alpha-1)/(1-\mu) \ , \tag{8.2.3}$$

and

$$\mu^{-1} = x - t \ . \tag{8.2.4}$$

On eliminating x, the optimum value of μ is given by

$$\mu^* = [\, (t-\alpha) + \{(t-\alpha)^2 + 4t\}^{\frac{1}{2}} \,]/2t \ . \tag{8.2.5}$$

With this choice of μ,

$$M = M_t = \frac{(\alpha-1)^{\alpha-1} e^{-(\alpha-1+\mu^* t)}}{(1-\mu^*)^{\alpha-1} \mu^* \int_t^\infty v^{\alpha-1} e^{-v} dv} \ . \tag{8.2.6}$$

The condition for acceptance becomes

$$\frac{R(\alpha-1)^{\alpha-1}e^{-(\alpha-1)}}{(1-\mu^*)^{\alpha-1}} < Y^{\alpha-1}e^{-Y(1-\mu^*)} ,$$

which after some reduction becomes

$$E_2 > (1-\mu^*)Y - (\alpha-1)[1 + \ell nY + \ell n\{(1-\mu^*)/(\alpha-1)\}], \qquad (8.2.7)$$

where $E_2 = -\ell nR$. Thus a scheme for sampling variates is shown in Algorithm 8.1 .

Algorithm 8.1

0. $\mu^* = [(t-\alpha) + \{(t-\alpha)^2+4t\}^{\frac{1}{2}}]/2t$.

   $a = 1 - \mu^*$.

   $b = \ell n[a/(\alpha-1)]$.

1. Generate E_1 and E_2, a pair of standard Negative Exponential variates. $Y = t + E_1\mu^{*-1}$.

2. If $E_2 > aY - (\alpha-1)(1+\ell nY+b)$ deliver $X = Y$, else go to 2.

If the simulation involves repeated sampling from the same truncated distribution, then step 0 is performed just once, and the values for μ^*, a and b are saved between calls.

Schmeiser (1980), apparently unaware of the existence of this algorithm (published in 1978), has proposed its use with a value $\mu^* = 1 - \{(\alpha-1)/t\}$. This represents a non-optimal Exponential envelope. Schmeiser inserted an acceptance pre-test, resulting in the algorithm being valid only when $t \geq \alpha - 1 + \sqrt{(\alpha-1)}$, whereas the version given here is valid for all t.

8.3 Special Cases

The procedure for sampling from a complete Gamma distribution is obtained by taking the case $t = 0$. From (8.2.5),

$$\mu^* = \lim_{t\to 0} \left\{ [(t-\alpha)+\{(t-\alpha)^2+4t\}^{\frac{1}{2}}]/2t \right\} = \alpha^{-1} \quad . \tag{8.3.1}$$

Hence first stage sampling is from an Exponential distribution having the same mean as the Gamma distribution. The condition for acceptance is given by putting $\mu^* = 1/\alpha$ and $Y = \alpha E_1$ into (8.2.7) to yield

$$E_2 > (\alpha-1)(E_1-1-\ell n E_1) \quad , \tag{8.3.2}$$

which is the acceptance condition given by Fishman (1976a) in his method for generating Gamma variates. The efficiency is obtained by putting $t = 0$ and $\mu^* = \alpha^{-1}$ into (8.2.6) to yield

$$M_0^{-1} = \Gamma(\alpha) e^{\alpha-1} \alpha^{-\alpha} \quad . \tag{8.3.3}$$

A further special case is obtained by taking the truncation point to be the mode of the Gamma distribution. In this case $t = \alpha - 1$ and the optimum value of μ is

$$\mu^* = \{(1+4t)^{\frac{1}{2}} - 1\}/2t \quad . \tag{8.3.4}$$

The acceptance condition becomes

$$E_2 > -t[1 + \ell n\{(1-\mu^*)/t\}] + (1-\mu^*)Y - t\ \ell n\ Y \quad . \tag{8.3.5}$$

Results (8.3.4) and (8.3.5) are identical to those derived by Atkinson (1977) for sampling from a complete Gamma distribution. In that method first stage sampling is from a composite distribution involving a uniform density for $x \le t = \alpha - 1$, and an Exponential density for $x > t$.

8.4 Relative Efficiency

In assessing the efficiency of the proposed method for sampling variates from the truncated distribution, it is useful to compare it with the naive approach mentioned in the introduction. For $\alpha \geq 1$, an efficient scheme for generating Gamma variates is Cheng's method with a mean requirement of m trials per generated variate where

$$m = \frac{4\alpha^{\alpha} e^{-\alpha}}{(2\alpha-1)^{\frac{1}{2}}\Gamma(\alpha)} \quad . \tag{8.4.1}$$

We define the Relative Efficiency $R_t(\alpha)$ of Algorithm 8.1, with respect to sampling via Cheng's algorithm (rejecting values less than t), as the ratio of the mean requirement of trials per generated variate under the two methods. The Relative Efficiency is an indicator of how the relative speeds of the two methods change as α and t vary. Given this definition,

$$R_t(\alpha) = \frac{m\Gamma(\alpha)}{M_t \int_t^{\infty} v^{\alpha-1} e^{-v} dv} \quad . \tag{8.4.2}$$

From (8.2.6) and (8.4.1),

$$R_t(\alpha) = \left\{\frac{4\alpha^{\alpha}\mu^* e^{(\mu^* t-1)}}{(2\alpha-1)^{\frac{1}{2}}}\right\}\left\{\frac{1-\mu^*}{\alpha-1}\right\}^{\alpha-1} \quad . \tag{8.4.3}$$

When $t = 0$, $\mu^* = \alpha^{-1}$ giving,

$$R_0(\alpha) = 4e^{-1}/(2\alpha-1)^{\frac{1}{2}} \quad . \tag{8.4.4}$$

When $\alpha = 1$,

$$\lim_{\alpha \to 1} \{(1-\mu^*)/(\alpha-1)\} = (t+1)^{-1} \quad ,$$

giving

$$R_t(1) = 4e^{t-1} \quad . \tag{8.4.5}$$

Table 8.1 shows specimen values of $R_{t_p}(\alpha)$, where t_p is the p^{th} percentile of the distribution.

Table 8.1 Relative efficiency, $R_{t_p}(\alpha)$, of Algorithm 8.1 with respect to Cheng's algorithm.

p \ α	1	3	10	50
0	1.47	0.66	0.34	0.15
25	1.96	1.32	1.13	1.03
50	2.94	2.15	1.92	1.80
75	5.89	4.53	4.15	3.96
90	14.72	11.61	10.79	10.40
95	29.43	23.45	21.91	21.20
99	147.15	118.51	111.60	108.58

For fixed α, the relative efficiency increases rapidly with increasing t, because the Exponential target distribution is providing a progressively better fit to the truncated distribution and because the naive method rejects a larger proportion of variates (i.e. those having values less than t). For truncation at a fixed percentile, the relative efficiency decreases (slowly) as α increases, because the fit between target and truncated distribution becomes worse and because the sampling efficiency (m^{-1}) in Cheng's method increases slowly. However $R_{t_p}(\alpha)$ only approaches one or less when the truncation point is low or α is high. We conclude from the table that algorithm 8.1 is likely to be highly efficient for most combinations of p and α.

8.5 Computational Experience

Algorithm 8.1 was programmed as a Fortran subroutine TRUNG which appears in Appendix 7 under the program name GATRUN.FOR. To reduce the number of logarithmic evaluations, an acceptance pre-test based on the bound $E_2 = -\ell n\, R_2 > 2(1-R_2)/(1+R_2)$, where R_2 is a random number, was incorporated into the routine. Timings were carried out for various α and truncation points t_p, in the manner described in section 3.4. Table 8.2 gives the mean execution time per generated variate.

Table 8.2 Mean time (μsecs) to generate one Gamma variate, shape parameter α, truncated at t_p, on a DEC-20 computer.

t_p \ α	$1+\varepsilon^{\dagger}$	3	10	50
$\varepsilon^{\dagger}$	157	310	621	1445
25	160	195	228	244
50	154	181	196	205
75	154	168	173	180
90	157	165	168	174
95	161	163	168	174
99	163	160	162	166

$^{\dagger}\ \varepsilon = 0+$

The speed of the routine varies little for most values of α and p. With six exceptions all timings lie between 154 and 196 μsecs, reflecting the fact that the optimal Exponential envelope provides a good fit. When $\alpha \gtrsim 3$ and the truncation point is close to zero, the

routine becomes slower, due to a poorer fit. In these cases a better approach would be to sample from the complete Gamma distribution using one of the several efficient methods described in section 2.3 .

8.6 Tail Generation from the Normal Distribution

The method described for the Gamma distribution can also be applied to other distributions having an infinite domain. For example, to generate from the truncated Normal distribution,

$$f_X(x) = e^{-\frac{1}{2}x^2} / \int_a^\infty e^{-\frac{1}{2}u^2}\, du \qquad (x > a) \ , \tag{8.6.1}$$

where a is a non-negative constant, we may employ a target distribution,

$$g_Y(x) = \lambda e^{-\lambda(x-a)} \qquad (x > a,\ \lambda > 0) \ . \tag{8.6.2}$$

Thus

$$\frac{f_X(x)}{g_Y(x)} = \frac{e^{-\frac{1}{2}x^2}\, e^{\lambda(x-a)}}{\lambda \int_a^\infty e^{-\frac{1}{2}u^2}\, du} \ , \tag{8.6.3}$$

which has a saddle point (maximum and minimum with respect to x and λ respectively) at

$$x = \lambda \tag{8.6.4}$$

and

$$\lambda^* = \{a + \sqrt{(a^2+4)}\}/2 \ . \tag{8.6.5}$$

Choosing λ in this optimal fashion gives

$$M = \frac{e^{(\frac{1}{2}\lambda^{*2} - \lambda^* a)}}{\lambda^* \int_a^\infty e^{-\frac{1}{2}u^2}\, du} \ , \tag{8.6.6}$$

from which the sampling efficiency may be calculated.

Prospective variates Y are generated by setting $Y = a + (E_1/\lambda^*)$, where E_1 is a standard Negative Exponential variate. Given a random number R, (8.6.3) and (8.6.6) lead to the acceptance condition,

$$R < e^{-\frac{1}{2}(Y-\lambda^*)^2} \qquad (8.6.7)$$

or

$$E_2 > \tfrac{1}{2}(Y-\lambda^*)^2 , \qquad (8.6.8)$$

where E_2 is a standard Negative Exponential variate. A simple pre-test for (8.6.7) is

$$2(1-R)/(1+R) > \tfrac{1}{2}(Y-\lambda^*)^2 . \qquad (8.6.9)$$

If this is not satisfied, then the logarithmic evaluation for E_2 in (8.6.8) is required. Algorithm 8.2 below gives a scheme for generating the variates.

Algorithm 8.2

0. $\lambda^* = \{a + \sqrt{(a^2+4)}\}/2$.
1. generate $R \sim U(0,1)$ and E a standard Negative Exponential variate. $Y = a + (E/\lambda^*)$.
2. If $2(1-R)/(1+R) > \tfrac{1}{2}(Y-\lambda^*)^2$ deliver $X = Y$.
3. If $-\ell n\ R > \tfrac{1}{2}(Y-\lambda^*)^2$ deliver $X = Y$, else go to 1.

We remark that if instead of choosing an optimal value of λ given by (8.6.5), a value of $\lambda^* = a$ is used, then the method due to Schmeiser (1980) results. In this case the acceptance condition (8.6.7) becomes

$$R < e^{-\frac{1}{2}(Y-a)^2} , \qquad (8.6.10)$$

where $Y = a - \{(\ell n\ R_1)/a\}$, and R_1 is a random number. Since (8.6.10) can be written as

$$R < \frac{e^{-\frac{1}{2}(Y^2-a^2)}}{e^{-a(Y-a)}} \quad , \tag{8.6.11}$$

the denominator of the right hand side being R_1, the acceptance condition becomes

$$R < \frac{e^{-\frac{1}{2}(Y^2-a^2)}}{R_1} \quad . \tag{8.6.12}$$

When $Y > a > 1$, a suitable lower bound on $\exp\{-0.5(Y^2-a^2)\}$ is $1 - a(Y-a)$, and so Schmeiser uses an acceptance pre-test $\{1 - a(Y-a)\}/R_1 > R$. Schmeiser's algorithm results in a poorer sampling efficiency (since a non-optimal value of λ is used), and the pre-test restricts use of the method to $a \geq 1$. In connection with Schmeiser's method, it should be noted that on pp. 1015, $\kappa(x) = -(x-a)^2/a$ should read $\kappa(x) = (a^2-x^2)/2$.

Algorithm 8.2 requires either one or two logarithmic evaluations per trial. The M-value never exceeds $\sqrt{(2e/\pi)}$, and rapidly approaches one as a increases. In contrast, Marsaglia's method (2.2.13)-(2.2.14) requires only one logarithmic evaluation per trial, but has a uniformly higher value of M, which approaches infinity as $a \to 0$. Finally, a point which appears to have been overlooked by Schmeiser is that his method has an identical sampling efficiency to Marsaglia's. This may be verified by putting $\lambda^* = a$ in (8.6.6) and comparing with (2.2.12). Table 8.3 below gives specimen M-values for algorithm 8.2 (Dagpunar), Marsaglia's and Schmeiser's methods.

Table 8.3 M-values using optimal Exponential envelope (Dagpunar), Marsaglia's method and Schmeiser's method.

Method \ a	0	0.01	0.1	0.5	1	2	3	5	10
Dagpunar	1.3155	1.3129	1.2900	1.2082	1.1409	1.0711	1.0407	1.0175	1.0048
Marsaglia	∞	80.428	8.6262	2.2808	1.5251	1.1866	1.0944	1.0373	1.0098
Schmeiser	Not applicable				1.5251	1.1866	1.0944	1.0373	1.0098

Algorithm 8.2, Marsaglia's and Schmeiser's methods were programmed as Fortran subroutines TRUNCN, under the program names TRUNCN.FOR, TRUNCM.FOR and TRUNCS.FOR respectively. Program listings appear in Appendix 7. Timings were made in the manner described in section 3.4 . Table 8.4 gives the mean execution time per generated variate.

Table 8.4 Mean time (μsecs) to generate one Normal variate truncated at a, on a DEC-20 computer, using an optimally designed Exponential envelope (Dagpunar), Marsaglia's method and Schmeiser's method.

Method \ a	0	0.01	0.1	0.5	1	2	3	5	10
Dagpumar	150	150	148	140	131	120	114	111	108
Marsaglia	-	8555	921	250	172	135	125	119	117
Schmeiser	Not Applicable				185	147	136	130	128

Algorithm 8.2 was found to be uniformly faster than the other two methods. The difference between this algorithm and Marsaglia's is most evident when $a < 0.5$, corresponding to a very much lower sampling efficiency for the latter. For other values of a, algorithm 8.2 is still faster, presumably because the higher sampling efficiency more than compensates for the slightly larger number of logarithmic evaluations per trial. In comparing Schmeiser's and Marsaglia's methods, we note that the slightly larger number of logarithmic evaluations required by the former seems to explain its poorer performance.

From these timing results and the analysis, we conclude that algorithm 8.2 provides a fast generation procedure, which is robust for all values of a, and that these features are not jointly exhibited by the other two methods.

APPENDIX 1.PROGRAM LISTINGS OF
ZIPINF.FOR,ZIPFRP.FOR.

```
C
C
C           PROGRAM: ZIPINF.FOR
C
C
C
C     ROUTINE GENERATES ZETA VARIATES WITH
C     PDF(N)=A*(N**(-(1.0+RHO))).USES INVERSION
C     WITH CHOP - DOWN SEARCH.NOT SUITABLE FOR RHO<=1,
C     AS MEAN IS THEN INFINITE.
C
      SUBROUTINE IZ2(RHO,A,N)
      P=A
      R=RAN(Z)
      N=1
80    IF(R.LE.P) RETURN
      R=R-P
      N=N+1
      P=P*(((N-1)/FLOAT(N))**(RHO+1.0))
      GOTO 80
      END
```

```
C
C
C          PROGRAM: ZIPFRP.FOR
C
C
C
C       ROUTINE GENERATES ZETA VARIATES USING
C       ENVELOPE REJECTION WITH PARETO TARGET DISRIBUTION
C       TRUNCATED AT 0.5. PMF(N)=A*(N**-(1.0+RHO)). NOTE
C       THAT VALUE OF A IS NOT REQUIRED IN THIS METHOD. JC
C       MUST BE SET TO 1 ON FIRST CALL, TO AN INTEGER >1
C       ON SUBSEQUENT CALLS.IF N WOULD OVERFLOW,THEN
C       LOG(VARIATE) IS DELIVERED IN ZLOG, WITH N=0,
C       OTHERWISE N IS DELIVERED WITH ZLOG=-1.0.VERSION
C       INCLUDES ACCEPTANCE AND REJECTANCE PRETESTS.
C
            SUBROUTINE IZ1(RHO,JC,N,ZLOG)
            COMMON/SETUP/ALPHA,GA,BET,A,B,C,D,G,H,P,S,FLOW
            N=0
            ZLOG=-1.0
            IF(JC.GT.1) GOTO 10
            ALPHA=1.0+RHO
            GA=1.0/RHO
            BET=(2.0/3.0)**ALPHA
            A=ALPHA*ALOG(6.0)
            B=ALPHA*ALOG(.25)
            C=ALPHA*3.0*ALOG(1.5)
            D=(-2.0*C)/3.0
            G=ALPHA*(ALOG(1.5)+1.0)
            H=-ALPHA
            P=C/3.0
            S=ALOG(.5)
            FLOW=0.0
            IF(RHO.GT.1) GOTO 10
            FLOW=(.5/FLOAT(34359738365))**RHO
   10       R1=RAN(Z)
            R2=RAN(Z)
            IF(R1.LT.FLOW) GOTO 90
            FG=-ALOG(R2)
            Y=.5/(R1**GA)
            N=IFIX(Y+.5)
            W=Y/FLOAT(N)
C
C           ACCEPTANCE PRETESTS
C
            IF(W.GE.1.)GOTO 60
            IF(FG-(B*W).GE.A) RETURN
            GOTO 70
   60       IF(FG-(D*W).GE.C) RETURN
C
CC          REJECTANCE PRETEST
C
C
   70       IF(FG-(H*W).LT.G) GOTO 10
C
```

```
C       ACCEPTANCE TEST
C
        IF(FG+ALPHA*ALOG(W).GT.P) RETURN
        GOTO 10
90      IF(R2.GE.BET) GOTO 10
        ZLOG=S-GA*ALOG(R1)
        RETURN
        END
```

Appendix 2 Tests of Uniformity and Serial Correlation on Random Number generator RAN(•).

This appendix describes tests made on the Fortran pseudo random number function RAN(•), available on the DEC-20 computer. The aim is to determine whether the random number generator is adequate for the purposes of testing the performance of the random variate generators developed in this thesis. Two empirical tests were applied. The first was designed to determine whether the numbers were uniformly distributed. Evidence of significant long-term or short-term deviation was sought. The second test was designed to investigate serial correlation, and because many methods of variate generation require more than one random number per generated variate, our main concern was to detect any significant serial correlations of low order (up to 20 say).

Uniformity

In this case n_r realisations, each containing n_s random numbers were generated. Considering the i^{th} realisation, an empirical c.d.f. $\hat{F}^{(i)}(\cdot)$ was calculated and compared with the population c.d.f. using a Kolmogorov-Smirnov statistic,

$$K_{n_s}^{(i)} = \max_{0\le x\le 1} \left| n_s^{\frac{1}{2}} \{\hat{F}^{(i)}(x) - x\} \right| \qquad (i = 1,2,\ldots,n_r). \qquad \text{(A2.1)}$$

The $K_{n_s}^{(i)}$ values were used to compute an empirical c.d.f. $\hat{G}(\cdot)$, and this was compared with the asymptotic distribution of $K_{n_s}^{(i)}$ (under the assumption of uniformity and independence of the random numbers) which is

$$G(w) = 1 - 2 \sum_{r=1}^{\infty} (-1)^{r-1} e^{-2r^2w^2} . \qquad \text{(A2.2)}$$

This comparison is via the Kolmogorov-Smirnov statistic

$$\Delta_{n_r} = \max_{0 \le w \le \sqrt{n_s}} \left| \hat{G}(w) - G(w) \right| . \qquad \text{(A2.3)}$$

For values of $n_s \gtrsim 500$, the asymptotic distribution is sufficiently accurate. Δ_{n_r} was compared with critical values of the one sample Kolmogorov-Smirnov distribution at the 5% level.

This approach is recommended by Knuth (1968, pp 45). For small n_r the test is useful in detecting global deviation from uniformity, but not necessarily local deviation. The power of detecting the latter is thought to increase as n_r increases. Results for various values of n_r and n_s are shown in table A2.1 and were obtained from program UNIFOR.FOR. This together with output listings appears in the supporting material.

Table A2.1 Results of tests on Uniformity

n_r	n_s	Δ_{n_r}	Critical Value at 5% level
20	500	0.182	0.294
10	1000	0.409	0.409
5	2000	0.453	0.563
2	5000	0.741	0.842
1	10000	0.873	0.975
25	1000	0.169	0.264
12	2000	0.219	0.375
5	5000	0.288	0.563
3	10000	0.412	0.708

These results give no evidence of significant local or global deviation from uniformity.

Serial Correlation

Consider a realisation of k random numbers $R_1, R_2, \ldots, R_k$. Under a hypothesis of uniformity and independence, the statistic $Z_k = 12(n-k)^{\frac{1}{2}} V_k$, where

$$V_k = (n-k)^{-1} \sum_{i=1}^{n-k} (R_i - \tfrac{1}{2})(R_{i+k} - \tfrac{1}{2}), \qquad \text{(A2.4)}$$

and $n \gg k$, is distributed approximately as $N(0,1)$.

Two independent realisations of length 10000 were generated and values $Z_k^{(1)}$, $Z_k^{(2)}$ calculated for $k = 1,2,\ldots, 250$. Excessively high values of $|Z_k|$ would tend to indicate that there is a non-zero serial correlation of order k. Values for Z_k were obtained from a program SERIAL.FOR . This together with output listings appears in the supporting material. Table A2.2 gives (i) a listing of $Z_k^{(1)}$, $Z_k^{(2)}$ for $k = 1,2,\ldots,25$ and (ii) a listing of Z_k for those instances where $|Z_k| > 1.96$ and $k \leq 250$.

For low orders, there appears to be little evidence of serial correlation, apart perhaps from $k = 25$, although Z_{25} is well within limits on the first realisation. For higher orders, in those instances where a high $|Z_k|$ value was recorded in one realisation, a similar magnitude was never obtained in the other realisation, apart from one case, $k = 247$. These results suggest that there is little evidence of significant departures from non-zero serial correlation for orders less than 250.

Table A2.2 Summary of Results of test on Serial Correlations

(a) Values of $Z_k^{(1)}$, $Z_k^{(2)}$ $(k \leq 25)$

k	1	2	3	4	5	6	7	8	9	10
$Z_k^{(1)}$	0.4115	-1.0830	-0.0082	-0.0153	0.3110	0.7388	-1.6881	-0.3439	-0.8078	-0.5835
$Z_k^{(2)}$	1.0760	-0.4122	-0.4769	-0.0346	1.5953	0.4826	1.4586	-2.0082	0.9043	0.8764

k	11	12	13	14	15	16	17	18	19	20
$Z_k^{(1)}$	-0.0573	-0.1065	0.7687	0.3379	0.6047	-0.3965	-0.5320	1.1159	-0.7190	0.6294
$Z_k^{(2)}$	0.5963	-0.0197	-0.9510	0.0229	-0.9497	1.0508	0.2729	-1.0948	-0.3868	2.2081

k	21	22	23	24	25
$Z_k^{(1)}$	-0.0982	-1.0459	0.5069	-0.2890	0.8461
$Z_k^{(2)}$	-1.5701	-0.4373	-1.0468	0.7973	-3.0566

(b) Values of $Z_k^{(1)}$, $Z_k^{(2)}$ $(k \leq 250, |Z_k| > 1.96)$

k	40	50	104	116	123	158	185	243	247	
$Z_k^{(1)}$	-2.2216	2.7301	-2.1716	2.0180	-2.4662	-1.9744	2.0807	2.0927	2.3481	
k	8	20	25	64	67	74	99	120	132	146
$Z_k^{(2)}$	-2.0082	2.2081	-3.0566	2.0581	2.0773	2.2483	-1.9877	-2.0903	1.9756	2.0105
k	197	226	241	247	249					
$Z_k^{(2)}$	-2.6246	2.3229	-2.4631	2.1081	2.0193					

APPENDIX 3.PROGRAM LISTINGS OF MUL3.FOR,MUL1.FOR.

```
C
C
C              PROGRAM: MUL3.FOR
C
C
C
      SUBROUTINE BVAR1(X1,X2)
      R1=RAN(X)
      R2=RAN(X)
      X1=SQRT(2*R2*(1-(.75*R1)))
      X2=(1.0/SQRT(1.0-(.75*R1))-1.0)*X1
      RETURN
      END
```

```
C
CC
C               PROGRAM: MUL1.FOR
CC
C
      SUBROUTINE BVAR2(X1,X2)
10    R1=RAN(X)
      R2=RAN(X)
      R3=RAN(X)
      X1=1.4142136*R1
      X2=.7071068*R2
      IF(X1+X2.GT.1.4142136)GOTO 20
      IF(X2-X1.LE.0) GOTO 30
      X1=.7071068-X1
      X2=.7071068-X2
      GOTO 30
20    X1=2.1213203-X1
      X2=.7071068-X2
30    IF(R3.GT.X1/(X1+X2)) GOTO 10
      RETURN
      END
```

APPENDIX 4.PROGRAM LISTINGS OF WALDCH.FOR,INGAUS.FOR, INLOG.FOR,INLOG1.FOR.

```
C
CC
C              PROGRAM:WALDCH.FOR
CC
C
      SUBROUTINE WALDCH(JC,IP,FI,X)
C     ROUTINE GENERATES STANDARDISED WALD VARIATES X
C     HAVING PDF.  SQRT(FI/2PI)*EXP(FI)*(X**-1.5)
C     *EXP(-.5FI(X+1./X)). JC SET TO 1 ON FIRST CALL,
C     TO AN INTEGER  > 1 ON SUBSEQUENT CALLS. VARIATES
C     GENERATED IN PAIRS, IP=0 INDICATING
C     NEW PAIR REQUIRED. REFERENCE : MICHAEL,SCHUCANY AND
C     HAAS. AMERICAN STATISTICIAN,MAY 76,30,2,88-90.
C     CHI-SQUARED 1 = FI((X-1)**2)/X.
      COMMON/PRE/A,B,C,Y
C     SETUP
      IF(JC.GT.1) GOTO 10
      A=1.0/FI
      B=.5*A
      C=B*B
C     NEW PAIR TO BE GENERATED ?
10    IF(IP.EQ.0) GOTO 20
      X=Y
      IP=0
      RETURN
C     GENERATES PAIR OF CHI-SQUARED VARIATES CH1,CH2
20    R1=RAN(Z)
      R2=RAN(Z)
      R3=RAN(Z)
      R4=RAN(Z)
      V=-2*ALOG(R1)
      C2=(COS(6.2831853*R2))**2
      CH1=V*C2
      CH2=V*(1.0-C2)
C     GENERATES SMALLER ROOTS X AND Y FOR THESE 2
C     VARIATES.
      X=1.0+CH1*B-SQRT(C*(CH1**2)+A*CH1)
      Y=1.0+CH2*B-SQRT(C*(CH2**2)+A*CH2)
C     BERNOULLI TRIALS TO RETURN SMALLER/LARGER ROOTS
      IF(R3.LT.1.0/(1.0+X)) GOTO 30
      X=1.0/X
30    IF(R4.LT.1.0/(1.0+Y)) GOTO 40
      Y=1.0/Y
C     INDICATOR SET TO 1 TO SHOW 1 VARIATE REMAINS
40    IP=1
      RETURN
      END
```

```
C
C
C          PROGRAM: INGAUS.FOR
C
C
C
C     SUBROUTINE GENERATES STANDARDISED WALD VARIATES X
C     WITH PDF(X)=SQRT(FI/2*PI).EXP(FI).(X**-1.5)
C     .EXP(-.5*FI(X+1/X)) , USING RATIO OF UNIFORMS
C     METHOD.
C     FIRST CALL OF ROUTINE SHOULD BE WITH A=0,B=0,X=0,
C     J=1.
C     SUBSEQUENT CALLS REQUIRE ONLY THAT J BE GREATER
C     THAN 1.
C     A AND B ARE SET-UP VALUES STORED BETWEEN CALLS.
C     AMX AND AMY ARE MODES OF THE WALD AND RECIPROCAL
C     DISTRIBUTIONS RESPECTIVELY.
C
      SUBROUTINE WALD(FI,A,B,J,X)
      IF(J.GT.1) GOTO 10
      AMX=SQRT(1.+9./(4.*FI*FI))-3./(FI+FI)
      AMY=SQRT(1.+1./(4.*FI*FI))-1./(FI+FI)
      A=(AMX**3./AMY)**.25
      A=A*EXP(FI*(AMX+(1./AMX)-AMY-(1./AMY))/4.)
      B=.75*ALOG(AMX*AMY)
      B=B+(.25*FI*(3.*AMY+(3./AMY)+AMX+(1./AMX)))
10    R1=RAN(Z)
      R2=RAN(Z)
      X=A*R2/R1
      W=R1*(R2**3)
      Y=ALOG(W)+FI*(X+1./X)
      IF(Y.GT.B) GOTO 10
      RETURN
      END
```

```
C
C
C              PROGRAM: INLOG.FOR
C
C
C
C       SUBROUTINE GENERATES STANDARDISED WALD VARIATES X
C       WITH PDF(X)=SQRT(FI/2*PI).EXP(FI).(X**-1.5)
C       .EXP(-.5*FI(X+1/X)) , USING RATIO OF UNIFORMS
C       METHOD.
C       FIRST CALL OF ROUTINE SHOULD BE WITH A=0,B=0,X=0,
C       J=1.
C       SUBSEQUENT CALLS REQUIRE ONLY THAT J BE GREATER
C       THAN 1.
C       A AND B ARE SET-UP VALUES STORED BETWEEN CALLS.
C       AMX AND AMY ARE MODES OF THE WALD AND RECIPROCAL
CC      DISTRIBUTIONS RESPECTIVELY.
C
        SUBROUTINE WALD(FI,A,B,J,X)
        IF(J.GT.1) GOTO 10
        AMX=SQRT(1.+9./(4.*FI*FI))-3./(FI+FI)
        AMY=SQRT(1.+1./(4.*FI*FI))-1./(FI+FI)
        A=(AMX**3./AMY)**.25
        A=A*EXP(FI*(AMX+(1./AMX)-AMY-(1./AMY))/4.)
        B=.75*ALOG(AMX*AMY)
        B=B+(.25*FI*(3.*AMY+(3./AMY)+AMX+(1./AMX)))
10      R1=RAN(Z)
        R2=RAN(Z)
        X=A*R2/R1
        W=R1*(R2**3)
        T=W-1
        V=B-FI*(X+1.0/X)
        IF(T.LE.V)RETURN
        IF(ALOG(W).LE.V)RETURN
        GOTO 10
        END
```

```
C
CC
C              PROGRAM: INLOG1.FOR
CC
CC
CC
C       SUBROUTINE GENERATES STANDARDISED WALD VARIATES X
C       WITH PDF(X)=SQRT(FI/2*PI).EXP(FI).(X**-1.5)
C       .EXP(-.5*FI(X+1/X)) , USING RATIO OF UNIFORMS
C       METHOD.
C       FIRST CALL OF ROUTINE SHOULD BE WITH A=0,B=0,X=0,
C       J=1.
C       SUBSEQUENT CALLS REQUIRE ONLY THAT J BE GREATER
C       THAN 1.
C       A AND B ARE SET-UP VALUES STORED BETWEEN CALLS.
C       AMX AND AMY ARE MODES OF THE WALD AND RECIPROCAL
CC      DISTRIBUTIONS RESPECTIVELY.
C
        SUBROUTINE WALD(FI,A,B,J,X)
        IF(J.GT.1) GOTO 10
        AMX=SQRT(1.+9./(4.*FI*FI))-3./(FI+FI)
        AMY=SQRT(1.+1./(4.*FI*FI))-1./(FI+FI)
        A=(AMX**3./AMY)**.25
        A=A*EXP(FI*(AMX+(1./AMX)-AMY-(1./AMY))/4.)
        B=.75*ALOG(AMX*AMY)
        B=B+(.25*FI*(3.*AMY+(3./AMY)+AMX+(1./AMX)))
10      R1=RAN(Z)
        R2=RAN(Z)
        X=A*R2/R1
        W=R1*(R2**3)
        T=2*(W-1)/(W+1)
        V=B-FI*(X+1.0/X)
        IF(T.LE.V)RETURN
        IF(ALOG(W).LE.V)RETURN
        GOTO 10
        END
```

APPENDIX 5.PROGRAM LISTINGS OF ALIAS.FOR,MVALUE.FOR, POLOG.FOR,NBINOM.FOR.

```
C
C
C               PROGRAM: ALIAS.FOR
C
C
C       *******
C       ROUTINE GENERATES POISSON VARIATES IX,MEAN EJ.JC
C       MUST BE SET TO 1 ON FIRST CALL FOR FIXED EJ,AND
C       TO AN INTEGER > 1 ON SUBSEQUENT CALLS.
C       PROGRAM GENERATES VALUES IN RANGE Ø,1,     ..,M-1.
C       USING ALIAS TABLES F(.),IA(.),WHERE
C       M=IFIX(EJ+2.5*SQRT(EJ)+1).
C       PROGRAM GENERATES VALUES IN RANGE M,M+1,M+2,....
C       USING A GEOMETRIC TAIL PROCEDURE.
C       E(J)=PROB(IX=J-1]IX<=M-1) ; J=1,2,.... ,M.
C       W=PROB(IX<=M-1).
C       N=M, AND INITIALLY VARIATES IN THE RANGE
C       1,2,.....,N ARE GENERATED WITH SUBTRACTION OF 1
C       BEFORE RETURNING VARIATE.
C       *******
        SUBROUTINE POISAL(EJ,JC,IX)
        EXTERNAL FACTOR
        COMMON/LOCAL/ E(1100),F(1100),IA(1100),N,W
        COMMON/LOCAL1/YP,E1
        IF(JC.GT.1)GOTO 24
C       *******
C       *******
        DATA E,F/2200*0.0/,IA/1100*0/
        E1=EXP(1.)
        PI=3.1415926536
        YP=EJ
        M=1+IFIX(EJ+2.5*SQRT(EJ))
        N=M
        IF(N.GT.1100) GOTO 5000
        MU=IFIX(EJ)
        MS=IFIX(EJ/E1)
        PM=1.
        CALL REMULT(0,MS,MU,PM,FACTOR)
        E(1+MU)=PM*EXP(-(EJ-MU))
        W=E(1+MU)
        IF(MU.LE.0) GOTO 27
        DO 25 J=MU,1,-1
        E(J)=E(J+1)*FLOAT(J)/EJ
        IF(E(J).GT.1.E-20) GOTO 26
        E(J)=0.0
26      W=W+E(J)
25      CONTINUE
27      IF(MU+1.GE.N) GOTO 29
        DO 130 J=MU+2,N
        E(J)=E(J-1)*EJ/FLOAT(J-1)
```

Text cut off in original

```
      IF(E(I)-XI)10,30,40
10    IF(K.EQ.0) GOTO 20
      IA(K)=I
      K=I
      GOTO 60
20    KB=I
      K=I
      GOTO 60
30    F(I)=1.0
      GOTO 60
40    IF(J.EQ.0) GOTO 50
      IA(J)=I
      J=I
      GOTO 60
50    JB=I
      J=I
60    CONTINUE
70    IF(JB.EQ.0)RETURN
      J=JB
      JB=IA(J)
      F(J)=XN*E(J)
      IF(KT.EQ.K) GOTO 90
80    IF(KB.EQ.0)RETURN
      K=KB
      KB=IA(K)
      F(K)=XN*E(K)
90    F(J)=F(J)+F(K)-1
      IA(K)=J
      IF(F(J)-1.0)100,80,80
100   K=J
      KT=K
      GOTO 70
      RETURN
      END
C     *******
C     *******
      SUBROUTINE REMULT(MLLIM,MS,MRLIM,PM,FACTOR)
      IF(MS.GE.MLLIM.AND.MS.LE.MRLIM)GOTO 20
      MS=MLLIM
20    CONTINUE
      ML=MS-1
      MR=MS+1
      PS=FACTOR(MS)
      PM=PS
30    CONTINUE
      IF(ML.GE.MLLIM)GOTO 40
      IF(MR.LE.MRLIM) GOTO 70
      RETURN
C
C
40    CONTINUE
      IF(MR.LE.MRLIM) GOTO 60
50    CONTINUE
      PM=PM*FACTOR(ML)
      ML=ML-1
```

```
      GOTO 30
60    CONTINUE
      IF(PM.GT.PS) GOTO 70
      GOTO 50
70    CONTINUE
      PM=PM*FACTOR(MR)
      MR=MR+1
      GOTO 30
      END
C     *******
C     *******
      FUNCTION FACTOR(JT)
      COMMON/LOCAL1/YP,E1
      FACTOR=1.
      IF(JT.EQ.0) RETURN
      FACTOR=YP/(FLOAT(JT)*E1)
      END
```

```
C
C
C          PROGRAM: MVALUE.FOR
C
C
C
C     PROGRAM ESTIMATES M-VALUE FOR POISSON ENVELOPE
C     REJECTION METHOD (TRUNCATED LOGISTIC).USES
C     STIRLING LOWER BOUND TO COMPUTE FACTORIAL.
      WRITE(5,10)
10    FORMAT(' ENTER MEAN,NO. OF TRIALS')
      READ(5,*)E,NC
      T=1.E+20
      BET=3.1415926536/SQRT(3.*E)
      ALP=BET*E
      GA=EXP(ALP+.5*BET)
      F=.5*ALOG(6.2831853072*E)
      DO 50 J=1,NC
      R1=RAN(X)
      WP=ALOG((1.+GA*R1)/(1.-R1))/BET
      NREJ=IFIX(WP)
      W=ALOG((1.-R1)*(1.+GA*R1))
      IF(NREJ.EQ.0) GOTO 30
      Z=W+(NREJ+.5)*ALOG((NREJ/E))-NREJ
      S1=Z+F
      GOTO 20
30    S1=W
20    IF(S1.GE.T)GOTO 40
      T=S1
      IXMAX=NREJ
40    CONTINUE
50    CONTINUE
      RM=(1.+GA)*EXP(-(E+T))/BET
      WRITE(5,60)IXMAX,RM
60    FORMAT(' ENV. MAXIMISED AT ',I4,'M-VALUE ESTIMATE=
     1',F10.6)
      STOP
      END
```

```
C
C
C           PROGRAM: POLOG.FOR
C
C
C
C       FUNCTION GENERATES POISSON VARIATES MEAN E
C       (IN RANGE 0-1000) . JC MUST BE SET TO 0 ON FIRST
C       CALL , TO AN INTEGER > 1 ON SUBSEQUENT CALLS.
C       ENVELOPE REJECTION METHOD WITH LOGISTIC TARGET
C       DISTRIBUTION TRUNCATED AT X=-0.5.
C       USES LOWER / UPPER STIRLING BOUNDS TO
C       PERFORM ACCEPTANCE PRE TESTS.
C       *******
        FUNCTION NREJ(E,JC,RM)
        COMMON/LOC/BET,GA,DEL,F
        IF(JC.GT.1) GOTO 10
        BET=3.1415927/SQRT(3.*E)
        ALP=BET*E
        GA=EXP(ALP+.5*BET)
        DEL=-E+ALOG((1.+GA)*.2199484/RM)
        F=.5*ALOG(6.2831853*E)
10      R1=RAN(X)
        R2=RAN(X)
        V=0
        WP=ALOG((1.+GA*R1)/(1.-R1))/BET
        NREJ=IFIX(WP)
        W=ALOG(R2*(1.-R1)*(1.+GA*R1))
        IF(NREJ.EQ.0) GOTO 30
        Z=W+(NREJ+.5)*ALOG((NREJ/E))-NREJ
        IF(Z+(.0833333333/FLOAT(NREJ)).LE.DEL)RETURN
        IF(Z.GT.DEL) GOTO 10
        DO 20 I=1,NREJ
20      V=V+ALOG(I/E)
30      IF(W+V.LE.DEL+F)RETURN
        GOTO 10
        END
```

```
C
C
C            PROGRAM: NBINOM.FOR
C
C
C       *******
C       ROUTINE GENERATES NEGATIVE BINOMIAL VARIATES IX,
C       WITH PARAMETERS K AND P.JC MUST BE SET TO 1 ON
C       FIRST CALL,TO INTEGER > 1 ON SUBSEQUENT CALLS.
C       PROGRAM GENERATES VALUES IN RANGE 0,1,2...,N-1
C       USING ALIAS TABLES F(.),IA(.), WHERE
C       N=1+IFIX((K*P+2.5*SQRT(K*P))/Q).
C       PROGRAM GENERATES VALUES IN RANGE  N,N+1,.......
C       USING A GEOMETRIC TAIL PROCEDURE.
C       E(J)=PROB(IX=J-1[IX<=N-1)   ;  J=1,2,.....,N
C       W=PROB(IX<=N-1)
C       NOTE THAT FOR ALIAS GENERATION VARIATES ARE
C       INITIALLY GENERATED IN THE RANGE  1,2,.....,N
C       WITH SUBTRACTION OF 1 BEFORE RETURNING VARIATE.
C        GENERATOR SUITABLE PROVIDING MEAN + 2.5
C       STD. DEVS. DOES NOT EXCEED 11C0.
C       *******
        SUBROUTINE NBINOM(K,P,IX,JC)
        EXTERNAL FACTOR
        COMMON/LOCAL/E(1100),F(1100),IA(1100),N,W
        COMMON/PROD/MODE,KSS,PSS
        IF(JC.GT.1)GOTO 24
C       *******
C       *******
        Q=1.-P
        DATA E,F/2200*0.0/,IA/1100*0/,X/0.0/
        N=1+IFIX((K*P+2.5*SQRT(K*P))/Q)
        IF(N.GT.1100) GOTO 5000
        MODE=IFIX((K-1)*P/Q)
        PSS=Q*(P**(FLOAT(MODE)/FLOAT(K)))
        KSS=K
        MU=K-1
        MS=1+IFIX((MODE+K)/(1.+1./PSS))
        IF(MS.LE.K-1) GOTO 900
        MS=K-1
  900   CONTINUE
        PM=PSS
        CALL REMULT(0,MS,MU,PM,FACTOR)
        E(1+MODE)=PM
C       *******
C       *******
        W=E(1+MODE)
        IF(MODE.LE.0) GOTO 27
        DO 25 J=MODE,1,-1
        E(J)=E(J+1)*FLOAT(J)/(P*FLOAT(J+K-1))
        IF(E(J).GT.1.E-20) GOTO 26
        E(J)=0.0
   26   W=W+E(J)
   25   CONTINUE
   27   IF(MODE+1.GE.N) GOTO 29
```

```
      DO 130 J=MODE+2,N
      E(J)=E(J-1)*P*FLOAT(J+K-2)/FLOAT(J-1)
      IF(E(J).GT.1.E-20) GOTO 131
      E(J)=0.0
131   W=W+E(J)
130   CONTINUE
29    CONTINUE
      DO 32 J=1,N
32    E(J)=E(J)/W
C     *******
C     *******
      CALL PTAB
C     *******
C     *******
24    IF(N.GT.1100) GOTO 5000
      R1=RAN(X)
      IF(R1.GT.W) GOTO 100
      R1=R1/W
      S=R1*N
      IS=1+IFIX(S)
      IF(S+1-IS.GT.F(IS)) GOTO 80
      IX=IS-1
      RETURN
80    IX=IA(IS)-1
      RETURN
C     *******
C     *******
100   R1=RAN(X)
      R2=RAN(X)
      Z=-ALOG(R1)
      IX=N+IFIX(Z/ALOG(N/(P*FLOAT(N+K-1))))
      IF(IX.EQ.N) RETURN
C     *******
C     *******
      CP=FLOAT(N)/FLOAT(N+K-1)
      JP=1
      Y=1.0
31    Y=Y*(IX+K-JP)*CP/FLOAT(IX+1-JP)
      IF(Y.LT.R2) GOTO 100
      IF(JP.EQ.IX-N) RETURN
      JP=JP+1

      GOTO 31
C     *******
C     *******
5000  WRITE(5,5010)
5010  FORMAT(' MEAN + 2.5 STD. DEVS. EXCEEDS 1100')
      RETURN
      END
      SUBROUTINE PTAB
      COMMON/LOCAL/ E(1100),F(1100),IA(1100),N,W
      XN=FLOAT(N)
      XI=1.0/XN
      K=0
      J=0
```

```
      KT=0
      DO 60 I=1,N
      IA(I)=0
      IF(E(I)-XI)10,30,40
10    IF(K.EQ.0) GOTO 20
      IA(K)=I
      K=I
      GOTO 60
20    KB=I
      K=I
      GOTO 60
30    F(I)=1.0
      GOTO 60
40    IF(J.EQ.0) GOTO 50
      IA(J)=I
      J=I
      GOTO 60
50    JB=I
      J=I
60    CONTINUE
70    IF(JB.EQ.0)RETURN
      J=JB
      JB=IA(J)
      F(J)=XN*E(J)
      IF(KT.EQ.K) GOTO 90
80    IF(KB.EQ.0)RETURN
      K=KB
      KB=IA(K)
      F(K)=XN*E(K)
90    F(J)=F(J)+F(K)-1
      IA(K)=J
      IF(F(J)-1.0)100,80,80
100   K=J
      KT=K
      GOTO 70
      RETURN
      END
C     *******
C     *******
      SUBROUTINE REMULT(MLLIM,MS,MRLIM,PM,FACTOR)
      IF(MS.GE.MLLIM.AND.MS.LE.MRLIM)GOTO 20
      MS=MLLIM
20    CONTINUE
      ML=MS-1
      MR=MS+1
      PS=FACTOR(MS)
      PM=PS
30    CONTINUE
      IF(ML.GE.MLLIM)GOTO 40
      IF(MR.LE.MRLIM) GOTO 70
      RETURN
C
C
40    CONTINUE
      IF(MR.LE.MRLIM) GOTO 60
```

```
      50    CONTINUE
            PM=PM*FACTOR(ML)
            ML=ML-1
            GOTO 30
      60    CONTINUE
            IF(PM.GT.PS) GOTO 70
            GOTO 50
      70    CONTINUE
            PM=PM*FACTOR(MR)
            MR=MR+1
            GOTO 30
            END
      C     *******
      C     *******
            FUNCTION FACTOR(JT)
            COMMON/PROD/MODE,KSS,PSS
            FACTOR=PSS
            IF(JT.EQ.0) RETURN
            FACTOR=PSS*FLOAT(MODE+KSS-JT)/FLOAT(JT)
            END
```

APPENDIX 6:PROGRAM LISTINGS OF
VMISES.FOR,UNIVON.FOR,
BFISH.FOR,BFISHC.FOR.

```
C
CC
C              PROGRAM :VMISES.FOR
CC
CC
C       ROUTINE GENERATES VON MISES VARIATES HAVING
C       P.D.F.   CONSTANT*EXP(K*COS(H))  (-PI < H < PI),
C       USING COMPARISON OF RANDOM NUMBERS. IF K=0 , PUT
C       KD=0,SJ=0.0 ;IF K>0 AND 2K IS INTEGER PUT KD=2K
C       ,SJ=0.0 ;IF K>0 AND 2K IS NON INTEGER PUT KD=0,
C       SJ=K.SET JC=1 ON FIRST CALL , TO AN INTEGER >1 ON
C       SUBSEQUENT CALLS.
C       NK=NO. OF INTERVALS. BOUNDARY VALUES AND C.D.F'S
C       STORED IN TH(.) AND RP(.).
C       ******
        SUBROUTINE VON(KD,SJ,H,JC)
        COMMON/VONMIS/TH(10),RP(10),NK,SK
        COMMON/INDIC/JIND
        IF(JC.GT.1) GOTO 24
        IF(KD.EQ.0) GOTO 2
        NK=KD
        SK=.5*KD
        GOTO 4
2       SK=SJ
        NK=1+IFIX(SK+SK)
4       IF(NK.GT.10) NK=10
        IF(NK.EQ.1) GOTO 47
        DO 55 I=1,NK-1
55      TH(I)=ACOS(1.-I/SK)
47      TH(NK)=3.141592654
        CALL INTEG
C       ******
C       SELECTS INTERVAL
C       ******
24      THT=0.0
        RHT=0.0
        R1=RAN(X)
        JIND=JIND+1
        J=1
50      IF(RP(J).GE.R1) GOTO 60
        J=J+1
        GOTO 50
C       ******
C       GENERATES VARIATE UNIFORMLY WITHIN INTERVAL
C       ******
60      IF(J.EQ.1) GOTO 62
        THT=TH(J-1)
        RHT=RP(J-1)
62      R1=(R1-RHT)/(RP(J)-RHT)
63      H=THT+R1*(TH(J)-THT)
```

```
C       ******
C       PERFORMS COMPARISONS AND REJECTS IF EVEN
C       ******
        D=SK-J+1-SK*COS(H)
        IND=1
70      R2=RAN(X)
        JIND=JIND+1
        IF(R2.GT.D) GOTO 80
        D=R2
        IND=IND+1
        GOTO 70
80      IF(MOD(IND,2).EQ.1) GOTO 90
        R1=RAN(X)
        JIND=JIND+1
        GOTO 63
C       ******
C       ASSIGNS A RANDOM SIGN
C       ******
90      IF(R2+R2-D.GT.1.) H=-H
        RETURN
        END
C       ******
C       CALCULATES C.D.F. , RP(.) AT NK BOUNDARY POINTS
C       ******
        SUBROUTINE INTEG
        COMMON/VONMIS/TH(10),RP(10),NK,SK
        IF(NK.GT.1) GOTO 70
        RP(1)=1.0
        RETURN
70      CONTINUE
        DO 10 I=1,NK
        IF(I.NE.1) GOTO 15
        X0=0.0
        GOTO 20
15      X0=TH(I-1)
20      A=0.0
        CALL SIMP(X0,TH(I),32,SK,A)
        IF(I.NE.1) GOTO 25
        RP(I)=A
        GOTO 10
25      RP(I)=RP(I-1)+A
10      CONTINUE
        DO 30 I=1,NK
30      RP(I)=RP(I)/RP(NK)
        RETURN
        END
C       ******
C       NUMERICAL INTEGRATION BETWEEN X0 AND XF
C       USING 2NI STRIPS.
C       ******
        SUBROUTINE SIMP(X0,XF,NI,SK,A)
        H=(XF-X0)/(FLOAT(NI+NI))
        HX=X0
        HY=EXP(SK*COS(X0))
        DO 10 I=1,NI
```

```
      X0I=HX
      X1I=X0I+H
      X2I=X1I+H
      Y0I=HY
      Y1I=EXP(SK*COS(X1I))
      Y2I=EXP(SK*COS(X2I))
      HX=X2I
      HY=Y2I
10    A=A+H*(Y0I+4*Y1I+Y2I)/3.
      RETURN
      END
```

```
C
C
C          PROGRAM:UNIVON.FOR
C
C
C
C     ROUTINE GENERATES VON MISES VARIATES H , HAVING
C     P.D.F.   CONSTANT*EXP(K*COS(H))  (-PI < H < PI),
C     USING ENVELOPE REJECTION ON MIXTURE. IF K=0 , PUT
C     KD=0,SJ=0.0 ;IF K>0 AND 2K IS INTEGER PUT KD=2K
C     ,SJ=0.0 ;IF K>0 AND 2K IS NON INTEGER PUT KD=0,
C     SJ=K.SET JC=1 ON FIRST CALL , TO AN INTEGER >1 ON
C     SUBSEQUENT CALLS.
C     NK=NO. OF INTERVALS. BOUNDARY VALUES AND C.D.F'S
C     STORED IN TH(.) AND RP(.).
C     ******
      SUBROUTINE VON(KD,SJ,H,JC)
      COMMON/VONMIS/TH(10),RP(10),NK,SK
      COMMON/INDIC/JIND
      IF(JC.GT.1) GOTO 24
      IF(KD.EQ.0) GOTO 2
      NK=KD
      SK=.5*KD
      GOTO 4
2     SK=SJ
      NK=1+IFIX(SK+SK)
4     IF(NK.GT.10) NK=10
      IF(NK.EQ.1) GOTO 47
      DO 55 I=1,NK-1
55    TH(I)=ACOS(1.-I/SK)
47    TH(NK)=3.141592654
      CALL INTEG
C     ******
C     SELECTS INTERVAL
C     ******
24    THT=0.0
      RHT=0.0
      R1=RAN(X)
      JIND=JIND+1
      J=1
50    IF(RP(J).GE.R1) GOTO 60
      J=J+1
      GOTO 50
C     ******
C     GENERATES VARIATE UNIFORMLY WITHIN INTERVAL
C     ******
60    IF(J.EQ.1) GOTO 62
      THT=TH(J-1)
      RHT=RP(J-1)
62    R1=(R1-RHT)/(RP(J)-RHT)
63    H=THT+R1*(TH(J)-THT)
C     ******
C     ENVELOPE REJECTION STEP
C     ******
      D=SK-J+1-SK*COS(H)
      IND=1
```

```
      IF(E(J).GT.1.E-20) GOTO 131
      E(J)=0.0
131   W=W+E(J)
130   CONTINUE
29    CONTINUE
      DO 32 J=1,N
32    E(J)=E(J)/W
C     *******
C     *******
      CALL PTAB
C     *******
C     *******
24    IF(N.GT.1100) GOTO 5000
      R1=RAN(X)
      IF(R1.GT.W)GOTO 100
      R1=R1/W
      S=R1*N
      IS=1+IFIX(S)
      IF(S+1-IS.GT.F(IS)) GOTO 80
      IX=IS-1
      RETURN
80    IX=IA(IS)-1
      RETURN
C     *******
C     *******
100   R1=RAN(X)
      R2=RAN(X)
      Z=-ALOG(R1)
      V=FLOAT(N)/EJ
      IX=N+IFIX(Z/ALOG(V))
      IF(IX.EQ.N)RETURN
C     *******
C     *******
      JP=0
      Y=1.
31    Y=Y*N/FLOAT(IX-JP)
      IF (Y.LT.R2) GOTO 100
      IF(JP.EQ.IX-N-1)RETURN
      JP=JP+1
      GOTO 31
5000  WRITE(5,5010)
5010  FORMAT(' MEAN + 2.5 STD. DEVS. EXCEEDS 1100 ')
      RETURN
      END
C     *******
C     *******
      SUBROUTINE PTAB
      COMMON/LOCAL/ E(1100),F(1100),IA(1100),N,W
      XN=FLOAT(N)
      XI=1.0/XN
      K=0
      J=0
      KT=0
      DO 60 I=1,N
      IA(I)=0
```

```
 70     R2=RAN(X)
        JIND=JIND+1
        Z=-ALOG(R2)
        IF(Z.GT.D) GOTO 90
        R1=RAN(X)
        JIND=JIND+1
        GOTO 63
C       ******
C       ASSIGNS A RANDOM SIGN
C       ******
 90     R3=RAN(X)
        JIND=JIND+1
        IF(R3.GT.0.5)H=-H
        RETURN
        END
C       ******
C       CALCULATES C.D.F. , RP(.) AT NK BOUNDARY POINTS
C       ******
        SUBROUTINE INTEG
        COMMON/VONMIS/TH(10),RP(10),NK,SK
        IF(NK.GT.1)GOTO 70
        RP(1)=1.0
        RETURN
 70     CONTINUE
        DO 10 I=1,NK
        IF(I.NE.1) GOTO 15
        X0=0.0
        GOTO 20
 15     X0=TH(I-1)
 20     A=0.0
        CALL SIMP(X0,TH(I),32,SK,A)
        IF(I.NE.1) GOTO 25
        RP(I)=A
        GOTO 10
 25     RP(I)=RP(I-1)+A
 10     CONTINUE
        DO 30 I=1,NK
 30     RP(I)=RP(I)/RP(NK)
        RETURN
        END
C       ******
C       NUMERICAL INTEGRATION BETWEEN X0 AND XF
C       USING 2NI STRIPS.
C       ******
        SUBROUTINE SIMP(X0,XF,NI,SK,A)
        H=(XF-X0)/(FLOAT(NI+NI))
        HX=X0
        HY=EXP(SK*COS(X0))
        DO 10 I=1,NI
        X0I=HX
        X1I=X0I+H
        X2I=X1I+H
        Y0I=HY
        Y1I=EXP(SK*COS(X1I))
        Y2I=EXP(SK*COS(X2I))
```

```
      HX=X2I
      HY=Y2I
10    A=A+H*(Y0I+4*Y1I+Y2I)/3.
      RETURN
      END
```

```
C
CC
CC              PROGRAM:BFISH.FOR
CC
CC
CC        ******
CC        ROUTINE GENERATES VON MISES VARIATES H WITH
CC        PARAMETER SJ.USES BEST AND FISHER ENVELOPE
CC        REJECTION METHOD WITH A WRAPPED CAUCHY
CC        TARGET DISTRIBUTION.VERSION EMPLOYS
CC        PRETEST TO AVOID LOG EVALUATIONS.
C         COSINE IS EVALUATED CONVENTIONALLY.
          SUBROUTINE BFISH(SJ,H,JC)
          COMMON/INDIC/JIND
          SK=SJ
          IF(JC.GT.1)GOTO 24
          T=1.+SQRT(1.+4*SK*SK)
          RHO=(T-SQRT(T+T))/(SK+SK)
          R=(1.+RHO*RHO)/(RHO+RHO)
 24       U1=RAN(X)
          JIND=JIND+1
          Z=COS(3.141592654*U1)
          F=(1.+R*Z)/(R+Z)
          IF(F.GT.1.)F=1.
          IF(F.LT.-1.)F=-1.
          C=SK*(R-F)
          U2=RAN(X)
          JIND=JIND+1
          IF(C*(2.-C)-U2.GT.0)GOTO 4
          IF(ALOG(C/U2)+1.-C.LT.0)GOTO 24
 4        H=ACOS(F)
          U3=RAN(X)-.5
          JIND=JIND+1
          IF(U3.LT.0)H=-H
          RETURN
          END
```

```
C
C
C           PROGRAM:BFISHC.FOR
C
C
C
C       ROUTINE GENERATES VON MISES VARIATES H WITH
C       PARAMETER SJ.USES BEST AND FISHER ENVELOPE
C       REJECTION METHOD,WITH A WRAPPED CAUCHY TARGET
C       DISTRIBUTION.VERSION EMPLOYS PRETEST TO AVOID
C       LOG EVALUATIONS.COSINE IS EVALUATED VIA
C       POLAR METHOD.
C       ******
        SUBROUTINE BFISHC(SJ,H,JC)
        COMMON/INDIC/JIND
        SK=SJ
        IF(JC.GT.1)GOTO 24
        T=1.+SQRT(1.+4*SK*SK)
        RHO=(T-SQRT(T+T))/(SK+SK)
        R=(1.+RHO*RHO)/(RHO+RHO)
24      V=RAN(X)-.5
        W=RAN(X)-.5
        JIND=JIND+2
        D=V*V
        E=W*W
        SUM=4*(D+E)
        IF(SUM.GT.1.)GOTO 24
        TR=D/E
        Z=(1.-TR)/(1.+TR)
        F=(1.+R*Z)/(R+Z)
        IF(F.GT.1.)F=1.
        IF(F.LT.-1.)F=-1.
        C=SK*(R-F)
        U2=SUM
        IF(C*(2.-C)-U2.GT.0)GOTO 4
        IF(ALOG(C/U2)+1.-C.LT.0)GOTO 24
4       H=ACOS(F)
        U3=RAN(X)-.5
        JIND=JIND+1
        IF(U3.LT.0)H=-H
        RETURN
        END
```

APPENDIX 7.PROGRAM LISTINGS OF GATRUN.FOR,TRUNCN.FOR, TRUNCM.FOR,TRUNCS.FOR.

```
C
C
C              PROGRAM: GATRUN.FOR
C
C
C
C       ROUTINE GENERATES GAMMA VARIATES Y,SHAPE
C       PARAMATER ALPHA(>1) ,TRUNCATED AT T(>0).USES
C       OPTIMAL EXPONENTIAL ENVELOPE WITH PARAMATER
C       U.JC SET TO 1 ON FIRST CALL , TO AN INTEGER
C       >1 ON SUBSEQUENT CALLS.
        SUBROUTINE TRUNG(JC,T,ALPHA,Y)
        COMMON/PARAM/C,U,A,D
        IF(JC.GT.1) GOTO 10
        C=ALPHA-1.0
        U=((T-ALPHA)+SQRT((T-ALPHA)**2+4*T))/(T+T)
        A=1.0-U
        D=-C*(ALOG(A/C)+1.0)
10      R1=RAN(X)
        R2=RAN(X)
        Y=T-ALOG(R1)/U
        W=A*Y-C*ALOG(Y)+D
        IF(2*(1.-R2)/(1.+R2).GT.W)RETURN
        IF(-ALOG(R2).GT.W)RETURN
        GOTO 10
        END
```

```
C
C
C          PROGRAM:TRUNCN.FOR
C
C
C
C       ROUTINE GENERATES NORMAL DEVIATES Y,
C       TRUNCATED AT A (>0) , USING AN OPTIMAL
C       EXPONENTIAL ENVELOPE WITH PARAMATER ALAM.
C       JC SET TO 1 ON FIRST CALL , TO AN INTEGER
C       > 1 ON SUBSEQUENT CALLS.
        SUBROUTINE TRUNCN(JC,A,Y)
        COMMON/PAR/ALAM
        IF(JC.GT.1) GOTO 10
        ALAM=(A+SQRT(4.+A*A))/2.
10      R1=RAN(X)
        R2=RAN(X)
        Y=A-ALOG(R1)/ALAM
        Z=Y-ALAM
        W=.5*Z*Z
        V=2*(1.-R2)/(1.+R2)
        IF(V.GT.W)RETURN
        IF(-ALOG(R2).GT.W)RETURN
        GOTO 10
        END
```

```
C
CC
C                    PROGRAM: TRUNCM.FOR
C
C
C
C       ROUTINE GENERATES NORMAL DEVIATES Y,
C       TRUNCATED AT A (>0) USING MARSAGLIA'S
C       TAIL GENERATION PROCEDURE. JC SET TO
C       1 ON FIRST CALL ,TO AN INTEGER >1 ON
C       SUBSEQUENT CALLS.
        SUBROUTINE TRUNCN(JC,A,Y)
        COMMON/SPAR/B
        IF(JC.GT.1) GOTO 10
        B=A*A
10      R1=RAN(X)
        R2=RAN(X)
        Y=SQRT(B-2.*ALOG(R1))
        IF(R2.LT.A/Y) RETURN
        GOTO 10
        END
```

```
C
C
C           PROGRAM:TRUNCS.FOR
C
C
C
C       ROUTINE GENERATES NORMAL DEVIATES Y,
C       TRUNCATED AT A (>Ø) , USING A NON OPTIMAL
C       EXPONENTIAL ENVELOPE WITH PARAMATER ALAM=A.
C       JC SET TO 1 ON FIRST CALL , TO AN INTEGER
C       > 1 ON SUBSEQUENT CALLS.
C       ONLY VALID WHEN A>=1.
        SUBROUTINE TRUNCN(JC,A,Y)
        COMMON/PAR/ALAM
        IF(JC.GT.1) GOTO 1Ø
        ALAM=A
1Ø      R1=RAN(X)
        R2=RAN(X)
        Y=A-ALOG(R1)/ALAM
        Z=Y-ALAM
        IF(R2.LT.(1.-Z*ALAM)/R1)RETURN
        W=.5*Z*Z
        IF(-ALOG(R2).GT.W)RETURN
        GOTO 1Ø
        END
```

REFERENCES

ABROMOWITZ, M and STEGUN, I A (1965). Handbook of Mathematical Functions. New York, Dover Publications.

AHRENS, J H and DIETER, U (1972). Computer Methods for Sampling from the Exponential and Normal Distributions. Commun. Ass. Comput. Mach., 15, 873-882.

AHRENS, J H and DIETER, U (1973). Extensions of Forsythe's Method for Random Sampling from the Normal Distribution. Math. Comput., 27, 927-937.

AHRENS, J H and DIETER, U (1974). Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions. Computing, 12, 223-246.

AHRENS, J H and DIETER, U (1980). Sampling from Binomial and Poisson Distributions: A Method with Bounded Computation Times. Computing, 25, 193-208.

ATKINSON, A C (1977). An Easily Programmed Algorithm for Generating Gamma Random Variables. J. R. Statist. Soc. A, 140, 232-234.

ATKINSON, A C (1979a). A Family of Switching Algorithms for the Computer Generation of Beta Random Variables. Biometrika, 66, 141-145.

ATKINSON, A C (1979b). The Computer Generation of Poisson Random Variables. Appl. Statist., 28, 29-35.

ATKINSON, A C (1979c). Recent Developments in the Computer Generation of Poisson Random Variables. Appl. Statist., 28, 260-263.

ATKINSON, A C and PEARCE, M C (1976). The Computer Generation of Beta, Gamma, and Normal Random Variables. J. R. Statist. Soc. A., 139, 431-461.

ATKINSON, A C and WHITTAKER, J (1976). A Switching Algorithm for the Generation of Beta Random Variables with at least One Parameter less than 1. J. R. Statist. Soc. A, 139, 462-467.

BENDELL, A and SAMSON, W B (1981). The use of Rank Order Distributions for the Estimation of the Probabilities of Rare Events. Third National Reliability Conference - Reliability 81.

BEST, D J and FISHER, N I (1979). Efficient Simulation of the Von Mises Distribution. Appl. Statist., 28, 152-157.

BOX, G E P and MULLER, G M (1958). A Note on the Generation of Random Normal Deviates. Ann. Math. Statist., 29, 610-611.

BRENT, R P (1974). A Gaussian pseudo random number generator. Commun. Ass. Comput. Mach., 17, 704-706.

CHEN, H-C and ASAU, Y (1974). On generating random variates from an empirical distribution. AIIE Trans., 6, 163-166.

CHENG, R C H (1977). The Generation of Gamma Variables with Non-integral Shape Parameter. Appl. Statist., 26, 71-75.

CHENG, R C H (1978). Generating Beta Variates with Non-integral Shape Parameters. Commun. Ass. Comput. Mach., 21, 317-322.

CHENG, R C H and FEAST, G M (1979). Some Simple Gamma Variate Generators. Appl. Statist., 28, 290-295.

CHENG, R C H and FEAST, G M (1980). Gamma Variate Generators with Increased Shape Parameter Range. Commun. Ass. Comput. Mach., 23, 389-394.

DAGPUNAR, J S (1977). Sampling of Variates from a Truncated Gamma Distribution. Department of Statistics and Operational Research, Brunel University, Report STR/17.

DAGPUNAR, J S (1978). Sampling of Variates from a Truncated Gamma Distribution. J. Statist. Comput. Simul., 8, 59-64.

DAGPUNAR, J S (1979). Alias Methods for the Computer Generation of Binomial, Poisson and Negative Binomial Variates. Department of Statistics and Operational Research, Brunel University, Report STR/35.

DEÁK, I (1980). Fast Procedures for Generating Stationary Normal Vectors. J. Statist. Comput. Simul., 10, 225-242.

DIETER, U and AHRENS, J H (1973). A Combinatorial Method for the Generation of Normally Distributed Random Numbers. Computing, 11, 137-146.

FISHMAN, G S (1973). Concepts and Methods of Discrete Event Simulation. New York: Wiley - Interscience.

FISHMAN, G S (1976a). Sampling from the Gamma distribution on a computer. Commun. Ass. Comput. Mach., 19, 407-409.

FISHMAN, G S (1976b). Sampling from the Poisson distribution on a computer. Computing, 17, 147-156.

FISHMAN, G S (1978). Principles of Discrete Event Simulation. New York: Wiley - Interscience.

FISHMAN, G S (1979). Sampling from the Binomial distribution on a computer. J. Amer. Statist. Ass., 74, 418-423.

FORSYTHE, G E (1972). Von Neumann's Comparison Method for Random Sampling from the Normal and Other Distributions. Math. Comput., 26, 817-826.

GREENWOOD, A J (1974). A fast generator for gamma distributed random variables. COMPSTAT 1974 (G Bruckman et al, eds) Vienna: Physica Verlag, pp 19-27.

GREGORY, C G (1979). A Simulation Model of a Telephone Answering Service. Unpublished MSc dissertation, Department of Statistics and Operational Research, Brunel University.

GUMBEL, E J (1960). Bivariate Exponential Distributions. J. Amer. Statist. Ass., 55, 698-707.

HOAGLIN, D (1976). Discussion of Atkinson, A C and Pearce, M C (1976). The computer generation of beta, gamma and normal random variables. J. R. Statist. Soc. A, 139, 431-461.

HSUAN, F C (1979). Generating Uniform Polygonal Random Pairs. Appl. Statist., 28, 170-172.

HULL, J C (1977). Dealing with dependence in risk simulation. Opl. Res. Q., 28, 201-213.

JOHNK, M D (1964). Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen. Metrika, 8, 5-15.

JOHNSON, N L and KOTZ, S (1969). Distributions in Statistics. Discrete Distributions. John Wiley.

JOHNSON, N L and KOTZ, S (1970a). Distributions in Statistics. Continuous univariate distributions - 1. John Wiley.

JOHNSON, N L and KOTZ, S (1970b). Distributions in Statistics. Continuous univariate distributions - 2. John Wiley.

JOHNSON, N L and KOTZ, S (1972). Distributions in Statistics. Continuous multivariate distributions. John Wiley.

KEMP, A W (1981). Efficient Generation of Logarithmically Distributed Pseudo-random Variables. Appl. Statist., 30, 249-253.

KEMP, C D and LOUKAS, S (1978). The Computer Generation of Bivariate Discrete Random Variables. J. R. Statist. Soc. A, 141, 513-519.

KENDALL, M G and STUART, A (1963). The Advanced Theory of Statistics, Volume 1, Distribution Theory. London : Griffin.

KINDERMAN, A J and MONAHAN, J F (1977). Computer Generation of Random Variables using the Ratio of Uniform Deviates. ACM Transactions on Mathematical Software, 3, 257-260.

KINDERMAN, A J and MONAHAN, J F (1980). New Methods for Generating Student's t and Gamma Variables. Computing, 25, 369-377.

KINDERMAN, A J, MONAHAN, J F and RAMAGE, J G (1977). Computer Methods for Sampling from Student's t Distribution. Math. Comput., 31, 1009-1018.

KINDERMAN, A J and RAMAGE, J G (1976). Computer Generation of Normal Random Variables. J. Amer. Statist. Ass., 71, 893-896.

KNUTH, D E (1969). The Art of Computer Programming. Volume 2. Semi-numerical Algorithms. Reading, Massachusetts. Addison Wesley.

KRONMAL, R A and PETERSON, A V (1978). On the Alias Method for Generating Random Variables from a Discrete Distribution. Technical Report No 17, Department of Biostatistics, University of Washington, Seattle.

KRONMAL, R A and PETERSON, A V (1979). On the Alias Method for Generating Random Variables from a Discrete Distribution. Amer. Statistician, 33, 214-218.

KRONMAL, R A and PETERSON, A V (1981). A Variant of the Acceptance-Rejection method for Computer Generation of Random Variables. J. Amer. Statist. Ass., 76, 446-451.

KRONMAL, R A, PETERSON, A V and LUNDBERG, E D (1978). The Alias-Rejection-Mixture Method for Generating Random Variables from Continuous Distributions. Statistical Computing Section, Proceedings of the American Statistical Association, 106-110.

LURIE, D and MASON, R L (1973). Emperical Investigation of Several Techniques for Computer Generation of Order Statistics. Communications in Statistics, 2, 363-371.

MACLAREN, M D, MARSAGLIA, G and BRAY, T A (1964). A Fast Procedure for Generating Exponential Random Variables. Commun. Ass. Comput. Mach., 7, 298-300.

MARSAGLIA, G (1961). Generating Exponential Random Variables. Ann. Math. Statist., 32, 899-902.

MARSAGLIA, G (1963). Generating Discrete Random Variables in a Computer. Commun. Ass. Comput. Mach., 6, 37-38.

MARSAGLIA, G (1964). Generating a Variable from the Tail of the Normal Distribution. Technometrics, 6, 101-102.

MARSAGLIA, G (1977). The Squeeze Method for Generating Gamma Variates. Computers and Mathematics with Applications, 3, 321-325.

MARSAGLIA, G and BRAY, T A (1964). A Convenient Method for Generating Normal Variables. SIAM Rev., 6, 260-264.

MARSAGLIA, G, MACLAREN, M D and BRAY, T A (1964). A Fast Procedure for Generating Normal Random Variables. Commun. Ass. Comput. Mach., 7, 4-10.

MCGRATH, E J and IRVING, D C (1973). Techniques for Efficient Monte Carlo Simulation, Vol II, Random Number Generation for Selected Probability Distributions. Science Applications, Inc., March 1973.

MICHAEL, J R, SCHUCANY, W R and HAAS, R W (1976). Generating Random Variates using Transformations with Multiple Roots. Amer. Statistician, 30, 88-90.

MONAHAN, J F (1979). Extensions of Von Neumann's Method for Generating Random Variables. Math. Comput., 33, 1065-1068.

NEWBY, M J (1979). The Simulation of Order Statistics from Life Distributions. Appl. Statist., 28, 298-301.

NEWBY, M J (1981). A Simple Method for Generating Samples from the Beta Density. J. Opl. Res. Soc., 32, 945-947.

NORMAN, J E and CANNON, L E (1972). A Computer Program for the Generation of Random Variables from any Discrete Distribution. J. Statist. Comput. Simul., 1, 331-348.

PAGE, E (1977). Approximations to the Cumulative Normal Function and its Inverse for use on a Pocket Calculator. Appl. Statist., 26, 75-76.

PAYNE, W H (1977). *Normal Random Numbers : Using Machine Analysis to* choose the Best Algorithm. ACM Transactions on Mathematical Software, 3, 346-358.

PARKER, J B (1976). Discussion of Atkinson, A C and Pearce, M C (1976). The computer generation of Beta, Gamma and Normal random variables. J R Statist. Soc. A., 139, 431-461.

RELLES, D (1972). A Simple Algorithm for Generating Binomial Random Variables when N is large. J. Amer. Statist. Ass., 67, 612-613.

ROBERTSON, I and WALLS, L A (1980). Random Number generators for the Normal and Gamma distributions using the ratio of Uniforms Method. United Kingdom Atomic Energy Authority, Harwell, AERE-R 10032.

RONNING, G (1977). A Simple Scheme for Generating Multivariate Gamma distributions with non-negative covariance matrix. Technometrics, 19, 179-183.

SCHMEISER, B W (1980). Generation of Variates from Distribution Tails. Operations Research, 28, 1012-1017.

SCHMEISER, B W and BABU, A J G (1980). Beta Variate Generation via Exponential Majorizing Functions. Operations Research, 28, 917-926.

SCHMEISER, B W and LAL, R (1980). Squeeze methods for Generating Gamma Variates. J. Amer. Statist. Ass., 75, 679-682.

SCHMEISER, B W and SHALABY, M A (1980). Acceptance/Rejection Methods for Beta Variate Generation. J. Amer. Statist. Ass., 75, 673-678.

SEAL, H L (1947). A Probability distribution of deaths at age x when policies are counted instead of lives. Skandivist, Aktuarietidskrift, 30, 18-43.

SEIGERSTETTER, J (1974). Discussion of Kendall, D G (1974). Pole-seeking Brownian motion and bird navigation. J R Statist. Soc. B, 36, 411-412.

SHUSTER, J (1968). On the inverse Gaussian distribution function. J. Amer. Statist. Ass., 63, 1514-1516.

TADIKAMALLA, P R (1978a). Computer generation of Gamma random variables. Commun. Ass. Comput. Mach., 21, 419-422.

TADIKAMALLA, P R (1978b). Computer generation of Gamma random variables-II. Commun. Ass. Comput. Mach., 21, 925-928.

TADIKAMALLA, P R and RAMBERG, T S (1975). An approximate method for generating Gamma and other variates. J. Statist. Comput. Simul., 3, 275-282.

TOCHER, K D (1963). The Art of Simulation. The English Universities Press Ltd.

VON NEUMANN, J (1951). Various Techniques used in Connection with Random Digits. National Bureau of Standards, Applied Mathematics, Series 12, June 1951, 36-38.

WALD, A (1947). Sequential Analysis. New York : John Wiley.

WALKER, A J (1977). An Efficient Method for Generating Discrete Random Variables with General Distributions. ACM Transactions on Mathematical Software, 3, 253-256.

WALLACE, N D (1974). Computer Generation of Gamma random variates with non-integral shape parameters. Commun. Ass. Comput. Mach., 17, 691-695.

WILSON, E B and HILFERTY, M M (1931). The distribution of chi-square. Proceedings of the National Academy of Sciences, Washington, 17, 684-688.

ZIPF, G K (1949). Human Behaviour and the Principle of Least Effort. Reading; Mass: Addison Wesley.