Remark AS R58: A Remark on Algorithm AS 183. An Efficient and Portable Pseudo-Random Number Generator

A. Ian McLeod

Applied Statistics, Volume 34, Issue 2 (1985), 198-200.

Stable URL:
http://links.jstor.org/sici?sici=0035-9254%281985%2934%3A2%3C198%3ARARARO%3E2.0.CO%3B2-I

```
      DO 360 II = 1, N
      DO 360 JJ = 1, N
  360 A(II, JJ) = C(II, JJ)
      A(I, I) = CS * CS * CII + SN * SN * CJJ + TWO * CS * SN * CIJ
      IF(ABS(A(I, I) - ONE)  .GT. EPS) GOTO 30
      M(NN, 1) = I
      NK = NK + 1
      A(J, J) = CS * CS * CJJ + SN * SN * CII - TWO * CS * SN * CIJ
      A(J, I) = (CS * CS - SN * SN) * CIJ + CS * SN * (CJJ - CII)
      A(I, J) = A(J, I)
      DO 370 L = 1, N
      IF(L .EQ. I .OR. L .EQ. J) GOTO 370
      A(I, L) = CS * C(I, L) + SN * C(J, L)
      A(J, L) = -SN * C(I, L) + CS * C(J, L)
      A(L, I) = A(I, L)
      A(L, J) = A(J, L)
  370 CONTINUE
      DO 380 II = 1, N
      DO 380 JJ = 1, N
  380 C(II, JJ) = A(II, JJ)
  390 SUM = SUM + C(I, I)
  400 CONTINUE
C
C        SET THE REMAINING DIAGONAL ELEMENT TO BE (N-SUM) AND
C        SEE IF IT IS WITHIN THE SPECIFIED PRECISION LIMIT
C
      C(N, N) = FLOAT(N) - SUM
      IF(ABS(C(N, N) - ONE)  .GT. EPS) GOTO 30
      RETURN
      END
```

Remark AS R58

A Remark on Algorithm AS 183. An Efficient and Portable Pseudo-random Number Generator

By A. Ian McLeod

The University of Western Ontario, Canada

[Received October 1984. Revised February 1985]

Wichmann and Hill (1982) claim that their generator *RANDOM* will produce uniform pseudo-random variables which are strictly greater than zero and less than one. However, depending on the precision of the machine, some zero values may be produced due to rounding error. For example, in a sequence of 10^9 variables generated by *RANDOM* on a PR1ME-400 Computer starting with initial seeds $IX = IY = IZ = 1$, it was found that 364 of them were exactly 0.0 while the remainder were in the open interval (0, 1) as required. We will now show that this behaviour is explained by the fact that the PR1ME-400 uses chopped arithmetic with only 23 bits for the representation of the fractional part of a real variable (another 8 bits are used for the exponent and one more bit is used for the sign). See Golub and Van Loan (1983, p. 33) for a precise definition of the above terminology.

Present address: Department of Statistical and Actuarial Sciences, The University of Western Ontario, London, Canada N6A 5B9

0035–9254/85/34000 $2.00

In *RANDOM*, the sum of the standardized output from three prime modulus generators is calculated. Let us call this quantity *SUM*. Now, $p_x^{-1} + p_y^{-1} + p_z^{-1} > 2^{-23}$, where $p_x = 30269$, $p_y = 30307$ and $p_z = 30323$ are the moduli. Thus the value of *SUM* will never be 0.0 or 3.0 on any computer which has at least 23 bits for the fractional part. The difficulty arises when the value of the sum is very close to 1.0 or 2.0.

Suppose that U_x, U_y and U_z are independent uniform (0, 1) random variables and let $S = U_x + U_y + U_z$. Then the probability that S has a floating point representation exactly equal to 1.0 or 2.0 if only 23 bits are used for the fractional part and chopped as opposed to rounded arithmetic is used is respectively $p_1 = P\{1 < S < 1 + 2^{-22}\}$ and $p_2 = P\{2 < S < 2 + 2^{-21}\}$. Using the distribution of S given by Johnson and Kotz (1970, p. 64, equation 19), $p_1 = 1.19209 \times 10^{-7}$ and $p_2 = 2.38418 \times 10^{-7}$. Thus it may be expected that *RANDOM* will produce zero values about once in every $(p_1 + p_2)^{-1} = 2.8 \times 10^6$ calls on average on the PR1ME-400. The function *RANDOM* was slightly modified so that the value of *SUM* was assigned and this modified function was called 10^9 times starting with initial values $IX = IY = IZ = 1$. The results, summarized in Table 1, are in good agreement with the hypothetical probabilities p_1 and p_2. These results, which produce a value of the usual χ^2 goodness-of-fit statistic of 1.73 on 2 d.f., also provide an amusing additional test of the generator.

TABLE 1
Observed and expected number of times that the value of SUM is 1.0 *or* 2.0 *in* 10^9 *calls*

SUM	*Observed*	*Expected*
1.0	110	119.2
2.0	254	238.4

It is of interest that some other 32-bit computers, VAX and IBM for example, use rounded arithmetic with 24 bits for the fractional part of a single-precision variable. In this case,

$$p_1 = P\{1 - 2^{-25} < S < 1 + 2^{-24}\} = 4.5 \times 10^{-8}$$

and

$$p_2 = P\{2 - 2^{-24} < S < 2 + 2^{-23}\} = 8.9 \times 10^{-8}.$$

So a zero value might be expected to occur only about once in every 7.4×10^6 calls. Note also that, in this case about one-third of the zero values produced will correspond to values which should in fact be very close to one, whereas in the previous situation with chopped arithmetic, all zero values correspond to values which were in fact very close to zero.

In some situations the zero values possibly produced by *RANDOM* could cause program errors. A naive remedy would then be to discard such values. However, it is desirable that the same random sequence should be generated on any computer and so in order to maintain portability the following amendment to *RANDOM* is proposed:

insert the statements

```
      IF (RANDOM .GT. 0.0)  RETURN
      RANDOM = DMOD(DBLE(FLOAT(IX))/30269.0D0+
     *   DBLE(FLOAT(IY))/30307.0D0+DBLE(FLOAT(IZ))/30323.0D0, 1.0D0)
      IF (RANDOM .GE. 1.0)  RANDOM = 0.999999
```

immediately prior to the *RETURN* statement.

Provided that at least 47 bits are available for the fractional part of a double precision variable, as is the case, for example, on the PR1ME-400, this amendment is guaranteed to produce a result

different from 0.0. To see this, note that a value of *SUM* equal to 1.0 could occur only if there exist integers i_x, i_y and i_z such that $0 < i_x < p_x, 0 < i_y < p_y, 0 < i_z < p_z$ and

$$\left| 1 - \frac{i_x}{p_x} - \frac{i_y}{p_y} - \frac{i_z}{p_z} \right| < 2^{-47}. \tag{1}$$

But (1) holds only if

$$| p_x p_y p_z - i_x p_y p_z - i_y p_x p_z - i_z p_x p_y | < 1$$

$$\Leftrightarrow p_x p_y p_z - i_x p_y p_z - i_y p_x p_z - i_z p_x p_y = 0$$

$$\Leftrightarrow p_y p_z (p_x - i_x) = p_x (i_y p_z + i_z p_y).$$

But p_x cannot be a factor of the left-hand side of the last equation. Therefore there are no such integers i_x, i_y and i_z. Similarly *SUM* cannot equal 2.0.

Note that the third statement of the amendment may be needed on computers with rounded arithmetic in order to avoid producing an output value exactly equal to one when the conversion from double to single precision is made. Also in this statement, the largest machine representable single precision floating point number which is less than one could be used instead of 0.999999.

In addition to portability considerations, the proposed amendment will not unnecessarily ignore extreme values in the sequence which would happen if zero values were naively discarded. In hydrological simulation, for example, it is often extreme events which are of interest (see, for example, Askew, Yeh and Hall, 1971).

Finally, it should be noted that for some applications, output values of zero may do no harm apart from a possible slight lack of portability to 32-bit machines with rounded arithmetic. In this case, if portability is not crucial, no amendment would be needed.

Helpful comments from the referee are acknowledged.

References

Askew, A. J., Yeh, W. and Hall, W. A. (1971) A comparative study of critical drought simulation. *Water Resour. Res.*, 7, 52–62.

Golub, G. H. and Van Loan, C. F. (1983) *Matrix Computations*. Baltimore: Johns Hopkins University Press.

Johnson, N. L. and Kotz, S. (1970) *Continuous Univariate Distributions – 2*. New York: Houghton Mifflin.

Wichmann, B. A. and Hill, I. D. (1982) Algorithm AS 183. An efficient and portable pseudo-random number generator. *Appl. Statist.*, **31**, 188–190.