An Error in the Kinderman-Ramage Method and How to Fix It [1]

Günter Tirler [a], Peter Dalgaard [b], Wolfgang Hörmann [a,c], Josef Leydold [a,*]

[a] *University of Economics and Business Administration, Department for Applied Statistics and Data Processing, Augasse 2-6, A-1090 Vienna, Austria*

[b] *Department of Biostatistics, University of Copenhagen, Blegdamsvej 3, DK-2200 Copenhagen N, Denmark*

[c] *IE Department, Boğaziçi University Istanbul, 80815 Bebek-Istanbul, Turkey*

Abstract

An error in the Gaussian random variate generator by Kinderman and Ramage is described that results in the generation of random variates with an incorrect distribution. An additional statement that corrects the original algorithm is given.

Key words: Gaussian random variate generation
1991 MSC: 65C10

1 Introduction

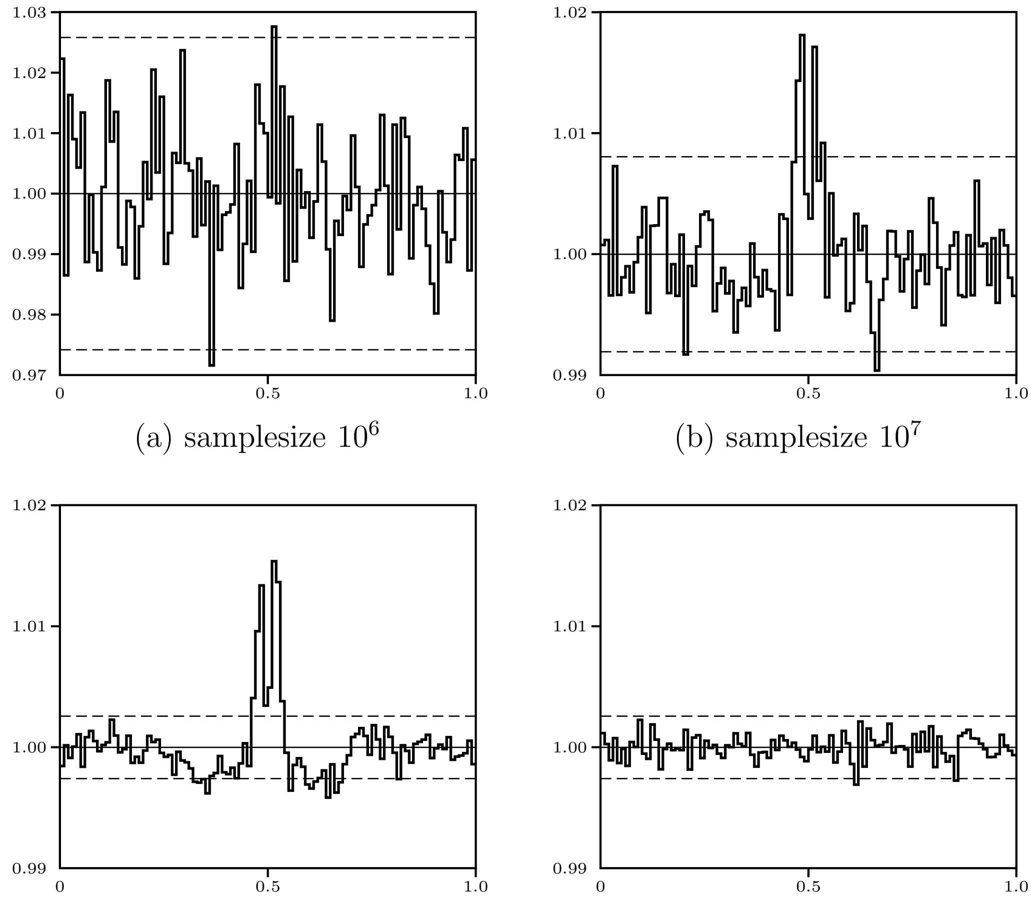
Random variate generation plays a crucial role in every stochastic simulation. This is in particular true for the Gaussian distribution. Thus the availability of very fast and exact generation methods is essential for many applications. Kinderman and Ramage [1] suggested such a fast Gaussian random variate generator (algorithm KR in the sequel). It is sometimes referred as "the fastest Gaussian variate generator" (which is not true any more as faster new methods have been proposed meanwhile, see e.g. [2, 3, 4]) and is included in some libraries for numerical and statistical computing, e.g. the IMSL library [5,

* Corresponding author. Tel +43 1 313 36–4695. FAX +43 1 313 36–738
Email address: `Josef.Leydold@statistik.wu-wien.ac.at` (Josef Leydold).

1 This work was supported by the Austrian Science Foundation (FWF), project no. P12805-MAT

routine `RNNOA`], `R` [6], or UNU.RAN [7], and in statistical programs, e.g. SPSS [8] or GAUSS [9, routine `rndKMn`]. However, when we tested this generator with `R` we encountered some deviations from the expected distribution near zero. We generated 10^8 normal variates and transformed them with the cumulative distribution function of the normal distribution. These transformed variates should follow a uniform distribution but on drawing a histogram we observed clear deviations from uniformity, see Fig. 1. As can be seen, this deviation is unlikely to be detected by a global test (e.g., χ^2-test for goodness of fit) if the sample size is less than 10^7. Nevertheless, a simulation that is sensitive to deviations from the normal distribution near the mode could produce wrong results even for much smaller sample sizes.

In this contribution we give a short description of the algorithm, analyze the reason for the observed error and propose a correction for it. For a more de-



(a) samplesize 10^6 (b) samplesize 10^7

(c) samplesize 10^8 (d) corrected algorithm, 10^8

Fig. 1. Histograms with 100 bins for a sample of Gaussian random variates generated by the algorithm of Kinderman and Ramage [1] (a–c) and by the corrected algorithm (d) for different sample sizes. The generated sequences have been transformed the cumulative distribution function. The expected relative frequency for each bin is 1% (solid horizontal line), the 99% confidence interval is shown by the dashed lines.

tailed introduction to the principles of non-uniform random variate generation see e.g. [10, 11].

2 The algorithm by Kinderman and Ramage

The entire algorithm is presented as Algorithm 1. For comparability, we use the same format as in [1] (with a fix for the error). It is based on two main principles for random variate generation: First it utilizes the *composition method*, that is, the density $\varphi(x)$ of the normal distribution is decomposed into a mixture of other densities $f_i(x)$, $\varphi(x) = \sum w_i f_i(x)$, where the weights $(w_1, w_2, \ldots)$ form a probability vector. Generation is then done by sampling one of these densities at random by means of this probability vector. In algorithm KR the normal density $\varphi(x)$ is decomposed into a triangular density and a density that is described by a difference function $f(t)$,

$$f(t) = \varphi(t) - 0.180025191068563 \max(\xi - |t|, 0) , \tag{1}$$

where $\xi = 2.2160358671$, see Fig. 2(a). Sampling from this triangular density is fast and easy by means of the sum of two uniform random numbers (step 1 in Algorithm 1). For the remaining part the *acceptance/rejection method* is used. It requires a density $h(x)$ and a constant α (≥ 1) such that $\alpha h(x) \geq g(x)$, where $g(x)$ denotes the density of the desired distribution. $\alpha h(x)$ is then called a *hat function* for $g(x)$. A random variate X that follows $h(x)$ is generated and accepted if for the generated uniform (0,1) random number U the acceptance

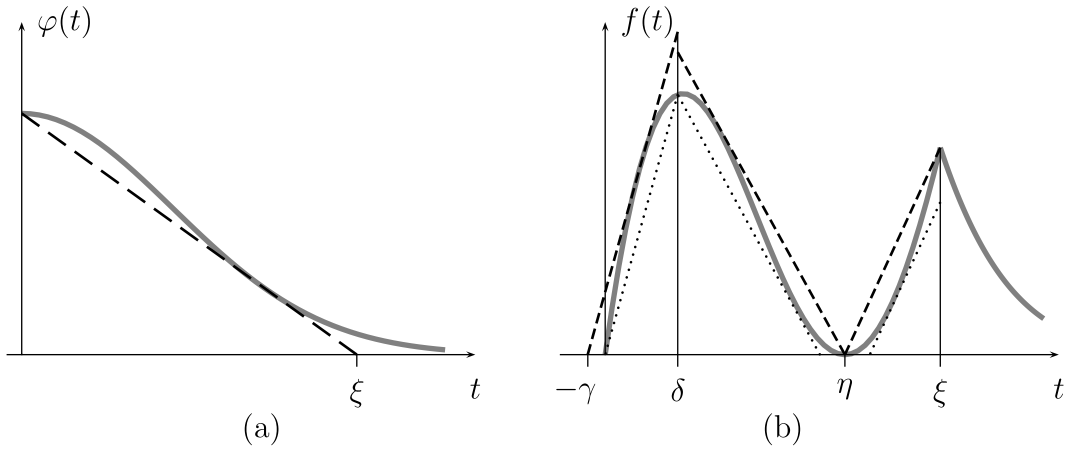


Fig. 2. (a) Decomposition of (half-) normal density $\varphi(x)$ into a triangular density (dashed line) and the difference function $f(t)$.
(b) Decomposition of the difference function $f(t)$ into four parts using the intervals $[0, \delta]$, $[\delta, \eta]$, $[\eta, \xi]$, and $[\xi, \infty)$. For the first three intervals rejection from a triangular hat (dashed line) is used; squeezes are shown as dotted lines.

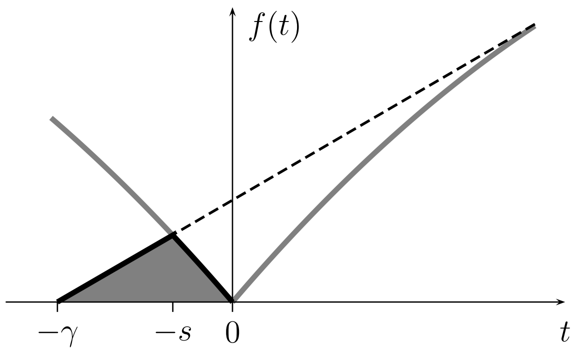


Fig. 3. Region of incorrect acceptance. Random variates less than zero should have been discarded. The coordinate $-s$ is used in the computation of the probability of incorrect acceptance.

condition $U\alpha h(X) \leq g(X)$ is fulfilled; otherwise X is rejected and we have to try again. Since the evaluation of the density $g(x)$ is often slow, simple lower bounds $s(x) \leq g(x)$ (called *squeezes*) can be used to avoid the evaluation of the density in most cases. As the difference function (1) is not easy to handle, its domain is further decomposed into subintervals, see Fig. 2(b). For the first three intervals, $[0, \delta]$, $[\delta, \eta]$, and $[\eta, \xi]$, triangular hats and squeezes are used. Generating random variates from these triangular distributions uses the fact that both the maximum and the minimum of two uniform random numbers have a triangular density. For the tail region $[\xi, \infty)$ rejection from the hat function $x\ \exp((\xi^2 - x^2)/2)$ is used, see [10, p. 381] for details. The four regions correspond to steps 8, 7, 5, and 3, respectively in Algorithm 1. Finally the described algorithm utilizes the symmetry of the normal distribution and samples a variate from the half-normal distribution with domain $[0, \infty)$ which is then multiplied by -1 with probability $1/2$.

3 Incorrect density function near zero

The difference function (1) is partitioned into four parts, where the first part has support $[0, \delta]$ with $\delta = 0.479727404222441$ (Fig. 2(b)). Sampling from this part is done by the acceptance/rejection method in step 8 of Algorithm 1, using the hat function $h(t) = k\,(t + \gamma)/(\delta + \gamma)$, where $k = 0.053377549506886$. The support of the hat function is given by $[-\gamma, \delta]$, where $\gamma = 0.1157797337934990$, and consequently, random variates Z less than zero must always be rejected, see Figs. 2(b) and 3. However, the condition $0.53377549506886\,|z| \leq f(t)$ also holds for some $z < 0$ and thus negative numbers are returned for the half-normal distribution. Hence we have an invalid area of acceptance which is indicated in Fig. 3. As a result the acceptance rate in the intervals $(-\gamma, 0)$ and $(0, \gamma)$ are too high whereas they are too low in the intervals $(-\delta, -\gamma)$ and (γ, δ).

Algorithm 1 Algorithm KR [1] in original notation with added correction (framed part). (u, v, and w are $(0,1)$ uniform random numbers.)

1: Generate u. If $u < .88407\,04022\,98758$, generate v and return $x = \xi(1.13113\,16354\,44180\,u + v - 1)$.
2: If $u < .97331\,09541\,73898$, go to 4.
3: Generate v, w. Set $t = \frac{\xi^2}{2} - \ln w$. If $v^2 t > \frac{\xi^2}{2}$, begin this step again. Otherwise return $x = \sqrt{2t}$ if $u < .98665\,54770\,86949$ or return $x = -\sqrt{2t}$ if not.
4: If $u < .95872\,08247\,90463$ go to 6.
5: Generate v, w. Set $z = v - w$ and $t = \xi - .63083\,48019\,21960 \min(v, w)$. If $\max(v, w) \leq .75559\,15316\,67601$ go to 9. If $.03424\,05037\,50111|z| \leq f(t)$ go to 9. Otherwise, repeat this step 5.
6: If $u < .91131\,27802\,88703$ go to 8
7: Generate v, w. Set $z = v - w$ and $t = .47972\,74042\,22441 + 1.10547\,36610\,22070 \min(v, w)$. If $\max(v, w) \leq .87283\,49766\,71790$, go to 9. If $.04926\,44963\,73128\,|z| \leq f(t)$, go to 9. Otherwise, repeat this step 7.
8: Generate v, w. Set $z = v - w$ and $t = .47972\,74042\,22441 - .59550\,71380\,15940 \min(v, w)$.
   $\boxed{\textbf{If } t < 0 \textbf{ repeat this step 8.}}$
   If $\max(v, w) \leq .80557\,79244\,23817$, go to 9.
   If $.05337\,75495\,06886\,|z| \leq f(t)$, go to 9.
   Otherwise, repeat this step 8.
9: If $z < 0$, return $x = t$; otherwise, return $x = -t$.

To compute the deviation from the correct rate of acceptance notice that step 8 in this algorithm is executed with probability $p_1 = 2\,A$ where $A = \int_0^\delta f(t)\,dt = 0.013621189$. Then the generated Z should fall into the interval $(0, \gamma)$ with probability A_l/A, where $A_l = \int_0^\gamma f(t)\,dt = 0.0011036267$ and into (γ, δ) with probability $1 - A_l/A$. The respective probabilities to sample a point in the intervals $(0, \gamma)$ and (γ, δ) via step 1 (triangular density) are $p_2 = \int_0^\gamma 0.180025191068563(\xi - t)\,dt = 0.0449828$ and $p_3 = 0.125685$. Hence we should find for the probability of a point to fall into the interval $(0, \gamma)$, $p_2 + A\,\frac{A_l}{A} = 0.0460864$, which of course is equal to $\int_0^\gamma \phi(t)\,dt$.

However, the area of invalid acceptance is given by $A_i = \int_{-\gamma}^{-s} h(t)\,dt + \int_{-s}^{0} f(t)\,dt = 0.000397114$ where $s = 0.0396471$ is the intersection of the hat and the difference function $f(|t|)$ for negative values of t (Fig. 3). Thus the generated Z in step 8 of the original algorithm falls into the interval $(0, \gamma)$ with probability $(A_l + A_i)/(A + A_i) = 0.1070558$ (which is larger than the correct value $A_l/A = 0.0810228$) and into (γ, δ) with probability $1 - (A_l + A_i)/(A + A_i)$ in the original algorithm. Thus we find for the probability of a point to fall into the interval $(0, \gamma)$, $p_2 + A\,(A_l + A_i)/(A + A_i) = 0.046441$. Table 1 summarizes the results.

This error, however, can be easily corrected if we add the framed statement in Algorithm 1 which always rejects negative values for z. Figure 1(d) shows that this modification leads to a correct algorithm.

Table 1
Probabilities in algorithm KR ($\gamma = 0.11577973379349904$, $\delta = 0.479727404222441$)

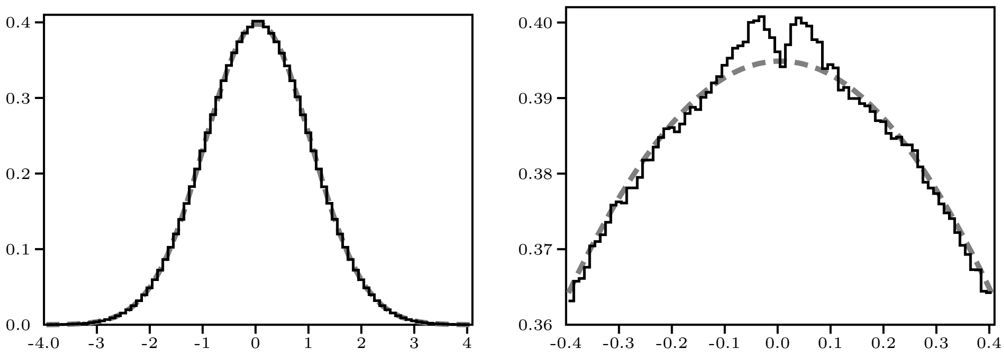
domain	normal distribution	incorrect KR	corrected KR
$(0, \gamma)$	4.6086 %	4.6441 %	4.6086 %
(γ, δ)	13.8203 %	13.7848 %	13.8203 %
(δ, ∞)	31.5711 %	31.5711 %	31.5711 %

4 Detecting deviations from normality

When looking at Table 1 one might ask, how such small deviations from the correct distribution can be detected empirically (or maybe contrary, why has this error not been detected earlier). However, this is not a trivial task. We detected the error *incidently*. The first author was curious about the random variate generators in `R` version 1.6.3. When he was "playing around" with the generator by Kinderman Ramage he found some small deviations from the correct distribution. (Meanwhile this error has been fixed in `R` version 1.7.1.)

It should be noted here that in practice techniques such as the χ^2-test for goodness-of-fit are mainly used for checking the correctness of the implementation rather than for verifying the proposed algorithm. The latter must *always* be done by mathematical methods. Any goodness-of-fit test (such as the χ^2-test, Kolmogorov-Smirnov test or Anderson-Darling test) has only limited power to detect small errors in a generation algorithm. For example, in our experiment with a χ^2-test with a sample of size 10^6, significance level $\alpha = 0.001$ and 100 intervals the null hypothesis was only rejected in 1 out of 100 trials. So the power is extremely small and we have almost no chance to detect the problem of the generator. With sample size 10^7, $\alpha = 0.001$ and 100 intervals we observed 96 rejections of the null hypothesis in 100 trials; this indicates a much better power but still some "luck" is required to find the error. For a sample of size 10^8, $\alpha = 0.001$ and 100 intervals we were able to reject the null hypothesis in all one hundred trials and we are sure to detect the error. Notice, however, that at the time when the paper was published it took more than seven hours only to generate the random numbers of such a sample (according to the timings given in the original paper [1]). Moreover, using such a sample size rather detects the deficiencies of most linear congruential generators (LCG) with period length of about 10^9 that were mainly used as source of uniform pseudo-random numbers at that time. When the deviations from the exact distribution are already known one could construct a test with higher power.

For visual inspection of random variates histograms are a very useful technique. However, deviations cannot be detected when they are small compared to the expected value; see Fig. 4(a) where even for a sample of size 10^8 the



(a) (b)

Fig. 4. Histograms with 100 bins for a sample of 10^8 Gaussian random variates generated by the algorithm of Kinderman and Ramage [1] in the intervals $(-4, 4)$ (a) and $(-0.4, 0.4)$ (b). The normal density function is shown by the dashed lines.

error is not apparent. When we zoom into the critical region the error can be clearly seen, Fig. 4(b).

A better way to visualize deviations from the correct distribution is the technique of transforming the generated random variates by means of the cumulative distribution function. This should result in a random sample of the uniform distribution and thus the bins of a histogram have the same expected probability. Then even outliers where the frequency is only a little bit larger than two times the standard deviation of the frequencies can be clearly seen (Fig. 1). This is also supported by most plotting routines that automatically zoom into the range of the given data. The technique of transforming data with the cumulative distribution function is folklore for people who investigate random variate generators but seems to be little known by many simulation practioners. We express our gratitude to an anonymous referee who drew our attention to this fact. Notice that the results of goodness-of-fit tests are in general not changed by this transform, but for the χ^2-test the transform leads to a simple choice of the intervals. There also exists a lot of empirical tests that have been developed especially for uniform random number generators, see [12, 13]. Transforming the generated variates allows to use most of these tests for nonuniform random variates as well; see e.g. [14].

5 Conclusion

There is an error in Algorithm KR [1]. When using global statistical tests this is only likely to be detected for samples of size at least 10^7 which was very large at the time when this algorithm was published (1976). However, with the improvements in computing power, it has become routine to generate larger samples. Any simulation that is sensitive to deviations from the correct density in the vicinity of zero would then produce results with some bias. This error can easily be fixed with the additional statement in Algorithm 1.

References

[1] A. J. Kinderman, J. G. Ramage, Computer generation of normal random variables, J. Am. Stat. Assoc. 71 (356) (1976) 893–898.

[2] J. H. Ahrens, U. Dieter, Efficient table-free sampling methods for the exponential, cauchy, and normal distributions, Commun. ACM 31 (1988) 1330–1337.

[3] W. Hörmann, G. Derflinger, The ACR method for generating normal random variables, OR Spektrum 12 (3) (1990) 181–185.

[4] G. Marsaglia, W. W. Tsang, The monty python method for generating random variables, ACM Trans. Math. Soft. 24 (3) (1998) 341–350.

[5] IMSL, IMSL Fortran Library User's Guide, STAT/LIBRARY Volume 2, Visual Numerics, Inc., version 5.0, `http://www.vni.com/books/docs/imsl/StatV2.pdf` (1994–2003).

[6] The R project for statistical computing, `http://www.r-project.org/`.

[7] J. Leydold, W. Hörmann, E. Janka, G. Tirler, UNU.RAN – A Library for Non-Uniform Universal Random Variate Generation, Institut für Statistik, WU Wien, A-1090 Wien, Austria, available at `http://statistik.wu-wien.ac.at/unuran/` (2002).

[8] SPSS Inc. Headquarters, 233 S. Wacker Drive, Chicago, Illinois 60606, version 11, `http://www.spss.com/tech/stat/Algorithms.htm`.

[9] GAUSS, Language Reference, Aptech Systems, Inc., 23804 SE Kent-Kangley Road, Maple Valley, WA 98038 USA, version 5.0 `http://www.aptech.com/`.

[10] L. Devroye, Non-Uniform Random Variate Generation, Springer-Verlag, New-York, 1986.

[11] W. Hörmann, J. Leydold, G. Derflinger, Automatic Non-Uniform Random Variate Generation, Springer-Verlag, Berlin Heidelberg, 2004.

[12] D. E. Knuth, The Art of Computer Programming. Vol. 2: Seminumerical Algorithms, 3rd Edition, Addison-Wesley, 1998.

[13] P. L'Ecuyer, Random number generation, in: J. Banks (Ed.), Handbook of Simulation, Wiley, 1998, Ch. 4, pp. 93–137.

[14] J. Leydold, H. Leeb, W. Hörmann, Higher dimensional properties of non-uniform pseudo-random variates, in: H. Niederreiter, J. Spanier (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 341–355.