SAMPLING FROM DISCRETE AND CONTINUOUS DISTRIBUTIONS WITH C–RAND

Ernst Stadlober Ralf Kremer
Institut für Statistik, Techn. Universität Graz
Lessingstraße 27, A–8010 Graz, Austria
e–mail: statistik@rech.tu–graz.ada.at

Abstract

C–RAND is a system of Turbo–C routines and functions intended for use on microcomputers. It contains up–to–date random number generators for more than thirty univariate distributions. For some important distributions the user has the choice between extremely fast but rather complicated methods and somewhat slower but also much simpler procedures. Menu driven demo programs allow to test and analyze the generators with regard to speed and quality of the output.

1. Introduction

The purpose of our package is to provide random number generators for classical distributions as well as for nonstandard models which are of growing importance in simulation. However, most simulation packages may not be able to handle some of the more exotic models. Therefore the researcher my need some routines of our collection to perform his own specific simulation problem. For this he should be prepared at least to link the C–programs to his own software. The C–functions written in Turbo C 2.0 are designed for use on an IBM PC/AT or equivalent microcomputer with an 80287 coprocessor. The source codes of each generator are rather self-documented. They contain comments about references, the purpose of the specific algorithm, the meaning of input and output parameters and statements about subprograms needed. Detailed inline remarks should help to look through the programs if necessary. We decided also to incorporate only C–routines for the uniform generators in order to maintain the portability of the functions. Hence it should be no problem to transfer the programs and run them on workstations or other computers.

For most generators demonstration programs are accessible which run at the moment on PC's with EGA graphics environment (monocolor or color). With this the user may study some performance measures of generators, he could also look at graphs of distributions with respect to changing shape parameters or he could observe whether the output of different random number sequences from the same generator behave like random samples.

2. Uniform Random Number Generators

Random numbers u_i from the (0,1) uniform distribution ($U(0,1)$) are generated by the *multiplicative congruential method.* They are based on sequences

$$z_0 \equiv 1 \,(\mathrm{mod}\ 4), \quad z_{i+1} \equiv a\ z_i \,(\mathrm{mod}\ 2^E), \quad a \equiv 5 \,(\mathrm{mod}\ 8). \tag{1}$$

During one period of length $n = 2^{E-2}$ all integers of the form $0 < 4k+1 < m = 2^E$ are generated as z_i. The derived quotients $u_i = z_i/m$ are the required uniform random numbers in (0,1).

It is well known (see Ripley, 1987) that an adequate generator should have a long period and should produce random numbers u_i in such a way that the k–tuples $(u_i, \ldots, u_{i+k-1})$ are very close to the uniform distribution in the hypercube $(0,1)^k$, i.e. the generated k–tuples should fill a lattice in $(0,1)^k$. In C–RAND two generators — written in C as mentioned in the introduction — are implemented: the simple generator `drand` with period 2^{30} and the more complicated generator `krand` with longer period 2^{52}. Their special choices for multipliers a and moduls m are as follows.

`drand` : $a = 663608941,\ m = 2^{32}$; `krand` : $a = 2783377641436325,\ m = 2^{54}$.

The multipliers are close to $\theta n = \theta 2^{E-2}$ where θ is the golden section number $\theta = \frac{1}{2}(\sqrt{5}-1)$. These values are motivated by the theory of the distribution of pairs (u_i, u_{i+1}) in $(0,1)^2$ (see Dieter, 1971). The lattice structure up to 8 dimensions induced by these multipliers has been assessed by the spectral test of Coveyou and MacPherson (1967), and the results indicate that our choices behave 'well' also in higher dimensions. Our code of `drand` on an IBM PC is:

```
double drand(prand)
unsigned long *prand;
{
union ur { double d; int i[4];} x;
x.d = *prand *=0X278DDE6DL;
x.i[3] -=0X0200;
return(x.d);
}
```

`*prand` must be initialized in the calling program to some `*prand = 4*k+1;` $(k = 0, 1, \ldots)$. Then the generator may be called by the C–statement `u = drand(prand);`. Standard initialization with `*prand = 1;` should result in the following sequence of random numbers: $u_1 = .15450849686749$, $u_2 = .98173871100880, u_3 = .35125617426820, u_4 = .82583231129684, u_5 = .54327537049538, \ldots$, $u_{20} = .46241982677020, \ldots, u_{50} = .92243475676514, \ldots, u_{100} = .32570007839240, \ldots$.

These values may be helpful to the user for checking the output of the generator on his own machine. All algorithms for generating nonuniform variates in the package use this basic uniform generator `drand` as stated above. But it could be easily replaced by `krand` if a longer period is preferred for some reason. Additionally the observed performance of the ready–made C–routines on the 80286 AT with 12 MHz 80287 coprocessor might be of interest.

`drand` occupies 313 bytes compiled code and needs 72μsec per sample.
`krand` occupies 570 bytes compiled code and needs 332μsec per sample.

3. Non–Uniform Random Number Generators

The main task of non–uniform random number generation is to convert the (0,1)–uniforms into variates that follow a given probability distribution $F(x)$. Many specific sampling algorithms for common distributions exist. They are based on theoretically exact transformation schemes: *inversion, rejection, rejection/composition, ratio of uniforms, acceptance/complement* and *distributional properties*. For a thorough description of the theory we refer to the books of Devroye (1986) and Bratley et al. (1987). For the development of our random number package we examined more than 80 algorithms and implemented the fastest generators found (see Kremer, 1989). We denote them as *efficient* procedures. In case of distributions with one or more shape parameters (e.g. gamma and Poisson (1 par.), beta and binomial (2 par.)) they are fast even when the parameters are frequently changing. However, several of these methods need long auxiliary tables and the programs are rather sophisticated. Hence we offer also so called *convenient* procedures for some distributions. These methods are based on a simple theory resulting in short programs and they still maintain their reasonable speed regardless of shifting parameter values.

3.1 Sampling from continuous distributions

The selected sampling methods for various continuous distributions without any shape parameter and with one or two shape parameters are listed in Table 1 below. It suffices to consider only algorithms for standardized distributions with location parameter = 0 and scale parameter = 1. A remarkable feature is that some of the methods are nothing but inversion by $X = F^{-1}(U)$ or $X = F^{-1}(1-U)$ where $U \sim U(0,1)$.

The very important case of the *exponential* distribution results in $X = F^{-1}(1-U) = -\ln U$ which is now superior to different methods (Ahrens and Dieter, 1972, 1988), since in a PC environment with floating point coprocessor the evaluation of a logarithm is very cheap ($\sim$ 4 floating point additions).

For the *normal* distribution more than a dozen published methods have become fairly well–known. A *convenient* method is based on the transformation of Box and Muller (1958):

$$X = \sqrt{-2\ln U}\cos(2\pi V), \quad Y = \sqrt{-2\ln U}\sin(2\pi V), \tag{2}$$

which converts the two independent (0,1)–uniforms U and V into the two independent standard normal variates X and Y (sine–cosine method `nsc`). The fastest known normal generators rely on auxiliary tables containing tables of constants, which depend usually on the employed computer arithmetic (see Knuth, 1981). Recently, Ahrens and Dieter (1989) developed the *alias* method `nal` which is at least as fast as the older methods but simpler. Another advantage of `nal` is that the three tables of 128 one–byte integers needed *do not* depend on the precision of the arithmetic. However, the performance of the algorithm depends heavily on the programming language, since there is some bit or byte manipulation involved. Coding `nal` in C could be done very efficiently, because of its standard facilities for logical and shift operations. The execution times (compiled codes) of `nal` and `nsc` are 117μsec (1828 bytes) and 593μsec (624 bytes), respectively. `nal` needs 1.0618 uniforms per sample, but `nsc` requires exactly 1 uniform per normal variate.

Gamma distributions with shape parameters a received some attention in the literature (see Zechner, 1990, for an extensive comparison). The most efficient method `gd` ($a > 1$) is *acceptance/complement* with transformed normal variates (Ahrens and Dieter, 1982a), provided that a fast normal generator (in our case `nal`) is utilized. Our generator `gds` combines `gd` with the simple rejection method `gs` ($a \leq 1$) due to Ahrens and Dieter (1974). As *convenient* procedure rejection with log–logistic envelopes, denoted by `gll`, is employed (Cheng, 1977). The performance measures for `gds` and `gll` in case of $a \geq 1$ are listed below.

`Algorithm`	`gds`	`gll`
Execution times (μsec/sample)	560–1000	1300–2000
Set–up times (μsec/sample)	160 – 260	160 – 260
Compiled code (bytes)	5465	1113
No. of uniforms ($a = 1 \mid a \to \infty$)	1.97 | 1.57	2.94 | 2.26

Ratio of uniforms introduced by Kinderman and Monahan (1977) is the method of choice for *Chi(a)*, *Student–t(a)* and *generalized inverse Gaussian(λ, β)* distributions. The algorithms seem to be suitable combinations of convenience and speed. Good rejection algorithms exist for the *exponential–power(τ)*, *von Mises(k)* and *beta(a, b)* distributions. Generation from the *stable(a, d)* model is based on distributional properties. Sampling from the families of *Johnson* $S_B, S_L, S_U(m,s)$ distributions is done by their definitions as transformed normal variates. For the members of the *Burr*–family and various other distributions closed form inversion of $F(x)$ is possible. Table 1 below gives an overview of generators for all continuous distributions covered by C–RAND.

Table 1. Generators for continuous distributions		
Distribution	`Algorithm`	**Method and Reference**
No shape parameter		
Cauchy	`cin`	$X = \tan(\pi U)$
Exponential	`eln`	$= -\ln U$
Extreme valueI	`ev_I`	$= -\ln(-\ln U)$
Laplace (Double Exponential)	`lapin`	$= \pm \ln U$
Logistic	`login`	$= \ln(U/(1-U))$
Normal	`nal`	*Alias*, Ahrens/Dieter(1989)
	`nsc`	*sin/cos*, Box/Muller(1958)
Slash	`sla`	$X = Z/U$, $Z \sim N(0,1)$, $U \sim U(0,1)$
Triangular	`tra`	$X = \pm(1-\sqrt{U})$
One shape parameter		
BurrII(r), $r > 0$	`bu_II`	$X = -\ln(U^{-1/r} - 1)$, Burr(1942)
BurrVII(r), $r > 0$	`bu_VII`	$= \frac{1}{2}\ln((1+Y)/(1-Y))$, $Y = 2\,U^{1/r} - 1$
BurrVIII(r), $r > 0$	`bu_VIII`	$= \ln(\tan(\pi\, U^{1/r}/2))$
BurrX(r), $r > 0$	`bu_X`	$= \sqrt{-\ln(1-U^{1/r})}$
Chi(a), $a > 1$	`chru`	*Ratio of Uniforms*, Monahan(1987)
Exp.–Power(τ), $\tau \geq 1$	`epd`	*Uniform–Exponential Rejection*, Devroye(1986)
Extreme valueII(a), $a > 0$	`ev_II`	$X = (-\ln U)^a$
Gamma(a), $0 < a \leq 1\,/\,a > 1$	`gds`	*Rej./Acc.–Compl.*, Ahrens/Dieter(1974,1982a)
$a > 0$	`gll`	*Log–Logistic Rejection*, Cheng(1977)
Student–t(a), $a \geq 1$	`trouo`	*Ratio of Uniforms*, Kinderman/Monahan(1980)
von Mises(k), $k > 0$	`mwc`	*Wrapped Cauchy Rejection*, Best/Fisher(1979)
Weibull(c), $c > 0$	`win`	$X = (-\ln(1-U))^{1/c}$
Two shape parameters		
Beta(a,b), $a,b > 0$	`bbbc`	*Log–Logistic Rejection*, Cheng(1978)
BurrIII(r,k), $r,k > 0$	`bu_III`	$X = Y^{-1/k}$, $Y = U^{-1/r} - 1$
BurrIV(r,k), $r,k > 0$	`bu_IV`	$= k/(Y^k + 1)$
BurrV(r,k), $r,k > 0$	`bu_V`	$= \arctan(-\ln(Y/k))$
BurrVI(r,k), $r,k > 0$	`bu_VI`	$= \ln(Z + \sqrt{Z^2+1})$, $Z = -\frac{1}{r}\ln(Y/k)$
BurrIX(r,k), $r,k > 0$	`bu_IX`	$= \ln(W^{1/r} - 1)$, $W = 1 + 2\,U/(k(1-U))$
BurrXII(r,k), $r,k > 0$	`bu_XII`	$= Y^{1/k}$
G. I. Gauss(λ,β), $\lambda \geq 0, \beta > 0$	`gigru`	*Ratio of Uniforms*, Lehner(1989)
Johnson $S_B(m,s)$, $s > 0$	`j_sb`	$X = \exp(sZ+m)/(1+\exp(sZ+m))$, $Z \sim N(0,1)$, Johnson(1949)
Johnson $S_L(m,s)$, $s > 0$	`j_sl`	$= \exp(sZ+m)$, (Lognormal)
Johnson $S_U(m,s)$, $s > 0$	`j_su`	$= \sinh(sZ+m)$
Lambda(λ_1,λ_2), $\lambda_1,\lambda_2 \geq 0$	`lamin`	$= U^{\lambda_1} - (1-U)^{\lambda_2}$, Ramberg/Schmeiser(1974)
Stable(a,d), $0 < a \leq 2$, $\mid d \mid \leq 1$	`stab`	*Distributional Properties*, Chambers et al.(1976)

3.2 Sampling from discrete distributions

Efficient sampling methods have been developed for the classical *Poisson(μ)*, *binomial(n, p)* and *hypergeometric(N, M, n)* distributions. Poisson generator `pd`, due to Ahrens and Dieter (1982b), utilizes *acceptance–complement* with truncated normal variates whenever $\mu \geq 10$ and reverts to *table–aided inversion* otherwise. Binomial and hypergeometric distributions are handled by means of *composition–rejection* procedures (triangular–parallelogram–exponential method `btpe` and uniform–exponential method `h2pe`). Algorithms `btpec` (Kachitvichyanukul and Schmeiser 1988a, 1990) and `h2pec` (Kachitvichyanukul and Schmeiser 1985, 1988b) combine composition–rejection with *unstored inversion*, where inversion is performed as sequential search from the bottom using the recurrence relationships $p_k = g(p_{k-1})$.

Recently the *ratio of uniforms* method has been adapted to the distributions mentioned above (Poisson: Ahrens and Dieter, 1990, Stadlober, 1989a; binomial: Stadlober, 1989a, 1989b; hypergeometric: Stadlober, 1989a, 1990). It requires that the standardized histogram function $f(x) = p_k/p_m$, $k \leq x < k+1$, $p_m = \max_k p_k$, of the target discrete distribution is fitted under a (linearly transformed) *table mountain* $h(x) = \min(1, s^2/(x-a)^2)$ where a and s are suitably chosen parameters. With this the method can be stated as the following rejection procedure.

> *Generate a pair* (U, V) *uniformly distributed over the rectangle* $R = [0,1] \times [-1,1]$, *set* $X = s\,V/U + a$ *and return* $K = \lfloor X \rfloor$ *as sample from* $f(x)$ *whenever* $U^2 \leq f(X)$ *is fulfilled. Otherwise reject* X *and try again.*

This yields *convenient* algorithms with bounded computation times, i.e. the times do not grow with the parameters of the distributions. Again unstored or *simple inversion* is substituted in the utility routines `pruec`, `bruec` and `hruec` when it is faster than ratio of uniforms (see Table 2 below).

The *geometric* distribution with probability of success p and probability function $p_k = p\,(1-p)^k$, $k = 0, 1, \ldots$ permit inversion by $K = \lfloor \ln U/(1-p) \rfloor$. Its generalization, the *negative binomial(r, p)* distribution is the discrete counterpart of the gamma distribution. Sampling is carried out by *simple inversion* whenever mean $\mu = r(1-p)/p \leq 10$. It is complemented for $\mu > 10$ by the *compound* method of Ahrens and Dieter (1974) which uses a relationship to gamma and Poisson variates. Generators for the *logarithmic(p)* distribution have been constructed by Kemp (1981), who suggested to use a combination of *inversion* ($p < .9$) and *distributional properties* ($p \geq .9$). Rare events in linguistics or insurance may be simulated by the two parameter *Zeta(ρ, τ)* distribution via a *rejection* scheme proposed by Dagpunar (1988).

A general discrete random number generator is based on the ingenious *alias* method of Walker (1977). It uses the fact that any discrete distribution with n mass points can be expressed as a mixture of $n-1$ equiprobable two point distributions. Two tables have to be constructed from the given probabilities p_k: the 'aliases' a_j (integers in $\{1, \ldots, n\}$) and the 'alias probabilities' q_j ($j = 1, \ldots, n-1$). The algorithm for this table set–up is of O(n) time. The sampling procedure is very easy and uses exactly one uniform random number U.

> *Generate* $U \sim U(0,1)$, *transform it to* $X = (n-1)\,U$ $(X \sim U(0, n-1))$, *set* $K = \lceil X \rceil$ $(K \sim U\{1, \ldots, n-1\})$ *and* $U = K - X$ ($U \sim U(0,1)$ *again*). *If* $U \leq q_K$ *return* K *('criminal'), otherwise return* a_K *('alias') as sample from* p_k.

The method is extremely fast if the distribution stays fixed, but any change of parameters requires a new set–up procedure (new a_j and q_j).

Table 2. Generators for discrete distributions		
Distribution	`Algorithm`	**Method and Reference**
One shape parameter		
Geometric(p), $0 < p < 1$	`geo`	$K = \lfloor \ln U/(1-p) \rfloor$
Logarithmic(p), $0 < p < 1$	`lsk`	*Inversion/Distr. Prop.*, Kemp(1981)
		$(p < .9)$ $(p \geq .9)$
Poisson(μ), $\mu > 0$	`pd`	*Table Inv./Acc.–Compl.*, Ahrens/Dieter(1982b)
		$(\mu < 10)$ $(\mu \geq 10)$
	`pruec`	*Inversion/Ratio of Unif.*, Stadlober(1989a)
		$(\mu < 5)$ $(\mu \geq 5)$
Two shape parameters		
Binomial(n, p), $n \geq 1, 0 < p < 1$,	`btpec`	*Inversion/Rej.–Comp.*, Kachitvichyanukul/
$\ell = \min(np, n(1-p))$		$(\ell < 10)$ $(\ell \geq 10)$ Schmeiser(1988a,1990)
	`bruec`	*Inversion/Ratio of Unif.*, Stadlober(1989a,b)
		$(\ell < 5)$ $(\ell \geq 5)$
Neg. Bin.(r, p), $r > 0, 0 < p < 1$,	`nbp`	*Inversion/Compound*, Ahrens/Dieter(1974)
$\mu = r(1-p)/p$		$(\mu < 10)$ $(\mu \geq 10)$
Zeta(ρ, τ), $\rho > 0, \tau \geq 0$	`zet`	*Rejection*, Dagpunar(1988)
Three shape parameters		
Hypergeometric(N, M, n),	`h2pec`	*Inversion/Rej.–Comp.*, Kachitvichyanukul/
$N, M, n \geq 1; n \leq N, M \leq N$,		$(\ell < 10)$ $(\ell \geq 10)$ Schmeiser(1985,1988b)
$m = \lfloor (n+1)(M+1)/(N+2) \rfloor$,	`hruec`	*Inversion/Ratio of Unif.*, Stadlober(1989a,1990)
$\ell = m - \max(0, n - N + M)$		$(\ell < 5)$ $(\ell \geq 5)$
Arbitrary distributions		
Given prob. $p_k, k = 1, \ldots, n$	`gdal`	*Alias Method*, Walker(1977)

4. An example of a demo program

Menu driven demo programs which run with EGA graphics card allow to visualize some features of the procedures implemented. As an example let us consider the flexible beta(a, b) distribution with generator `bbbc` of Cheng (1978). Starting the `bbbc.exe` program produces an image on the screen which is partitioned into seven windows. Two of these windows ('Beta Distribution', 'BBBC') are fixed, two windows allow for some input ('PAR...', 'Seed=...'), whereas in the three remaining windows information about the output is contained. The user can now easily select input windows in order to change some input parameters or he can request to open output windows for looking at the results. Output window 'Moments' delivers the mean number of uniforms needed for one sample, empirical and theoretical mean, variance, standard deviation, skewness and kurtosis of the random deviates. The execution times and set–up times (both in μsec per sample) appear in the window 'Timing'. In 'Graphics' the histogram of the generated random numbers is compared with the true density function marked by dots (see Figure 1). The 'true randomness' of the random samples may be studied by viewing histograms and moments of random number sequences with different seed values of the underlying uniform generator `drand` as demonstrated in Figure 2. Different shapes of the densities can be displayed by varying the shape parameters of the distribution ($a = 0.4, b = 0.6$: ' U–shaped'; $a = 0.8, b = 1.2$: 'J–shaped') as visualized in Figure 3.

Figure 1: Moments, Timing and Graphics

Figure 2: Effect of different seeds

Figure 3: Varying shape parameters

Acknowledgement

We are grateful to Jo Ahrens for his valuable help in implementing the uniform generators `drand`, `krand` and his normal generator `nal`.

References

Ahrens, J.H. and Dieter, U. (1972): *Computer methods for sampling from the exponential and normal distributions*, Comm. ACM **15**, 873–882.

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. (1982a): *Generating gamma variates by a modified rejection technique*, Comm. ACM **25**, 47–54.

Ahrens, J.H. and Dieter, U. (1982b): *Computer generation of Poisson deviates from modified normal distributions*, ACM Trans. Math. Software **8**, 163–179.

Ahrens, J.H. and Dieter, U. (1988): *Efficient table–free sampling methods for the exponential, Cauchy and normal distributions*, Comm. ACM **31**, 1330–1337.

Ahrens, J.H. and Dieter, U. (1989): *An alias method for sampling from the normal distribution*, Computing **42**, 159–170.

Ahrens, J.H. and Dieter, U. (1990): *A convenient sampling method with bounded computation times for Poisson distributions*, Amer. J. Math. Management Sci. **25**, in press.

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, M.E. (1958): *A note on the generation of random normal deviates*, Ann. Math. Statist. **29**, 610–611.

Bratley, P., Fox, B.L. and Schrage, L.E. (1987): *A Guide to Simulation, 2^{nd} Edition*, Springer, New York.

Burr, I.W. (1942): *Cumulative frequency functions*, Ann. Math. Statist. **13**, 215–232.

Chambers, J.M., Mallows, C.L. and Stuck, B.W. (1976): *A method for simulating stable random variables*, J. Amer. Statist. Assoc. **71**, 340–344; (Correction: J. Amer. Statist. Assoc. **82**, (1987), 704).

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 nonintegral shape parameters*, Comm. ACM **21**, 317–322.

Coveyou, R.R. and MacPherson, R.D. (1967): *Fourier analysis of uniform random number generators*, J. Assoc. Comput. Mach. **14**, 100–119.

Dagpunar, J. (1988): *Principles of Random Variate Generation*, Clarendon Press, Oxford.

Devroye, L. (1986): *Non–Uniform Random Variate Generation*, Springer, New York.

Dieter, U. (1971): *Pseudorandom numbers: the exact distribution of pairs*, Math. Comp. **25**, 855–883.

Johnson, N.L. (1949): *Systems of frequency curves generated by methods of translation*, Biometrika **36**, 149–176.

Kachitvichyanukul, V. and Schmeiser, B.W. (1985): *Computer generation of hypergeometric random variates*, J. Statist. Comput. Simulation **22**, 127–145.

Kachitvichyanukul, V. and Schmeiser, B.W. (1988a): *Binomial random variate generation.* Comm. ACM **31**, 216–222.

Kachitvichyanukul, V. and Schmeiser, B.W. (1988b): *ALGORITHM 668 H2PEC: Sampling from the hypergeometric distribution*, ACM Trans. Math. Software **14**, 397–398.

Kachitvichyanukul, V. and Schmeiser, B.W. (1990): *ALGORITHM BTPEC: Sampling from the binomial distribution*, ACM Trans. Math. Software **16**, to appear.

Kemp, A.W. (1981): *Efficient generation of logarithmically distributed pseudo–random variables*, Appl. Statist. **30**, 249–253.

Kinderman, A.J. and Monahan, J.F. (1977): *Computer generation of random variables using the ratio of uniform deviates*, ACM Trans. Math. 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.

Knuth, D.E. (1981): *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 2^{nd} Edition*, Addison Wesley, Reading.

Kremer, R. (1989): *C–RAND: Generatoren für nicht–gleichverteilte Zufallszahlen*, Diplomarbeit, 152 pp., Techn. Universität Graz.

Lehner, K. (1989): *Erzeugung von Zufallszahlen für zwei exotische stetige Verteilungen*, Diplomarbeit, 107 pp., Techn. Universität Graz.

Monahan, J.F. (1987): *An algorithm for generating chi random variables*, ACM Trans. Math. Software **13**, 168–172; (Correction: ACM Trans. Math. Software **14** (1988), 111).

Ramberg, J.S. and Schmeiser, B.W. (1974): *An approximate method for generating asymmetric random variables*, Comm. ACM **17**, 78–82.

Ripley, B.D. (1987): *Stochastic Simulation*, John Wiley, New York.

Stadlober, E. (1989a): *Sampling from Poisson, binomial and hypergeometric distributions: ratio of uniforms as a simple and fast alternative*, Math. Statist. Sektion **303**, 93 pp., Forschungsgesellschaft Joanneum Graz.

Stadlober, E. (1989b): *Ratio of uniforms as a convenient method for sampling from classical discrete distributions*, Proc. 1989 Winter Simulation Conf., Eds. E.A. MacNair et al., 484–489.

Stadlober, E. (1990): *The ratio of uniforms approach for generating discrete random variables*, J. Comput. Appl. Math. **31**, 181–189.

Walker, A.J. (1977): *An efficient method for generating discrete random variables with general distributions*, ACM Trans. Math. Software **3**, 253–256.

Zechner, H. (1990): *Erzeugung gammaverteilter Zufallszahlen mit allgemeinem Formparameter*, Grazer Math. Berichte **311**, 123 pp., Graz.