MATHEMATICS OF COMPUTATION, VOLUME 27, NUMBER 124, OCTOBER 1973

Extensions of Forsythe's Method for Random Sampling from the Normal Distribution*

J. H. Ahrens and U. Dieter

To the Memory of G. E. Forsythe
April 9, 1972

Abstract. This article is an expansion of G. E. Forsythe's paper "Von Neumann's comparison method for random sampling from the normal and other distributions" [5]. It is shown that Forsythe's method for the normal distribution can be adjusted so that the average number $\bar{N}$ of uniform deviates required drops to 2.53947 in spite of a shorter program. In a further series of algorithms, $\bar{N}$ is reduced to values close to 1 at the expense of larger tables. Extensive computational experience is reported which indicates that the new methods compare extremely well with known sampling algorithms for the normal distribution.

1. Introduction. This article was originally intended to be a joint paper of G. E. Forsythe and the authors. It grew out of discussions following a presentation of [2] and [4] at Stanford in October 1971. The latter paper stimulated Forsythe to modify J. von Neumann's classical method for the generation of exponential variables (in [9]). In the resulting Stanford report, a surprisingly general sampling algorithm was presented whose specialization to the normal distribution was then programmed by the authors and subsequently modified. Professor Forsythe was enthusiastic about the alternative versions and proposed to write a comprehensive joint paper. His untimely death has rendered this project impossible. Instead, the Stanford report has been submitted separately (as [5]), and the present paper is dedicated to Forsythe's memory. Its main purpose is to remove all objections to the practical applicability of the method in [5] which in its original form required too many samples from the uniform distribution.

The paper starts with a restatement of Forsythe's generalization of von Neumann's comparison method. A neater proof is given for the validity of the approach. The calculation of the expected number $\bar{N}$ of uniform deviates required is also done in a shorter way. Subsequently, this quantity $\bar{N}$ is considered in more detail for the special case of the normal distribution. It is shown that $\bar{N}$ may be decreased by means of suitable subdivisions of the range $[0, \infty)$.

In the central part (Sections 4 and 5), Forsythe's special algorithm for the normal distribution (called FS) is embedded in a series of sampling procedures which range from the table-free center-tail method (CT) through an improvement of FS (called FT) to algorithms that require longer tables (FL). For the transition from FT to FL, a

Received July 7, 1972.

AMS (MOS) subject classifications (1970). Primary 65C10; Secondary 68A55.

Key words and phrases. Random number generation, normal distribution.

* Research supported by DFG (Deutsche Forschungsgemeinschaft) and N.R.C. (National Research Council of Canada).

couple of new ideas are added to the basic approach. One of these has recently been found independently by Richard Brent.

In Section 6 computational experience is reported including a comparison of CT, FS, FT and the algorithms of type FL with the most efficient previously known sampling methods for the normal distribution. The new procedures compare very well in terms of computation times and memory requirements. In the series FL, it is possible to reduce the number $\bar{N}$ of uniform variates required to any desired value above 1 at the expense of longer tables. As a reasonable compromise between speed, size of $\bar{N}$ and program length, the special algorithm FL_5 may be recommended. It needs 136 tabulated constants for 14-decimal accuracy—they are all listed in two tables. The low value of $\bar{N} = 1.23156$ ensures that it remains a superior method no matter what procedure is used for the generation of the required uniform variates.

2. Forsythe's Method. G. E. Forsythe's generalization of J. von Neumann's method creates random samples in an interval $[a, b)$ from any probability density function of the form

$$f(x) = \alpha e^{-G(x)}, \qquad 0 \leqq G(x) \leqq 1, \quad a \leqq x < b, \tag{1}$$

where α is a constant. The procedure is defined as follows:

Method FO (*Forsythe, sampling from an interval*).

1. *Generate a sample u from the uniform distribution between zero and one. Set $x \leftarrow a + (b - a)u$ (x is uniformly distributed between a and b).*

2. *Calculate $t \leftarrow G(x)$.*

3. *Generate a sequence of independent samples $u_1, u_2, \cdots, u_k$ from the uniform distribution in* [0, 1) *where k is determined by the condition $t \geqq u_1 \geqq \cdots \geqq u_{k-1} < u_k$. (If $t < u_1$, then $k = 1$.) If k is even, reject x and go back to* 1. *If k is odd, return x as a sample from* (1).

The validity of FO will follow easily from a simple lemma.

LEMMA. *Let t be a given number in* [0, 1). *Generate independent* [0, 1)*-uniform variates $u_1, \cdots, u_k$. k is determined by the condition $t \geqq u_1 \geqq \cdots \geqq u_{k-1} < u_k$ ($k = 1$ if $t < u_1$). Then*

(a) *The probability of k being an odd number is $P(t) = e^{-t}$.*

(b) *The expected value of k, no matter whether k is odd or even, amounts to $E(t) = e^t$.*

Proof of (a). When $u_1 \leqq t$, the event that the sequence terminates with k odd and the event that u_1 has an odd number of successors (k even) are mutually exclusive. Hence

$$P(t) + \int_0^t P(u_1)\, du_1 = 1.$$

It follows that $P(0) = 1$ and $P'(t) = -P(t)$ leaving $P(t) = e^{-t}$ as the only possible solution.

Proof of (b). The expected value of k is equal to 1 plus the expected value $E(u_1)$ if $u_1 \leqq t$. Hence,

$$E(t) = 1 + \int_0^t E(u_1)\, du_1.$$

It follows that $E(0) = 1$ and $E'(t) = E(t)$, and finally $E(t) = e^t$.

The Lemma is now applied to the sequence $t = G(x) \geqq u_1 \geqq \cdots \geqq u_{k-1} < u_k$ where $x = a + (b - a)u$ and all the u and u_i are [0, 1)-uniformly distributed. Part (a) yields that the probability of k being odd is

$$P(k \text{ odd}) = \frac{1}{b-a}\int_a^b e^{-G(s)}\,ds.$$

$P(k \text{ odd}) \geqq e^{-1}$ since $G(x) \leqq 1$; hence the process always terminates. The probability density function of x is given by (1) where α is determined as $\alpha^{-1} = \int_a^b e^{-G(s)}\,ds$.

Part (b) of the Lemma allows to calculate the expected number $\bar{N}$ of uniform random numbers $u, u_i\ (i = 1, 2, \cdots)$ needed for the generation of one accepted sample $x = a + (b - a)u$.

$$\bar{N} = P(k \text{ odd})E(k+1, k \text{ odd}) + P(k \text{ even})\{E(k+1, k \text{ even}) + \bar{N}\}$$

or

$$\bar{N} = \frac{E(k+1)}{P(k \text{ odd})} = \frac{1 + E(k)}{P(k \text{ odd})}.$$

$E(k)$ was determined in part (b) of the Lemma. Hence

$$\bar{N} = \frac{1 + (b-a)^{-1}\int_a^b e^{G(s)}\,ds}{(b-a)^{-1}\int_a^b e^{-G(s)}\,ds} = \frac{b - a + \int_a^b e^{G(s)}\,ds}{\int_a^b e^{-G(s)}\,ds}. \tag{2}$$

3. The Normal Distribution. As in [5], the general algorithm FO is applied to the normal distribution. After the initial determination of a random sign, it is sufficient to consider the positive half of the distribution function whose density is

$$\varphi(x) = (2/\pi)^{1/2}e^{-x^2/2}, \qquad 0 \leqq x < \infty.$$

The range $[0, \infty)$ has to be split into finite intervals $[a_i, a_{i+1})$, and the function $G(x)$ is defined as

$$G(x) = \tfrac{1}{2}(x^2 - a_i^2) = (w/2 + a_i)w \quad \text{where } w = x - a_i \text{ for } x \in [a_i, a_{i+1}). \tag{3}$$

The numbers a_{i+1} are subject to the condition $G(x) \leqq 1$ which means

$$a_{i+1}^2 - a_i^2 \leqq 2. \tag{4}$$

It is desirable to keep the expected number $\bar{N}$ of uniform variates required small. $\bar{N}$ is now calculated for a fixed interval $[a, a + d)$ according to (2):

$$\bar{N} = \frac{d + \int_a^{a+d} e^{(x^2-a^2)/2}\,dx}{\int_a^{a+d} e^{-(x^2-a^2)/2}\,dx} = \frac{d + \int_0^d e^{(w/2+a)w}\,dw}{\int_0^d e^{-(w/2+a)w}\,dw}. \tag{5}$$

$G(x)$ is monotonic and attains its maximum at $x = a + d$, that is $w = d$. Replacing the exponential functions by their Taylor series yields

$$\bar{N} = \frac{1 + d^{-1}\int_0^d \{1 + aw + \frac{1}{2}w^2 + \frac{1}{2}(aw + \frac{1}{2}w^2)^2\}\, dw + o(d^2)}{d^{-1}\int_0^d \{1 - aw - \frac{1}{2}w^2 + \frac{1}{2}(aw + \frac{1}{2}w^2)^2\}\, dw + o(d^2)}$$

$$= \frac{2 + \frac{1}{2}ad + \frac{1}{6}(1 + a^2)d^2 + o(d^2)}{1 - \frac{1}{2}ad - \frac{1}{6}(1 - a^2)d^2 + o(d^2)},$$

or

$$\bar{N} = 2 + \tfrac{3}{2}ad + d^2(\tfrac{1}{2} + \tfrac{7}{12}a^2) + o(d^2). \tag{6}$$

Consequently, $\bar{N}$ will be close to 2 if d is small.

If d attains its maximum $d = -a + (a^2 + 2)^{1/2} \approx 1/a$ which is determined by the condition $2ad + d^2 = 2$ (see (4)), then $\bar{N} = e/(1 - e^{-1}) + o(d) = 4.30026 + o(d)$. This follows also from Eq. (5):

$$\bar{N} = \frac{1 + \int_0^1 e^{(u^2-u)d^2/2}e^u\, du}{\int_0^1 e^{-(u^2-u)d^2/2}e^{-u}\, du} = \frac{1 + \int_0^1 e^u\, du + o(d)}{\int_0^1 e^{-u}\, du + o(d)}.$$

Hence, a large value of d has to be avoided.

4. Existing Algorithms for the Normal Distribution.

Algorithm CT (*Center-Tail Method, Dieter/Ahrens*). The formal description is given in [4]. Here it is merely pointed out that the 'center' part of CT is *precisely* the general algorithm FO applied to the interval $[0, \sqrt{2})$. According to (4), $a_2 = \sqrt{2}$ is in fact the maximum interval size for $a_1 = 0$ covering 84.27% of the total area under the normal curve. The 'tail' $x \geqq \sqrt{2}$ is handled by a totally different approach that goes back to an idea of G. Marsaglia [7].

The center-tail algorithm is table-free and relatively short. On the other hand, the expected number $\bar{V}$ of uniform variates required is rather large: $\bar{V} = 4.87889$ per completed sample.

The second sampling procedure is Forsythe's original method.

Algorithm FS. The formal description is given in [5]. The method is based on the split of the range $0 \leqq x < \infty$ into the intervals $[0, 1), [1, \sqrt{3}), [\sqrt{3}, \sqrt{5}), \cdots$. It requires three tables: (i) the boundaries $0, 1, \sqrt{3}, \sqrt{5}, \cdots$ for the quantity a in step 1 of FO; (ii) the differences $1 - 0, \sqrt{3} - 1, \sqrt{5} - \sqrt{3}, \cdots$ which are the interval lengths $b - a$ in step 1; and (iii) the cumulative probabilities

$$r_i = \int_0^{(2i-1)^{1/2}} (2/\pi)^{1/2}e^{-x^2/2}\, dx, \qquad i = 1, 2, \cdots. \tag{7}$$

The r_i are needed for the initial determination of the interval ($[0, 1)$ or $[(2i - 1)^{1/2}, (2i + 1)^{1/2})$, $i \geqq 1$) to which the general method FO is applied.

The authors' test program for a CDC 6400 system contained only the tables (i) and (iii) (forming the differences of (ii) from the square roots in (i) did not result in any measurable decrease in speed). The high accuracy (14 digits) of the machine required 32 entries in both (i) and (iii) (an IBM 360/370 computer would be satisfied with 16 entries each).

The expected number $\bar{N}$ of uniform variates per sample as stated in [5] is $\bar{N} = 4.03585$. This was verified on the basis of (2).

5. New Algorithms for the Normal Distribution. The remaining two algorithms are improvements on FS. A different choice of intervals is all that is required both to shorten the computer program and to increase its speed. For this, the consecutive intervals $[a, b)$ are determined such that the areas under $\varphi(x)$ belonging to these intervals become $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$, etc. In other words, if

$$\Phi(x) = \int_x^\infty (2/\pi)^{1/2} e^{-z^2/2}\, dz$$

then the interval boundaries are now defined as

$$a_1' = 0, \qquad a_2' = \Phi^{-1}(\tfrac{1}{2}), \qquad a_3' = \Phi^{-1}(\tfrac{1}{4}), \cdots, a_{i+1}' = \Phi^{-1}(2^{-i}), \cdots .$$

It has to be shown that this special selection is legitimate by verifying condition (4). An accurate proof may be based on the asymptotic expansion

$$\Phi(x) \sim \left(\frac{2}{\pi}\right)^{1/2} \frac{1}{x} e^{-x^2/2}\left\{1 - \frac{1}{x^2} + \frac{1\cdot 3}{x^4} - \frac{1\cdot 3\cdot 5}{x^6} + \cdots\right\} \tag{8}$$

in which the error is always smaller in absolute value than the first neglected term. The tedious but straightforward precise bounding is left to the reader.

In the following formal description of the new algorithm (FT), it is assumed that the interval lengths

$$d_i = a_{i+1}' - a_i' = \Phi^{-1}(2^{-i}) - \Phi^{-1}(2^{-i+1}) \tag{9}$$

are tabulated. These are the only constants required, they are listed as Table 1 in this section. In order to achieve the full accuracy of the computer in hand, i must run from 1 to $B - 1$ where B is the number of bits used by the uniform random number generator ($B = 48$ for a CDC 6400 computer). (In the description below, 'generate u' (or u^*) indicates taking a sample from the uniform distribution in [0, 1); in the case of several generations, the samples are supposed to be independent.)

Algorithm FT (*Forsythe, modified*).

1. Generate u. Store the first bit of u as a sign s ($s = 0$ if $u < \frac{1}{2}$, $s = 1$ if $u \geqq \frac{1}{2}$). Left-shift u by 1 bit ($u \leftarrow 2u - s$). Initialize $i \leftarrow 1$ and $a \leftarrow 0$.
2. Left-shift u by 1 bit ($u \leftarrow 2u$). If $u \geqq 1$ go to 4.
3. Set $a \leftarrow a + d_i$ and then $i \leftarrow i + 1$. Go back to 2.
4. Set $u \leftarrow u - 1$. (Now u is again uniformly distributed in [0, 1).)
5. Set $w \leftarrow ud_i$ and $t \leftarrow (w/2 + a)w$ (cf. (3)).
6. Generate u^*. If $u^* > t$ go to 9 (acceptance).
7. Generate u. If $u^* < u$ generate u and go to 5 (rejection).
8. Set $t \leftarrow u$ and go to 6 (continue the sequence $\{u_i\}$ in step 3 of FO).
9. Set $y \leftarrow a + w$. If $s = 0$ return $x \leftarrow y$; if $s = 1$ return $x \leftarrow -y$.

According to the final performance table in Section 6, the new algorithm FT seems to be better than FS in all respects. FS needs one more uniform deviate than FT for the determination of the interval $[a_i, a_{i+1})$. In FT this is done by the initial shifting mechanism: note that the probability of $u \geqq \frac{1}{2}$ in step 2 is exactly 2^{-i} as required.

Although the determination of the sign s and subsequent shifts deprive the variable u of two or more of its front bits, this does not constitute a noticeable loss of accuracy. The information gained (sign and interval) is equal to the number of bits expanded in steps 1 and 2. Only the one bit which is shifted out in step 4 is really lost.

The relatively smaller intervals in FT further reduce the expected number $\bar{N}$ of uniform variates per sample which is obviously

$$\bar{N} = \sum_{i=1}^{\infty} 2^{-i} \bar{N}_i \tag{10}$$

where the $\bar{N}_1$ are the expected numbers of uniform deviates for the individual intervals $[a_i', a_i' + d_i)$ according to formula (5). The result is $\bar{N} = 2.53947$ as the final value for algorithm FT.

FT is shorter than FS in terms of memory requirements, but for an even faster algorithm, the program length has to be increased again. The last method (FL) is an attempt to improve speed and minimize $\bar{N}$ at the expense of relatively long tables inside the program. It is developed from FT in two steps.

Step I. The subdivision of the range $0 \leqq x < \infty$ into intervals is made finer. A small number m (typically $m = 5$) is chosen, and the partial areas under the density function are adjusted to $2^{-m}, 2^{-m}, \cdots, 2^{-m}, 2^{-m-1}, 2^{-m-2}, \cdots$ ($2^m - 1$ times areas of size 2^{-m} in the 'center', plus decreasing areas as in FT for the 'tail'). For instance, for $m = 5$ the interval boundaries a_i are now $a_1 = 0$, $a_2 = \Phi^{-1}(31/32)$, $a_3 = \Phi^{-1}(30/32)$, $\cdots$, $a_{32} = \Phi^{-1}(1/32)$, $a_6' = a_{32}$, $a_7' = \Phi^{-1}(1/64)$, $a_8' = \Phi^{-1}(1/128)$, $\cdots$. The finer split increases the probability that acceptance in FO is rapid (that $k = 1$). However, no matter how small the intervals, FO still requires at least two uniform deviates (cf. (6)). In order to decrease $\bar{N}$ below 2, a second idea is needed:

Step II. If the maximum possible value $t_{\max}$ of t in step 5 of FT is small then x will usually be accepted as soon as step 6 is used for the first time (case $k = 1$). In particular, if u^* in step 6 is greater than $t_{\max}$, acceptance does not depend on the size of w. This allows us to dispense with u^* with a probability of $1 - t_{\max}$ by modifying the general method FO as follows:

Method FO* (*modified sampling from an interval*).

1. Generate u^*. Let $t_{\max}$ be the maximum of $G(x)$ in $[a, b)$.

2. If $u^* > t_{\max}$ return $x \leftarrow a + (b - a)(u^* - t_{\max})/(1 - t_{\max})$ and exit. (Given that $u^* > t_{\max}$ this x is again uniformly distributed between a and b.)

3. If $u^* \leqq t_{\max}$ generate u and set $x \leftarrow a + (b - a)u$ and $t \leftarrow G(x)$.

4. Set $u_1 \leftarrow u^*$ and generate $u_2, \cdots, u_k$. k is determined by the condition $t \geqq u_1 \geqq u_2 \geqq \cdots \geqq u_{k-1} < u_k$. If k is odd return x.

5. If k is even go back to 1.

The above modification was discovered independently by Richard Brent.

In FO* the expected number $\bar{N}$ of uniform variates is reduced from (2) to

$$\bar{N}^* = \frac{(b - a)t_{\max} + \int_a^b e^{G(x)}\, dx}{\int_a^b e^{-G(x)}\, dx} \tag{11}$$

which is easily verified by repeating the argument that led to (2) and observing that one uniform deviate is saved in each trial with probability $1 - t_{\max}$. As a consequence,

formula (5) for the normal distribution changes to

$$\bar{N}^* = \frac{d(ad + d^2/2) + \int_0^d e^{(w/2+a)w}\,dw}{\int_0^d e^{-(w/2+a)w}\,dw} \tag{12}$$

since $t_{\max} = ad + d^2/2$. The approximate formula (6) becomes

$$\bar{N}^* = 1 + 2ad + d^2(\tfrac{5}{6} + a^2) + o(d^2) \tag{13}$$

which tends to 1 if d tends to 0.

It remains to build Step I and Step II into an easily programmable algorithm (FL) for the standard normal distribution. In the formal description below, it is assumed that the following quantities are tabulated (the new three types of constants (ii)–(v) are listed in Table 2 for the special case $m = 5$):

(i) The (tail) interval lengths d_i as defined in (9) for $i = m + 1, m + 2, \cdots, B - 1$ (they are contained in Table 1).

TABLE 1

Lengths of Intervals: $d_i = \Phi^{-1}(2^{-i}) - \Phi^{-1}(2^{-i+1})$

i	d_i	i	d_i	i	d_i
1	0.67448975019607	17	0.15040938382813	33	0.10597677198479
2	0.47585963017993	18	0.14590257684509	34	0.10433484129317
3	0.38377116397654	19	0.14177003276856	35	0.10276601206127
4	0.32861132306910	20	0.13796317369537	36	0.10126505151402
5	0.29114282663980	21	0.13444176150074	37	0.09982723448906
6	0.26368432217502	22	0.13117215026483	38	0.09844828202068
7	0.24250845238097	23	0.12812596512583	39	0.09712430874765
8	0.22556744380930	24	0.12527909006226	40	0.09585177768776
9	0.21163416577204	25	0.12261088288608	41	0.09462746119186
10	0.19992426749317	26	0.12010355965651	42	0.09344840710526
11	0.18991075842246	27	0.11774170701949	43	0.09231190933664
12	0.18122518100691	28	0.11551189226063	44	0.09121548217294
13	0.17360140038056	29	0.11340234879117	45	0.09015683778986
14	0.16684190866667	30	0.11140272044119	46	0.08913386650005
15	0.16079672918053	31	0.10950385201710	47	0.08814461935364
16	0.15534971747692	32	0.10769761656476		

TABLE 2

31 *Center Intervals* [*case m* = 5]

i	Boundaries: a_i	Maximal t: t_i	$(b-a)/(1-t_{max})$: h_i
1	0.00000000000000	0.00076738283767	0.03920617164634
2	0.03917608550309	0.00230687039764	0.03932704963665
3	0.07841241273311	0.00386061844387	0.03950999486086
4	0.11776987457909	0.00543845406707	0.03975702679515
5	0.15731068461017	0.00705069876857	0.04007092772490
6	0.19709908429430	0.00870839582019	0.04045532602655
7	0.23720210932878	0.01042356984914	0.04091480886081
8	0.27769043982157	0.01220953194966	0.04145507115859
9	0.31863936396437	0.01408124734637	0.04208311051344
10	0.36012989178957	0.01605578804548	0.04280748137995
11	0.40225006532172	0.01815290075142	0.04363862733472
12	0.44509652498551	0.02039573175398	0.04458931789605
13	0.48877641111466	0.02281176732513	0.04567522779560
14	0.53340970624127	0.02543407332319	0.04691571371696
15	0.57913216225555	0.02830295595118	0.04833486978119
16	0.62609901234641	0.03146822492920	0.04996298427702
17	0.67448975019607	0.03499233438388	0.05183858644724
18	0.72451438349236	0.03895482964836	0.05401138183398
19	0.77642176114792	0.04345878381672	0.05654656186515
20	0.83051087820539	0.04864034918076	0.05953130423884
21	0.88714655901887	0.05468333844273	0.06308488965373
22	0.94678175630104	0.06184222395816	0.06737503494905
23	1.00999016924958	0.07047982761667	0.07264543556657
24	1.07751556704027	0.08113194985866	0.07926471414968
25	1.15034938037600	0.09462443534514	0.08781922325338
26	1.22985875921658	0.11230007889456	0.09930398323927
27	1.31801089730353	0.13649799954975	0.11555994154118
28	1.41779713799625	0.17168856004707	0.14043438342816
29	1.53412054435253	0.22762405488269	0.18361418337460
30	1.67593972277344	0.33049802776911	0.27900163464163
31	1.86273186742164	0.58470309390507	0.70104742502766
32	2.15387469406144	-	-

(ii) The interval boundaries $a_i = \Phi^{-1}((2^m - i + 1)/2^m)$ in the center ($i = 1, 2, \cdots, 2^m$).

(iii) The maximal values $t_i = t_{\max}$ of t in the center intervals. They are calculated as $t_i = ((a_{i+1} - a_i)/2 + a_i)(a_{i+1} - a_i)$ for $i = 1, 2, \cdots, 2^m - 1$.

(iv) Factors of the form $(b - a)/(1 - t_{\max})$ as needed in step 2 of FO*. They are defined as $h_i = (a_{i+1} - a_i)/(1 - t_i)$ for $i = 1, 2, \cdots, 2^m - 1$.

Steps 2–8 below constitute an adaptation of FO* to the 2^{m-1} center intervals. Steps 9–16 merely copy the old method (FT) for the tail since there the modification FO* would be too expensive in terms of table space and not very efficient.

Algorithm FL (*long-table method*).

1. Generate u. Store the first bit of u as a sign s. Left-shift u by 1 bit ($u \leftarrow 2u - s$).
2. Shift u to the left by m bits ($u \leftarrow 2^m u$) and equate i to the integer part of u ($i \leftarrow [u]$). If $i = 0$ go to 9.
3. (Start center.) Set $u^* \leftarrow u - i$ and $a \leftarrow a_i$.
4. If $u^* > t_i$ set $w \leftarrow (u^* - t_i)\, h_i$ and go to 17.
5. Generate u. Set $w \leftarrow u(a_{i+1} - a)$ and calculate $t \leftarrow (w/2 + a)w$.
6. If $u^* > t$ go to 17.
7. Generate u. If $u^* < u$ generate u^* and go to 4.
8. Set $t \leftarrow u$, generate u^* and go to 6.
9. (Start tail.) Set $i \leftarrow m + 1$ and $a \leftarrow a_{2^m}$.
10. Left-shift u by 1 bit ($u \leftarrow 2u$). If $u \geqq 1$ go to 12.
11. Set $a \leftarrow a + d_i$, $i \leftarrow i + 1$ and go to 10.
12. Set $u \leftarrow u - 1$.
13. Set $w \leftarrow ud_i$ and $t \leftarrow (w/2 + a)w$.
14. Generate u^*. If $u^* > t$ go to 17.
15. Generate u. If $u^* < u$ generate u and go to 13.
16. Set $t \leftarrow u$ and go to 14.
17. Set $y \leftarrow a + w$. If $s = 0$ return $x \leftarrow y$, if $s = 1$ return $x \leftarrow -y$.

The expected number $\bar{N}$ of uniform variates in FL is

$$\bar{N} = 2^{-m} \sum_{i=1}^{2^m - 1} \bar{N}_i^* + \sum_{m+1}^{\infty} 2^{-i} \bar{N}_i \tag{14}$$

where the tail-interval averages $\bar{N}_i$ are again calculated according to (5) and the center-interval averages $\bar{N}_i^*$ are determined by (12).

As the number 2^m of center intervals increases, the number $\bar{N}$ of uniform deviates decreases towards 1 (see (13)). The numerical values for the cases $m = 4$, 5, and 6 are listed in the next section.

6. **Comparisons of the Algorithms.** The four algorithms of this article are now compared. All programs were written in CDC 6400 Assembler Code [COMPASS]. The computation times were measured on the basis of 10000 trials using a multiplicative congruential method (period 2^{46}) for generating uniform variates. (For its properties see [1].)

CT	Center Tail	33 words (no table)	80 μsec	$\bar{N} = 4.87889$
FS	Forsythe [5]	87 words (23 + 64 for tables)	71 μsec	$\bar{N} = 4.03585$
FT	Areas 2^{-i}	63 words (16 + 47 for tables)	67 μsec	$\bar{N} = 2.53947$

FL_m Long-Table Algorithm: 2^m is the number of equal areas in the center.

FL_4	($2^m = 16$)	117 words (28 + 89 for tables)	50 μsec	$\bar{N} = 1.38009$
FL_5	($2^m = 32$)	164 words (28 + 136 for tables)	47 μsec	$\bar{N} = 1.23156$
FL_6	($2^m = 64$)	259 words (28 + 231 for tables)	46 μsec	$\bar{N} = 1.13644$

All these methods compare well with older procedures. For instance, generating two normal deviates x_1, x_2, from two independent uniform variates u_1, u_2 according to $x_1 \leftarrow (-2 \ln u_1)^{1/2} \sin(2\pi u_2)$, and $x_2 \leftarrow (-2 \ln u_1)^{1/2} \cos(2\pi u_2)$ (Box-Muller [3]) took 2×247 μsec on the CDC 6400 computer. The so-called modified polar method, as defined in [2], needs 56 words and 90 μsec; it is therefore weaker than the center-tail algorithm and certainly inferior to FT. The three examples of FL compare well with the Taylor series method in [2] which requires 154 words (53 + 101 for tables) and 56 μsec.

The only known method which is a little faster than FL_5 and FL_6 is Marsaglia's rectangle-wedge-tail algorithm (described in [6], [8], and [2]). It was not programmed for the CDC 6400 but, on the basis of experience on another machine, an estimate of 44 ± 2 μsec and 300–400 words can be made. Nevertheless FL remains fully competitive for three reasons.

First of all, FL is easier to describe because it is homogeneous. Second, FL does not suffer from the loss of accuracy ($\sim$8 bits) in the main case of Marsaglia's algorithm. Finally, FL yields a choice of methods and permits to decrease $\bar{N}$ towards 1, whereas the rectangle-wedge-tail algorithm could be modified in this direction only at some further loss of accuracy. Small $\bar{N}$ are important if in the future more complex (and therefore slower) generators for uniform variates are preferred.

Neither the quantity $\bar{N}$ nor the observed computation times and memory requirements on one particular machine can determine the value of a method completely. The reader is referred to [2] for a comparison of some competing algorithms for the normal distribution. There an IBM 360/50 as well as a CDC 6400 computer were used for the trials. The relative performances proved similar in this case, but there will still be machines with different preferences on account of different hardware characteristics. New assessments will certainly be required for future systems which have large-scale parallel arithmetic.

Nevertheless, it can be said at this time that George E. Forsythe's approach to sampling from the normal distribution has produced excellent and accurate methods that range from short medium-speed algorithms to longer and very fast sampling procedures.

Department of Applied Mathematics
Nova Scotia Technical College
Halifax, N. S., Canada

Institut für Mathematische Statistik
Technische Hochschule in Graz
8010 Graz, Austria

1. J. H. AHRENS, U. DIETER & A. GRUBE, "Pseudo-random numbers: A new proposal for the choice of multiplicators," *Computing (Arch. Elektron. Rechnen)*, v. 6, 1970, pp. 121–138. MR 43 #5679.

2. J. H. AHRENS & U. DIETER, "Computer methods for sampling from the exponential and normal distributions," *Comm. Assoc. Comput. Mach.*, v. 15, 1972, pp. 873–881.

3. G. E. Box & M. E. Muller, "A note on the generation of random normal deviates," *Ann. Math. Statist.*, v. 29, 1958, pp. 610–611.

4. U. Dieter & J. H. Ahrens, "A combinatorial method for the generation of normally distributed random numbers," *Computing*, v. 10, 1973.

5. G. E. Forsythe, "Von Neumann's comparison method for random sampling from the normal and other distributions," *Math Comp.*, v. 26, 1972, pp. 817–826.

6. D. E. Knuth, *The Art of Computer Programming*, Vol. 2: *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969. MR **44** #3531.

7. G. Marsaglia, "Generating a variable from the tail of the normal distribution," *Technometrics*, v. 6, 1964, pp. 101–102.

8. M. D. MacLaren, G. Marsaglia & T. A. Bray, "A fast procedure for generating normal random variables," *Comm. Assoc. Comput. Mach.*, v. 7, 1964, pp. 4–10.

9. J. von Neumann, "Various techniques used in connection with random digits," *Collected Works*, Vol. 5, Pergamon Press, New York, 1963, pp. 768–770.