A Uniform Random Number Generator That Is Reproducible, Hardware-Independent, and Fast

C. GARDNER SWAIN* and MARGUERITE S. SWAIN

Department of Chemistry, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Received May 18, 1979

URN is a short program for generation of sequences of 8-digit random numbers (uniformly distributed from 0 to 99 999 999 or to 0.999 999 99) that are reproducible (predetermined by seed numbers). It has a repeat period of over 10^{12} random numbers, it has the novel property that a single Fortran version without any modification yields exactly the same sequences even from computers of widely differing precision, and it is faster than most previous random number generators. Equivalent programs can be coded into programmable pocket calculators and yield exactly the same sequences. It was used in a Monte Carlo evaluation of uncertainties of first-order rate constants in the preceding paper (with P. F. Strong).

INTRODUCTION

Random numbers are widely used as initial values in many kinds of iterative computer calculations, in Monte Carlo calculations, and in the design of experiments.[1]

URN is an acronym for this uniform random numerator. Its principal advantages over other programs that generate random numbers are (1) reproducibility of the sequences that it generates using computers of different precision with a single Fortran program without any modification, and (2) greater speed. When building a random number generator into a program or subprogram that may have widespread use, it is advantageous to incorporate one with this novel first feature, i.e., one that will operate and give numbers that are the same within the desired precision regardless of the hardware and particular Fortran compiler employed.[2] This not only permits work to be repeated and checked, but also removes the restriction to one machine or system, and facilitates comparison of output produced or execution times required by whatever alternative facilities are available. URN also produces random numbers faster than RANDU,[3] URAND,[4] or GGUB,[5] the random number generators that have been most commonly used or recommended in the past for computers.

Most of the commonly used random number generators, including RANDU and URAND, rely on integer overflow or register shifts and are therefore machine-specific.[6] Most programs in which subsequent numbers depend on previous real (floating point) numbers also give variable output from one machine to another because of the problems of binary-decimal interconversion and different precisions of real numbers in different machines: 64 bits (16 decimal digits) in International Business Machines (IBM) 360 or 370 at double precision, and 72 bits (18 decimal digits) for Honeywell (HW) 6180, but only 6–10 decimal digits for most programmable pocket calculators. For example, the random number generator suggested for use with the Hewlett-Packard (HP) 25 or 25C programmable calculator[7]

$$r_n = \text{fractional part of } (\pi + r_{n-1})^5 \qquad (1)$$

generates numbers that are quite different after the first three numbers, depending on whether one uses an IBM 370 computer, an HW 6180 computer, this pocket calculator, or some other calculator.

GGUB[5] avoids accumulation of roundoff errors by use of double-precision calculations coupled with the discarding of less significant digits when obtaining each new random number. However, it uses a different Fortran program for IBM 360 or 370 computers than for HW 6000 series computers because of unequal precisions and other differences. Because it requires double precision, it is not programmable on single precision computers or small programmable desk or pocket calculators. It is relatively slow in execution because all calculations are in double precision and a DMOD function, involving division and multiplication, must be calculated for each number. Furthermore, its incorporation into main programs is restricted by copyright by International Mathematical and Statistical Libraries, Inc. (IMSL). Their stated "basic intent" is to allow only internal use of their programs at one computational site, on one computer type, for members of a paying subscribing organization only. This practice seems unfair, even though perfectly legal, because IMSL appropriated previously published procedures when packaging this program. We therefore aimed for a generator that would be freely available, more versatile, and faster than GGUB.

Speed is important in random number generators used in Monte Carlo type calculations. Our objective of a faster routine than GGUB after compilation by standard Fortran compilers, e.g., IBM G1 or HW Multics, practically excludes the use of division, or more than one multiplication, or use of the MOD (remaindering) function. Powers, square roots, exponentials, logarithms, and trigonometric functions are even more time consuming. On the other hand, addition, subtraction, and "if" statements are so much faster in execution that a combination of up to a dozen of these is faster after such compilation than most one- or two-line generators involving MOD.

To ensure a unique sequence longer than the group of only 10^4 to 10^8 numbers that could be expected if each random number were based on single preceding eight-digit random integer as a seed, we aimed for a repeat period above 10^{12} by use of three preceding such integers to determine each random number. To avoid roundoff errors and produce new integer seeds directly, we rely on integer arithmetic and, if real numbers are needed, we convert only when they are needed and as late as possible.

FORTRAN VERSION

Figure 1 is a Fortran version of URN suitable for any standard Fortran compiler.[8] The logic of URN is extremely simple. If the initial seed numbers M1, M2, and M3 are random, so should be the last eight digits of their sum, except possibly for the last two digits, a deficiency corrected by irregular but frequent additions of 1357. The next two statements limit the range of the new random number (next M3) to that from 0 to 99 999 999. The last statement changes the range to that from 0.0 to 0.999 999 99. If fewer digits are wanted, the numbers can obviously be truncated or rounded at the number of digits desired.

The slowest statement is the last one, labeled 9. In many applications integers are adequate and this floating is not required. In many more, the array and dimension statement

0095-2338/80/1620-0056$01.00/0

```
      DIMENSION SN(10000)
      DATA M1, M2, M3/32007779, 23717810, 52636370/
      DO 9 I=1,10000
      M4=M1+M2+M3
      IF(M2.LT.50000000)M4=M4+1357
      IF(M4.GE.100000000)M4=M4-100000000
      IF(M4.GE.100000000)M4=M4-100000000
      M1=M2
      M2=M3
      M3=M4
    9 SN(I)=1.E-8*M3
```

Figure 1. URN as a standard Fortran IV program.

can be eliminated because the numbers can be used as produced without need for storage. In many, the factor of 10^{-8} can be introduced later, or some other factor is more appropriate. The second index of the DO loop must, of course, be set to the number of numbers needed.

Initial seed numbers (M1, M2, M3) are conveniently incorporated into a data statement. Any other three eight-digit random integers could be used as seeds instead of these three. However, obviously special or less-than-eight-digit seeds should be avoided, because certain sets can cause the earliest numbers generated to be also special: in the worst example, with 0,0,0 as initial values, the first 17 numbers generated have abnormably low values, although subsequent numbers are completely random. If one wants to continue later with only a 1 in 10^{12} chance of repeating a triad and hence the subsequent sequence, one can print out M1, M2, and M3 at the end and use these to replace the values in the data statement. However, for most purposes, it is sufficient to replace only one value, e.g., the initial M3, by any other number from 0 to 99 999 999 inclusive. Thus a new sequence of random numbers can be obtained quite satisfactorily for most applications by replacement of only the initial M3 seed by any other number, e.g., by the last M3 or M4.

The numbers generated appear to be quite random, although they are actually only pseudo-random because they are reproducible from seed numbers. (This reproducibility is often an advantage for repeating and checking work.) Of the first 10 000 numbers generated from the seed numbers in Figure 1, the number of them having a specified value for a specified digit never differs by more than 66 from the expected 1000 for any of the $10 \times 8 = 80$ cases (and the χ^2 of 62.7 for all 80 is not exceptional). The numbers of them in the four ranges 0–0.1, 0.9–1.0, 0–0.5, and 0.5–1.0 are each within 30 of those expected (1000, 1000, 5000, 5000). The means of numbers in these four ranges are each within 0.6% of the expected means (0.05, 0.95, 0.25, 0.75). The means of numbers immediately following them are within 1.4% of the expected 0.50 for each of these four ranges. No significant deviations or biases are indicated by these results. No tendency of the numbers to recur in pairs or triads or any other repeating pattern was detected in any digit. This generator (URN) has proved quite satisfactory in several Monte Carlo applications involving both IBM and HW computers, and it allows exact reproducibility with easy interconvertibility between the two systems. In a possibly more critical (not Monte Carlo) application using many different sets of 102 random numbers for initial values of 51 pairs of interdependent parameters which were iteratively refined by least-squares recalculations, we observed apparently nonrandom bunching of successful convergences followed by unexpectedly long strings of unsuccessful attempts when using GGUB, in contrast to more random distributions of successes with URN.

URN is universally faster than GGUB on standard (multiple-pass) Fortran compilers. The URN program of Figure 1 run on HW 6180 multics requires only 36 μs per number execution time (for 10 000 eight-digit numbers) after compilation on the new (1977) Fortran compiler, vs. 150 μs per number using the version of GGUB specified for Multics (reducible only to 142 μs by optimization). Execution times are less on an IBM 370-168 computer after Fortran G1 compilation, but again URN is faster than Fortran versions of RANDU, URAND, or GGUB.

Table I. URN as a Program for a Programmable Calculator Using Reverse Polish Notation[a]

program line	key entry	program line	key entry
00	none	13	RCL 5
01	RCL 1[b]	14	STO + 3[d]
02	RCL 2	15	RCL 6
03	STO 1	16	RCL 3
04	+	17	$X < Y$
05	RCL 3	18	GTO 22[c]
06	STO 2	19	RCL 6
07	+	20	STO − 3[d]
08	STO 3	21	GTO 15
09	RCL 4	22	RCL 3
10	RCL 1	23	RCL 6
11	$X \geq Y$	24	÷
12	GTO 15[c]	25	GTO 00

[a] E.g., HP 25, 25C, 29C, 19C, 33E, 67 or 97. [b] Storage registers 1 to 6 contain M1, M2, M3, 5E7, 1357, and 1E8 initially. [c] Statement 12 (or 18) is executed only if X, the number just recalled, is as large as (or less than) Y, the number previously recalled. [d] STO 3, STO + 3 and STO − 3 replace, add to, and subtract from the contents of register 3.

Use of URN, without any multiplication or division, in the random trials of a Monte Carlo application is illustrated by the faster version 1 of UNCERT in the preceding paper.[10]

If normally distributed random numbers are desired, they can be generated in several ways from these uniformally distributed random numbers.[4] The simplest method, which is a fairly good approximation and fast enough for most purposes, sums 12 uniform random numbers and then adjusts the mean and standard deviation.[9] With URN, this adjustment involves subtraction of 600 000 000 from the sum of 12 successive M3 values to give a mean of zero, and multiplication by 10^{-8} times the standard deviation desired. These random numbers are used in the reserve version 2 of UNCERT in the preceding paper.[10]

CALCULATOR VERSION

Table I shows an equivalent 25-step program for generating the same eight-digit uniformly distributed random numbers on a small programmable calculator such as an HP 25. This is not just an oddity, but a highly practical program. It is oftentimes quicker and more convenient to generate reproducible random numbers this way by taking a minute to key this program into a pocket calculator if only a few dozen are wanted than to hook up to a large community computer. Before execution, M1, M2, M3, 5×10^7, 1357, and 1×10^8 are inserted initially into storage registers 1–6. Minor modifications of this program are needed for Texas Instruments or other programmable calculators using algebraic notation. With this kind of program, URN can be run on any programmable calculator that allows conditional branching. This branching occurs in statements 11–12 and 17–18 in the program of Table I.

ACKNOWLEDGMENT

This work was supported in part by a research grant from the Donors of the Petroleum Research Fund administered by the American Chemical Society.

REFERENCES AND NOTES

(1) E. B. Wilson, Jr., "An Introduction to Scientific Research", McGraw-Hill, New York, 1952, p 286.
(2) Another reason for our choice of the acronym URN is that urns, used since antiquity for random drawing of different objects, colors, names,

or numbers, likewise furnished hardware- and software-independent sequences.
(3) International Business Machines Corp., White Plains, N.Y., "System/360 Scientific, Subroutine Package, Version III, Programmer's Manual" (SSP), Program No. 360A-CM-03X, GH20-0205-4, 5th ed, 1970, subroutines RANDU and GAUSS on p 77.
(4) G. E. Forsythe, M. A. Malcolm, and C. B. Moler, "Computer Methods for Mathematical Computations", Prentice-Hall, Englewood Cliffs, N.J., 1977, random numbers and URAND in Chapter 10.
(5) International Mathematical and Statistical Libraries, Inc., Houston, Texas, "IMSL Library 1", 6th ed for IBM S/370-360 and Xerox Sigma users, July 1977, subroutine GGUB (ISEED, N, R) dated Dec 1976 (reference manual description dated Nov 1975); "IMSL Library 2", 6th ed for Honeywell, Univac and DEC users, July 1977, subroutine GGUB (ISEED, N, R) dated Jan 1974 (reference manual description dated Nov 1975).
(6) Association for Computing Machinery, special supplement on random number generators, *Sigsim Simuletter*, **8**, No. 1, 10–37, 79–91 (1976).
(7) Hewlett-Packard Co., Cupertino, Calif., "HP-25 Applications Programs", 00025-90011, rev. E, July 1976, pp 116–117.
(8) Suitable for any computer for which the range of positive integers equals or exceeds 2^{29} bits or 9 decimal digits.
(9) R. W. Hamming, "Numerical Methods for Scientists and Engineers", McGraw-Hill, New York, 1962, pp 34, 389; and ref 3, subroutine GAUSS.
(10) C. G. Swain, M. S. Swain, and P. F. Strong, preceding paper in this issue.

NEWS AND NOTES

BOOK REVIEWS

How to Write and Publish a Scientific Paper. By R. A. DAY. ISI Press, 3501 Market St., Philadelphia, PA. 1979. viii + 160 pp. $15.00 (clothbound); $8.95 (paperback).

As scientists, our thoughts are mere daydreams and our experiments are but pleasant diversions until we communicate the results of our work. Writing, e.g., by letter, report, or journal article, is still the traditional mode. Because writing is one of our most difficult tasks, we seek the vade mecum to teach us in ten easy lessons or chapters how to write quickly and effectively. Unfortunately, there is no unique formula nor magic wand that can materialize us into superior, or good, or even fair writers.

After 12 years of public school, four years of college, and three to five years of graduate work we are about as good at writing as we can be. Most of us reach our peak in high school. I do not think, however, the peak is fixed for all. It is not providing we are motivated to become better, but unlike Walter Mitty, our motivation has to be implemented with serious and analytical reading and writing, writing, and writing, not dreaming about it, nor just reading "how to" books. If there is a formula, I submit it is constant reading and writing, not one or the other, but both.

People who write well have read and do read articles and books on "how to write" and on communications in general, and most are well aware of the better books, e.g., "Elements of Style" by Strunk and White, "Effective Writing for Engineers, Managers, Scientists" by Tichy, and the ACS "Handbook for Authors". None is listed in R. A. Day's references (pp 153–4).

I enjoyed reading Day's book, and I think it is worth having around, but not without the above three, and others I could list, being in the library. The basic weakness of the book is its orientation to the biological sciences. Editors of chemical journals do not insist on a stylized format, certainly not a detailed Materials and Methods section following the Introduction. We generally write this first but place it last in the paper. We also generally think in terms of Title, Abstract, Introduction, Discussion of Results, Conclusions, Experimental Section, References, and Appendix. The order is determined by the work itself and the best way to communicate it, not by prescribed order.

As the majority of chemists, who are in the industrial sector, are concerned with writing reports, it is unfortunate that Day neglected this area. Academic chemists probably write more reports now than journal articles in connection with government grants and contracts. Instead of "publish or perish", the keynote is "get grants or perish". Some day we may progress to "quality not quantity".

Herman Skolnik

NEWS ITEMS

Gordon Conference on Scientific Information

The Gordon Research Conference on "Scientific Information Problems in Research" has been approved for 1980. The conference will be held the week of July 14–18 at Plymouth State College in New Hampshire.

The conference theme will be centered on current societal problems that require both the integration of concepts from chemistry, biology, geology, etc., and reliable information on a timely basis, to solve the problems. Each day's problems will be developed around a major technological problem such as acid rain, ozone, or fuel from coal and shale oil. The morning speakers will be researchers who have had a significant role in the research problem, and evening speakers will be people involved in the development and operation of an information program related to the technical discussion of the morning session.

Martha E. Williams, University of Illinois in Urbana, and John Murdock, Informatics, Inc., Rockville, Md., would like to hear suggestions from scientists who have been involved in such programs.

CAS Reorganization

Chemical Abstracts Service has reorganized its operations into four major units reporting to Director Dale B. Baker.

Editorial, bibliographic support, and production operations have been brought together under Dr. Russell J. Rowlett, Jr., who continues to be editor and assumes the additional title of director of publications and services.

Finance, personnel, and administrative support functions have been combined in a new business administration division with Glen D. Wallace as director.

John C. Dean, director of marketing, has assumed additional responsibilities for public relations and government and commerical contract activities.

Dr. Ronald L. Wigington, director of research and development, is responsible for research activities and for the development and implementation of new computer-based systems and processes.

Decision-making authority on all operational and policy issues has been centralized in a management board chaired by Director Baker and consisting of Rowlett, Wallace, Dean,