A NOTE ON THE ECOCGEN LANGUAGE BUILT-IN RANDOM DEVIATE GENERATORS
JORDI OCAÑA, M. CARMEN RUZ DE VILLA, GUILLEM ALONSO
UNIVERSITAT DE BARCELONA

The standard ECOCGEN (a simulation language based on Pascal) random deviate generators are described. For every one of them, a short usage note and a description of the algorithm and underlying theory is presented. This paper must be considered as an addenda to a previous one where the ECOCGEN language was described. The ECOCGEN random deviate generators include the continuous and discrete uniform, Poisson, binomial, exponential, Cauchy, normal or Laplace-Gauss, beta, gamma, Weibull, Pareto and Laplace distributions and the possibility of simulating repeated Bernoulli trials. Further developments are discussed, specially in the sense of allowing the application of variance reduction techniques.

Keywords: RANDOM NUMBER, RANDOM DEVIATE GENERATION, REJECTION, INVERSION, VARIANCE REDUCTION.

1. INTRODUCTION.

In a recent paper /13/, we described ECOCGEN, a discrete-event simulation language based on Pascal. ECOCGEN performs well as a general purpose simulation language (despite the fact its main design goal was to facilitate simulation in Population Biology, especially in Population Genetics). The before cited paper was something like an ECOCGEN mini-manual and preliminary report, previous to any implementation. Now the language is fully implemented. It runs on IBM 43XX and 30XX series. Pascal/VS language and VM/CMS operating system must be available. It may be obtained upon request to the authors.

In /13/ we argued that there was no need for a complete set of standard, built-in the language, facilities for random deviate generation. The argument was mainly based on the existence of good packages to do this.

Our practical experience with ECOCGEN during the last year has greatly changed the preceding point of view. First, we have realized that the most reputed and widely-used packages are not so convenient, in aspects like the possibility of maintaining separated random seed sequencies, of giving facilities for variance reduction techniques, or simply providing subroutines and functions for a wide range of probability distributions. Second, even users with with a non-trivial statistical training, may find some difficulties while implementing algorithms for some common distributions. They also frequently use not very good (but popular) algorithms, like the former Box-Muller method /7/ for the normal distribution (slow sin, cos and log computations, worse things may happen when it is used jointly with linear congruential uniform random number generators, see /8/).

In this paper we describe the actual ECOCGEN standard, built-in the language, set of random variate generators. It must be considered just an addenda to section 4.3 in /13/. According to its "user manual" view, only short usage notes are included. Implementation details and the study of its performances (comparing it with preexisting packages and the random deviate generators from other simulation languages) will be the subject of a --
forthcoming paper.

2. ECigen Standard Resources for Random Variate Generation.

A description of ECigen facilities for random variate generation follows. For every function we present its heading declaration, where the names of the arguments, their type and the result type are clearly stated. After the function heading declaration there is (if necessary) the distribution (or density) definitions, mainly to show the meaning of the parameters. Every generator description is closed by a short discussion on the algorithm and its underlying theory.

First, assume the following type declarations

```plaintext
nonegint = 0..maxint;
rangeseed = 1..maxint;
posint = rangeseed;
```

maxint being an integer constant with the greatest (implementation dependent) positive integer value. The first argument of all -- functions is a variable called nseed of rangeseed type. It is the random seed, its value being changed in every case, while -- performing the random variate generation.

We will often use the not very precise (but short) phrase "...returning a value with... distribution..." instead of something like "...returning a value obtained by means of a random experience associated to a random variable with...distribution..." longer and not much more precise.

1. function rand (var nseed: rangeseed): real;

This is the most basic resource, returning a value with uniform distribution over the (0, 1) interval. It is based on a portable generator described in /8/. Possibly, it will be changed in the future.

2. function discrand(var nseed: range: rangeseed; n: posint): posint;

Returning a value with discrete uniform distribution over {1,2,...,n},

\[ \text{Prob } (X = i) = \frac{1}{n}, \text{ if } i = 1,2,\ldots,n. \]

3. function trial (var nseed: rangeseed, p: real): boolean;

Simulating the possible occurrence of an event with probability \( p \), \( 0 \leq p \leq 1 \). It returns value true with \( p \) probability and false with \( 1 - p \) probability.

4. function Poisson (var nseed: rangeseed; lambda: real): nonegint;

Returning a value with Poisson distribution of Lambda (lambda > 0) parameter:

\[ \text{Prob } (X=x) = \exp(-\lambda) \frac{\lambda^x}{x!}, \text{ if } x=0,1,\ldots \]

This is based on the well known (and not very efficient) algorithm founded in the relation between the exponential and Poisson distributions. For large values of lambda (greater than 6) a normal approximation

\[ \frac{1}{\sqrt{2\pi}} \]

is used.

5. function binomial (var nseed: rangeseed; n: nonegint; p: real): nonegint;

Returning a value with binomial distribution of \( 0 \leq p \leq 1 \) and \( n \geq 0 \) parameters:

\[ \text{Prob } (X = x) = \binom{n}{x} p^x (1-p)^{n-x}, \]

if \( x = 0,1,\ldots,n \).

Its algorithm is based on counting the absolute frequency of an event with \( p \) probability. \( n \) independent trials are performed by means of the trial function.

For large \( n \) values (%n > 30%):

- When \( p \) is intermediate (0.1<p<0.9) a normal

\[ \frac{N(n, np(1-p))}{\sqrt{2\pi}} \]

approximation is used.

- When \( p \) is small (\( p \leq 0.1 \)) a Poisson deviate with lambda = np parameter, \( P(np) \), approximation is used.

- When \( p \) is large (\( p \geq 0.9 \)) the variable is generated as \( n - Y \), where \( Y \) is taken as a Poisson \( P(n(1-p)) \).
6. function hipoergometric(var n.seed: ranseed; 
N,M,nex : nonegint); nonegint;

Returning a value with hipoergometric dis-
tribution
\[ P(X = x) = \binom{M}{x} \binom{N-nex}{x} \binom{N}{M} \]
if \( \max(0,N-nex-N) \leq x \leq \min(N,nex) \) (integer x)

where 0 \( \leq M \leq N \) and 0 \( \leq n \leq N \).

The algorithm is based on directly simu-
lating the following experiment: nex "balls" are "drawn" at random and with-
out replacement from an "urn" containing N balls, some of them (M) being "marked". x is the number of marked balls in the (size nex) random sample. For large N values, more exactly, when 0.1N > nex (see /10/) a binomial, B(nex, N/N) approx-
imation is used. This binomial is itself approximated by a Poisson P (nex N/N) or by a Normal N(m,s) with /10/.

\[ m = \frac{n \cdot M}{N} \]
\[ s = \left\{ \frac{(N - nex)/N}{(nex M/N)(1 - M/N)} \right\}^{1/2} \]
according to the same criteria used for the binomial distribution.

7. function geometric(var n.seed: ranseed; p: real); nonegint;

Returning a value with geometric distri-
bution
\[ P(X = x) = (1-p)^{x} \cdot p, \text{ if } x = 0,1,2,\ldots \]
0 \( \leq p \leq 1 \).

For large values of p (p \( \geq 0.5 \)), random independent trials (function trial) are performed until the occurrence of the p probability event (trial = true). For -- smaller values of p, as this algorithm would be slow (too many trials until oc-
currence of the event), direct inversion of the commulative distribution function is performed:
\[ X = [\ln(1 - U) / \ln(1 - p) \]
where \( \lfloor \rfloor \) is the lower part function and U is a random variable with uniform distribution over (0,1) (function rand).

The critical value \( p = 0.5 \) has been em-
pirically determined.

8. function negbinom(var n.seed: ranseed; r, n: real); nonegint;

Returning a value with negative binomial dis-
tribution
\[ P(X = x) = \binom{r+x-1}{x} \cdot r^{x} \cdot (1 - p)^{r}, \text{ if } x = 0,1,2,\ldots \]
with \( r \geq 0 \) and 0 \( \leq p \leq 1 \).

Direct composition of a Poisson with a gamma distribution is performed: a Pois-
son deviate is generated, with random -- lambda parameter drawn from a gamma dis-
tribution with \( a = r \) and \( b = (1-p)/p \) pa-
rameters.

9. function exponential(var n.seed: ran-
seed; b: real); real;

Returning an exponentially distributed value
\[ f(x) = \frac{1}{b} \cdot \exp(-x/b), \text{ if } x \geq 0 \]
with \( b > 0 \).

Direct inversion of the cummulative dis-
tribution function is performed:
\[ x = -b \ln(1 - U) \]

10. function Cauchy(var n.seed: ranseed; 
m(* median *), s(* scale *): real):real;

Returning a value with Cauchy distri-
bution
\[ f(x) = \frac{1}{\pi} \cdot \frac{s}{s^2 + (x - m)^2} \text{ if } -\infty < x < \infty \]
with \( -\infty < m < \infty \) and \( s > 0 \).

Direct inversion is performed. This algo-
"
to the tangent function $1/x$ is faster than the rejection algorithm based on the well-known fact that the quotient between the coordinates of a point randomly chosen (with uniform distribution) over the two-dimensional unit circle has a standard ($m = 0$, $s = 1$) Cauchy distribution.

11. function normal(var nseed; ngseed; m(* mean and median *), s(* standard deviation *)); real); real;

Returning a value with normal distribution

$$f(x) = \frac{1}{s \sqrt{2\pi}} \exp\left(-\frac{(x-m)^2}{2s^2}\right), \text{ if } -\infty < x < \infty$$

with $-\infty < x < \infty$ and $s > 0$.

This uses the rational approximation of Odeh and Evans /14/ for the inverse standard normal cumulative distribution function:

$$X = \Phi^{-1}(Y) = Y + \frac{P(Y)}{Q(Y)}$$

where

$$Y = \ln\left((1-U)^2\right)$$

and $P$, $Q$ are degree 4 polynomials.

12. function beta(var nseed; ngseed; a, b; real); real;

Returning a value with beta distribution

$$f(x) = \frac{x^{a-1}(1-x)^{b-1}}{\beta(a,b)}, \text{ if } 0 < x < 1$$

where $\beta$ is the "beta function" and $a > 0$, $b > 0$. The algorithm is based on rejection, combining the method from Jönk /6/ for small values of $a$ and $b$ ($a+b<1$) and the method of Cheng /9/ for larger values of $a+b$ ($>1$).

13. function gamma(var nseed; ngseed; a, b; real); real;

Returning a value with gamma distribution,

$$f(x) = \frac{x^{a-1}}{a^{\Gamma(a)}} \exp\left(-\frac{x}{b}\right), \sum_{n=0}^{\infty} \frac{1}{a^n b^n}$$

with $a > 0$ and $b > 0$, where $\Gamma$ is the "gamma function". This is based on the rejection algorithms from Ahrens and Dieter /3/ when $a < 1$ and /4/ otherwise.

14. function Weibull(var nseed; ngseed; b, c; real); real;

Returning a value with Weibull distribution. The expression for the cumulative distribution function is easier to write than the corresponding density:

$$\text{Prob}\{X \leq x\} = 1 - \exp\left(-\frac{x}{b}\right)^c,$$

$$\text{if } x \geq 0$$

with $c > 0$, $b > 0$.

This is generated directly by inversion:

$$X = b \left(\text{exponential (nseed,1)}\right)^{1/c}$$

15. function Pareto(var nseed; ngseed; a, b; real); real;

Returning a value from a Pareto distribution. It is customary to define such a distribution from

$$\text{Prob}\{X > x\} = \left(\frac{a}{x}\right)^b, \text{ if } x \geq 1$$

with $a > 0$ and $b > 0$.

It is directly generated by inversion:

$$X = \frac{a}{(1-U)^{1/b}}$$

16. function Laplace(var nseed; ngseed; m(* mean and median *), s(* scale *); real);real;

Returning a value with Laplace (or type I error) distribution:

$$f(x) = \frac{1}{2\pi} \exp\left(-\frac{|x-m|}{s}\right), \text{ if } -\infty < x < \infty$$

where $-\infty < m < \infty$ and $s > 0$. 146
It is directly generated by inversion:

\[ X = \begin{cases} 
    \eta + \gamma \ln(2\eta), & \text{if } \eta \leq 0.5 \\
    \eta - \gamma \ln(2(1-\eta)), & \text{if } \eta > 0.5 
\end{cases} \]

3. FURTHER DEVELOPMENTS AND DISCUSSION.

Apart from questions about some obviously necessary improvements like the provision of generators for more distributions (mainly for some multivariate distributions) and for some basic stochastic processes, many questions arise on the algorithms at present in use. Why is inversion used instead of other algorithms that are in principle more efficient, for some distributions like the exponential /1/ or the normal /2/? Why not introduce some facilities for generation from tabulated distributions like a "buckets" method /5/ or an "alias" method /12/?, etc.

We think that these questions must be answered under a more general view. Our project is to provide the ECOGEN user with the possibility of deciding (in some extent) the methods.

As the present running implementation (under VM/CMS and based on Pascal/VMS) is based on a preprocessor translating from ECOGEN into Pascal, there will be preprocessor options introducing variance reduction or not (the latter being the standard possibility).

Under the first option, inversion or other generators, all based on monotonic transforms of uniform deviates, will be provided to ensure the possibility of applying variance reduction techniques. As is well known, the most widely applied (in model simulation) variance reduction techniques, as antithetic variates and common random numbers, lie on monotonicity and synchronization assumptions. Inversion methods based on transforming a single random number by means of the inverse of the cumulative distribution function, appear as the best possibility (in variance reduction). If necessary, slow (in setting time) generators -- from previously tabulated distributions will be used (this will be for example the case for the gamma distribution, with no widely applicable analytical or empirical formulas for the inverse of the distribution function).

Under the second option, the main coal will be speed. The most efficient methods (to the extent they are known by the implementors) will be provided, disregarding any other consideration like the possibility of correctly applying variance reduction techniques.

4. REFERENCES.


/11/ KNUTH, D.E.: "The Art of Computer Pro-
gramming". Vol. 2. Addison-Wesley.
1981.

/12/ KRONMAL, R.A., PETERSON, A.V.: "On the
alias method for generating random va-
riables from a discrete distribution".

/13/ OCAFÀ, J.: "The ECODEN language for ge-
etic simulation". QUESTIO Vol. 9:
7-34. 1985.

/14/ GDEH, R.E., EVANS, J.O.:"Algorithm AS
70: percentage points of the normal dis-
1974.