Statistical Testing of Additive Congruential Random Number (ACORN) Generators

Roy Wikramaratna

REAMC Limited (Reservoir Engineering and Applied Mathematics Consultancy)

INTRODUCTION

ACORN generators represent an approach to generating uniformly distributed pseudo-random numbers which is straightforward to implement for arbitrarily large order k and modulus $M=2^{30t}$ (integer t). They give long period sequences which can be proven theoretically to approximate to uniformity in up to k dimensions.

ACORN GENERATOR

The ACORN pseudo-random number generator was first discovered in the mid-1980s and published in 1989 [1].

Let k be a finite, strictly positive integer. The k -th order Additive Congruential Random Number (ACORN) generator is defined from an integer modulus M, an integer seed ${Y^0}_0$ satisfying $0 < {Y^0}_0 < M$ and an arbitrary set of k integer initial values ${Y^m}_0$, $m = 1, ..., k$, each satisfying $0 \le {Y^m}_0 < M$ by the equations

$${Y^0}_n = {Y^0}_{n-1} \quad n \ge 1 \qquad (1)$$

$${Y^m}_n = [{Y^{m-1}}_n + {Y^m}_{n-1}]_{\mathrm{mod}M} \quad n \ge 1, m = 1, \dots, k \qquad (2)$$

where $[Y]_{\mathrm{mod}\,M}$ means the remainder on dividing Y by M.

Finally, the sequence of numbers ${Y^k}_n$ can be normalised to the unit interval by dividing by M

$${X^k}_n = [{Y^k}_n/M] \quad n \ge 1 \qquad (3)$$

It turns out [2, 3, 4, 5] that the numbers ${X^k}_n$ defined by equations (1) - (3) approximate to being uniformly distributed on the unit interval in up to k dimensions, provided a few simple constraints on the initial parameter values are satisfied

- modulus M needs to be a large integer (typically a prime power, with powers of 2 offering the most straightforward implementation); increasing modulus leads to improved statistical performance
- seed ${Y^0}_0$ and modulus chosen to be relatively prime (which means that their greatest common divisor is 1; for M a power of two this requires only that the seed is odd)
- initial values ${Y^m}_0$, $m = 1, ..., k$ can be chosen arbitrarily

The period length of resulting ACORN sequence can be shown to be a multiple of the modulus.

IMPLEMENTATION

The ACORN generator is straightforward to implement in a few tens of lines in high-level computer languages such as Fortran or C.

The following example is in Fortran; with 32-bit integers (as shown) it allows a modulus up to 2^{60}; by using 64-bit integers it would allow modulus up to 2^{120} with minimal modification to the source code.

This is simplest and most easily understood implementation; significantly faster implementation is possible while still producing identical sequences for any specified initialisation. It can be extended to allow larger order by straightforward modifications to the common block.

```
      DOUBLE PRECISION FUNCTION ACORNJ(XDUM)
C
C           ACORN GENERATOR
C           MODULUS =< 2^60, ORDER =< 12
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (MAXORD=12,MAXOP1=MAXORD+1)
      COMMON /IACO2/ KORDEJ
     1     ,MAXJNT,IXV1(MAXOP1),IXV2(MAXOP1)
      DO 7 I=1,KORDEJ
        IXV1(I+1)=(IXV1(I+1)+IXV1(I))
        IXV2(I+1)=(IXV2(I+1)+IXV2(I))
        IF (IXV2(I+1).GE.MAXJNT) THEN
          IXV2(I+1)=IXV2(I+1)-MAXJNT
          IXV1(I+1)=IXV1(I+1)+1
        ENDIF
      IF (IXV1(I+1).GE.MAXJNT)
     1         IXV1(I+1)=IXV1(I+1)-MAXJNT
    7 CONTINUE
      ACORNJ=(DBLE(IXV1(KORDEJ+1))
     1   +DBLE(IXV2(KORDEJ+1))/MAXJNT)/MAXJNT
      RETURN
      END
```

After appropriate initialisation of the common block, each call to the function ACORNJ generates a single variate drawn from a uniform distribution on the unit interval.

The ACORN generator has been used (alongside the widely-used Mersenne Twister algorithm [6] and a number of other algorithms that are based on linear congruential generators) by Numerical Algorithms Group Ltd since the Mark 22 release of their Fortran Numerical Software Libraries [7] and since the Mark 23 release of their C Numerical Software Libraries [8] as one of their standard base methods for generating uniformly distributed pseudo-random numbers.

A version of the ACORN algorithm is also included in the GSLIB geostatistical software library, Deutsch and Journel [9].

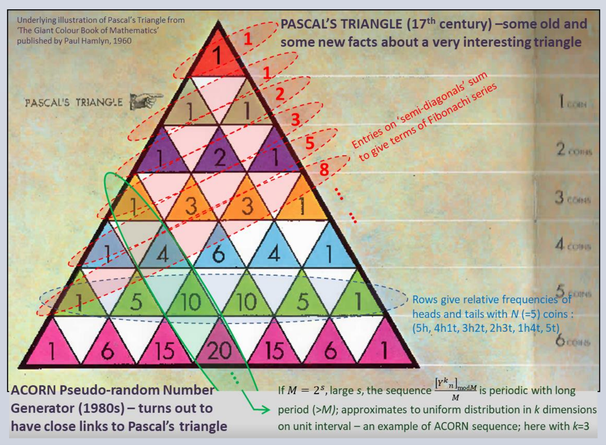
PASCAL'S TRIANGLE AND ACORN SEQUENCES

The ACORN generator turns out to have a close link with Pascal's triangle. Numbering the diagonals from 0 through k, the terms in the k-th diagonal turn out to be a particular special case of a k-th order ACORN sequence.

As a result we can show that the sequence formed by taking the terms in the k-th diagonal modulo M (where M is a large power of 2) and dividing by M is

- a periodic sequence whose period is a multiple of M
- a sequence which approximates to uniform distributed in k dimensions.

This is one example of some fascinating mathematical properties that can be demonstrated or proved for the ACORN sequences. Other examples are included in the references [1, 2, 3, 4, 5].

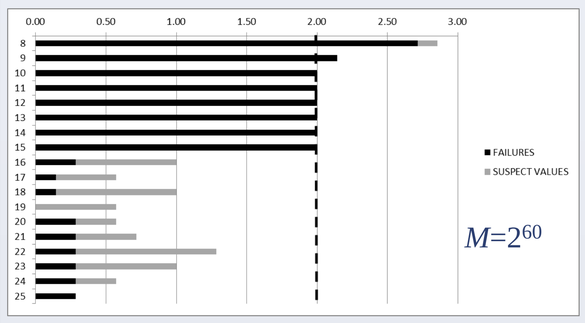


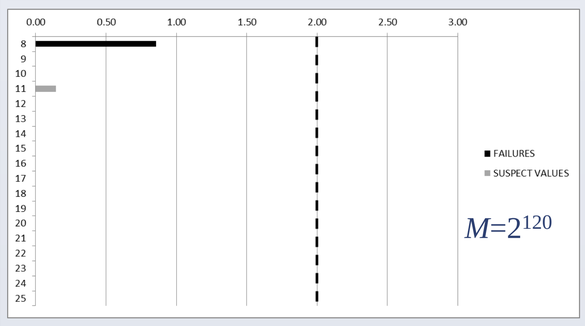
THE TestU01 TEST SUITE

The TestU01 package has been described by L'Ecuyer and Simard [10]. They considered the application of empirical tests of uniformity and randomness to sequences generated by a wide range of algorithms and developed a comprehensive set of empirical tests that were designed to detect undesirable characteristics in such sequences. L'Ecuyer and Simard present results of applying the TestU01 tests to a large number of different sequences, identifying generators that pass all of the tests (collectively called the BigCrush test battery), as well as identifying many generators (including some that are widely used) that have serious deficiencies in respect of certain specific tests.

Results presented below for ACORN generators (which were not included among generators considered by L'Ecuyer and Simard) were obtained using the latest version 1.2.3 of TestU01. The BigCrush battery of tests calculates 180 different test statistics for each sequence that is tested, making use of some 2^{38} pseudo-random numbers from each sequence. We follow L'Ecuyer and Simard in defining a "failure" to be a p-value outside the range $[10^{-10}, 1\text{-}10^{-10}]$ with a "suspect" value falling in one of the ranges $[10^{-10}, 10^{-4}]$ or $[1\text{-}10^{-4}, 1\text{-}10^{-10}]$.

RESULTS AND CONCLUSIONS





Results shown are the average number of 'failures' (black bars) and 'suspect values' (grey bars) obtained with seven different seeds and initialisations (plotted on the x-axis) for ACORN generators with order $8 \le k \le 25$ (plotted on the y-axis) and two values of modulus $M=2^{60}$ and $M=2^{120}$. The only constraint on initialisation for the seven cases was that in each case the seed be a different odd integer less than the modulus. Results of testing show improvement of the overall results with increasing modulus.

Corresponding results obtained for the Mersenne Twister MT19937 generator, are shown by the dotted line on the figures.

With $M=2^{120}$ and $k \ge 9$, ACORN generators passed all the tests for each of the 7 initialisations; since each choice of seed gives a different sequence this potentially gives more than 2^{119} different sequences, each of length at least 2^{120}, which might reasonably be expected to pass all of the tests in these test suites.

With $M=2^{60}$ and $k \ge 9$, ACORN generators failed on average no more than two of the tests across the 7 initialisations tested; with $M=2^{90}$ (not shown in the figures) the performance was intermediate between the two cases shown with no failures and only occasional suspect values.

This contrasts with corresponding results obtained for the widely-used Mersenne Twister MT19937 generator, which consistently failed on two of the tests in the BigCrush test suite.

Further, we assert that an ACORN generator might also reasonably be expected to pass any more demanding tests for p-dimensional uniformity that may be required in the future, simply by choosing $k \ge p$ and modulus $M=2^{30t}$ for sufficiently large t.

REFERENCES

[1] R.S. Wikramaratna, ACORN - A New Method for Generating Sequences of Uniformly Distributed Pseudo-random Numbers, *J. Comput. Phys.*, **83**, pp16-31, 1989.

[2] R.S. Wikramaratna, Theoretical Background for the ACORN Random Number Generator, Report AEA-APS-0244, AEA Technology, Winfrith, Dorset, UK, 1992.

[3] R.S. Wikramaratna, Pseudo-random Number Generation for Parallel Processing – A Splitting Approach, *SIAM News*, **33** number 9, 2000.

[4] R.S. Wikramaratna, The Additive Congruential Random Number Generator – A Special Case of a Multiple Recursive Generator, *J. Comput. and Appl. Mathematics*, **261**, pp371–387, 2008. [doi: 10.1016/j.cam.2007.05.018].

[5] R.S. Wikramaratna, Theoretical and Empirical Convergence Results for Additive Congruential Random Number Generators, *J. Comput. Appl. Math.*, **233**, pp2302-2311, 2010. [doi: 10.1016/j.cam.2009.10.015].

[6] M. Matsumoto and T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Trans. Model. Comput. Simul.*, **8**, 3, 1998.

[7] NAG, Numerical Algorithms Group (NAG) Fortran Library Manual, Mark 22, Numerical Algorithms Group Ltd, Oxford, UK, 2009. (www.nag.co.uk).

[8] NAG, Numerical Algorithms Group (NAG) C Library Manual, Mark 23, Numerical Algorithms Group Ltd, Oxford, UK, 2012. (www.nag.co.uk).

[9] C.V. Deutsch and A.G. Journel, GSLIB: Geostatistics Software Library and User's Guide, Oxford University Press, 384 pp, 1998.

[10] P. L'Ecuyer and R. Simard, TestU01: A C Library for Empirical Testing of Random Number Generators, *ACM Transactions on Mathematical Software*, **33**, 4, Article 22, 2007.

ACKNOWLEDGEMENT AND CONTACT

This work has been carried out with support from REAMC Limited.

Email contact address: rwikramaratna@gmail.com.

Poster presented at the meeting on 'Numerical algorithms for high-performance computational science', 8 – 9 April 2019, The Royal Society, London.