

# The Additive Congruential Random Number (ACORN) Generator - pseudo-random sequences that are well distributed in $k$ -dimensions

Roy Wikramaratna, REAMC Limited

[rwikramaratna@gmail.com](mailto:rwikramaratna@gmail.com)

11<sup>th</sup> June 2019

Oxford University Numerical Analysis Group Seminar

# Outline

- ACORN sequences
- TestU01 - empirical testing of pseudo-random numbers
- Results of testing with TestU01
  - ACORN vs Mersenne Twister MT19937
- Pascal's triangle and ACORN
- Conclusions

# ACORN Generators - background

- ACORN – a method for generating pseudo-random numbers uniformly distributed on the unit interval
  - straightforward to implement for arbitrarily large order  $k$  and modulus  $M=2^{30t}$  (integer  $t$ )
  - long period sequences which can be proven to approximate to uniformity in up to  $k$  dimensions
  - theoretical analysis also supported by extensive empirical testing
- Discovered by RSW in the mid-1980s and first published in 1989 [ref 1].
- This seminar is based on a poster presented at a recent meeting at the Royal Society in April 2019 [ref 2]; have now incorporated some additional results of testing carried out in last few months and drawn some more detailed conclusions.
- See also the website <http://www.acorn.wikramaratna.org/> for more background

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

[2] RSW, Statistical Testing of Additive Congruential Random Number (ACORN) Generators, Meeting on ‘Numerical algorithms for high-performance computational science’, April 2019, The Royal Society, London.

# Definitions

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  $Y^m_0$ ,  $m = 1, \dots, k$ , each satisfying  $0 \leq Y^m_0 < M$  by the equations

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

$$Y^m_n = [Y^{m-1}_n + Y^m_{n-1}]_{\text{mod } M} \quad n \geq 1, m = 1, \dots, k \quad (2)$$

where  $[Y]_{\text{mod } M}$  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  $M$

$$X^k_n = [Y^k_n / M] \quad n \geq 1 \quad (3)$$

It turns out [3, 4, 5, 6] 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  $Y^m_0$ ,  $m = 1, \dots, k$  can be chosen arbitrarily

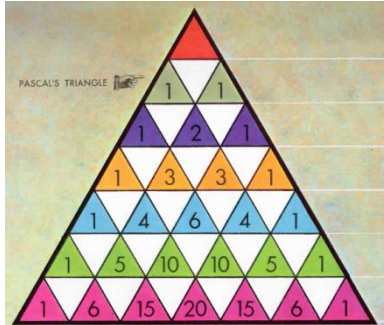
The period length of resulting ACORN sequence can be shown to be a multiple of the modulus.

References [3], [4], [5], [6] – see next slide

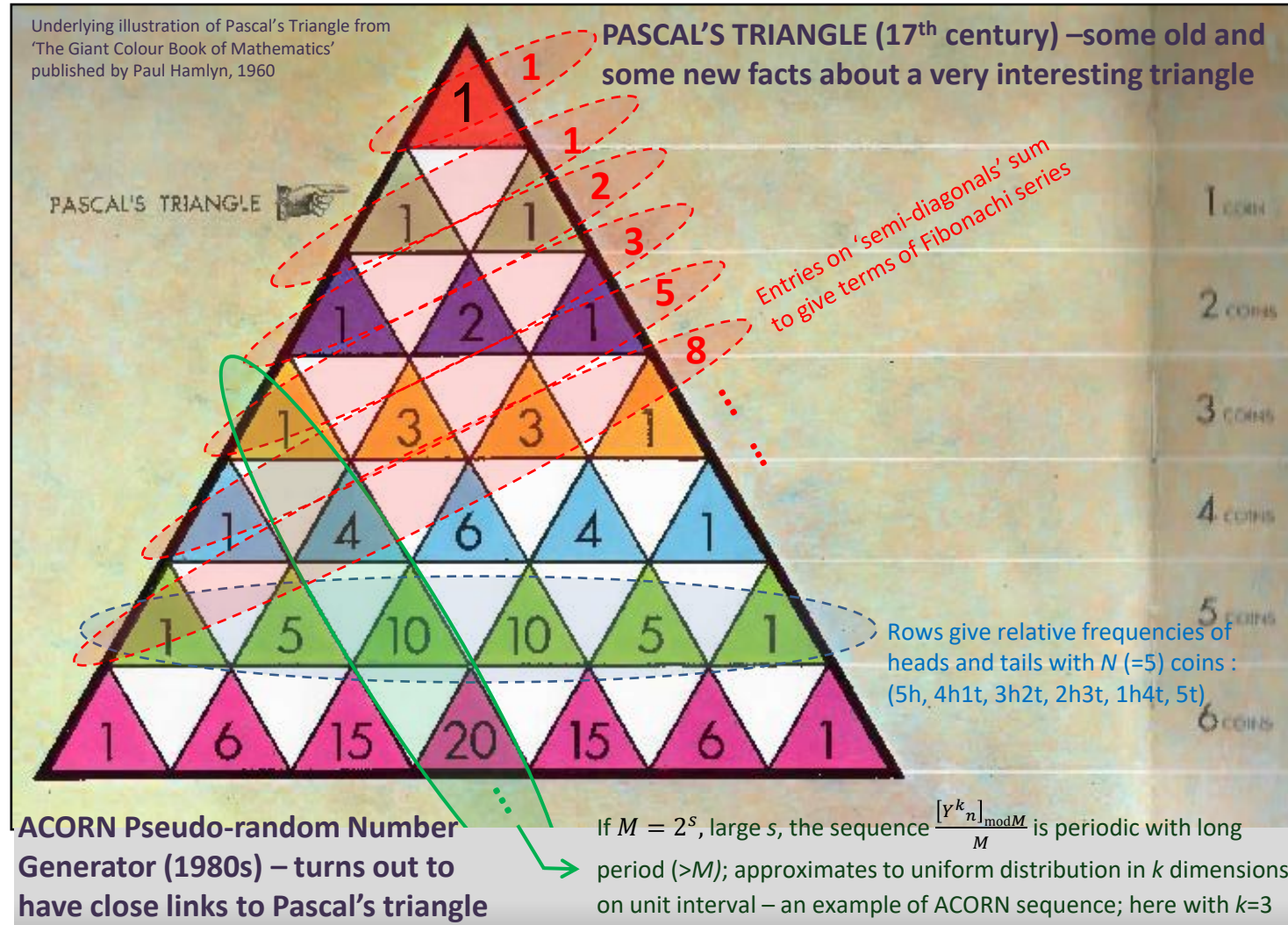
# References – [3], [4], [5], [6]

- [3] RSW, Theoretical Background for the ACORN Random Number Generator, Report AEA-APS-0244, AEA Technology, Winfrith, Dorset, UK, 1992.
- [4] RSW, Pseudo-random Number Generation for Parallel Processing – A Splitting Approach, *SIAM News*, **33** number 9, 2000.
- [5] RSW, 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].
- [6] RSW, 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].

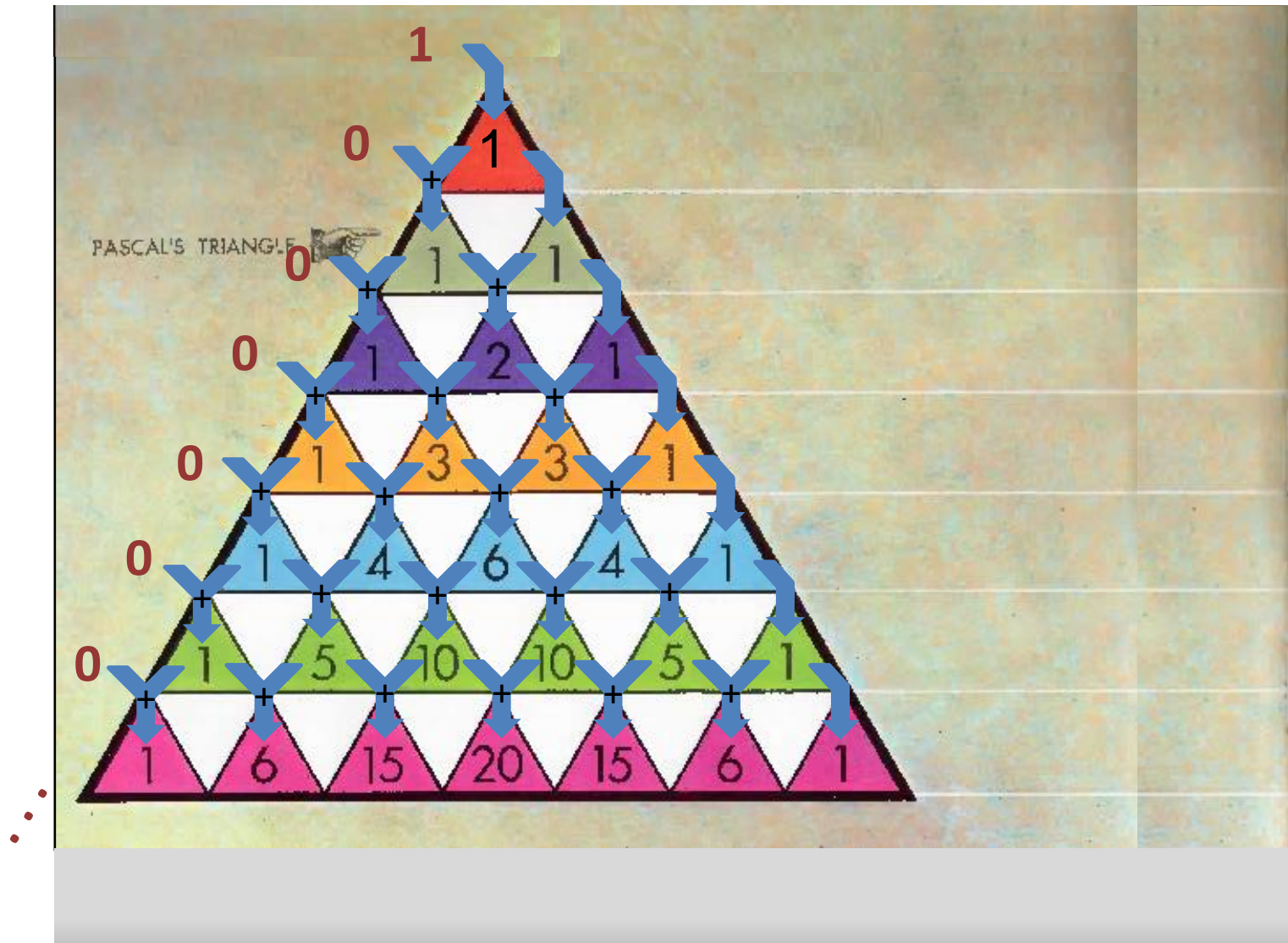
# ACORN Generators and Pascal's Triangle

- The ACORN generator turns out to have a close link with Pascal's triangle.
  - Numbering diagonals from 0 through  $k$ , the terms in  $k$ -th diagonal turn out to be a particular special case of a  $k$ -th order ACORN sequence.
- 
- Hence 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 with period a multiple of  $M$  and 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 in the references [1-6].

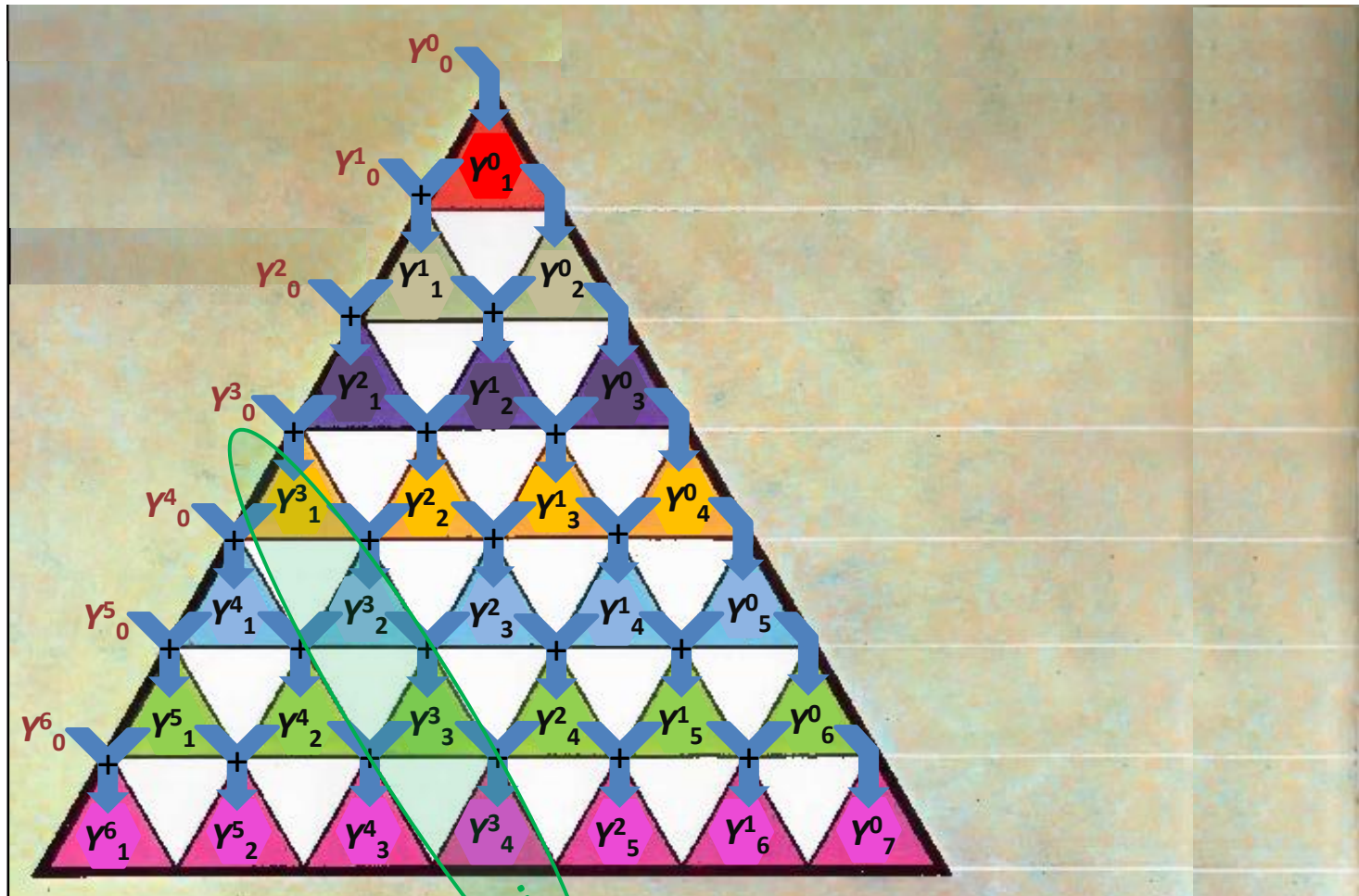
# Pascal's Triangle



## Augmented Pascal's triangle (modulo $M$ , a large power of 2)



ACORN triangle (all additions modulo  $M$ , a large power of 2).  
 If  $\gamma^0_0=1$  and  $\gamma^k_0=0$  (for  $k=1, 2, \dots$ ) this gives Pascal's triangle modulo  $M$ .



**ACORN Pseudo-random Number Generator (1980s) – turns out to have close links to Pascal's triangle**

If  $M = 2^s$ , large  $s$ , the sequence  $\frac{[\gamma^k_n]_{\text{mod } M}}{M}$  is periodic with long period ( $>M$ ); approximates to uniform distribution in  $k$  dimensions on unit interval – an example of ACORN sequence; here with  $k=3$

# Implementation

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

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 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.

```
C      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.

# Software Library Implementations

- ACORN generator used (alongside widely-used Mersenne Twister algorithm [7]) by Numerical Algorithms Group Ltd in their Fortran Numerical Software Libraries [8] and C Numerical Software Libraries [9] as one of their standard base methods for generating uniformly distributed pseudo-random numbers.
- A version of ACORN algorithm also included in the GSLIB geostatistical software library, Deutsch and Journel [10].

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

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

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

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

# TestU01 Test Suite

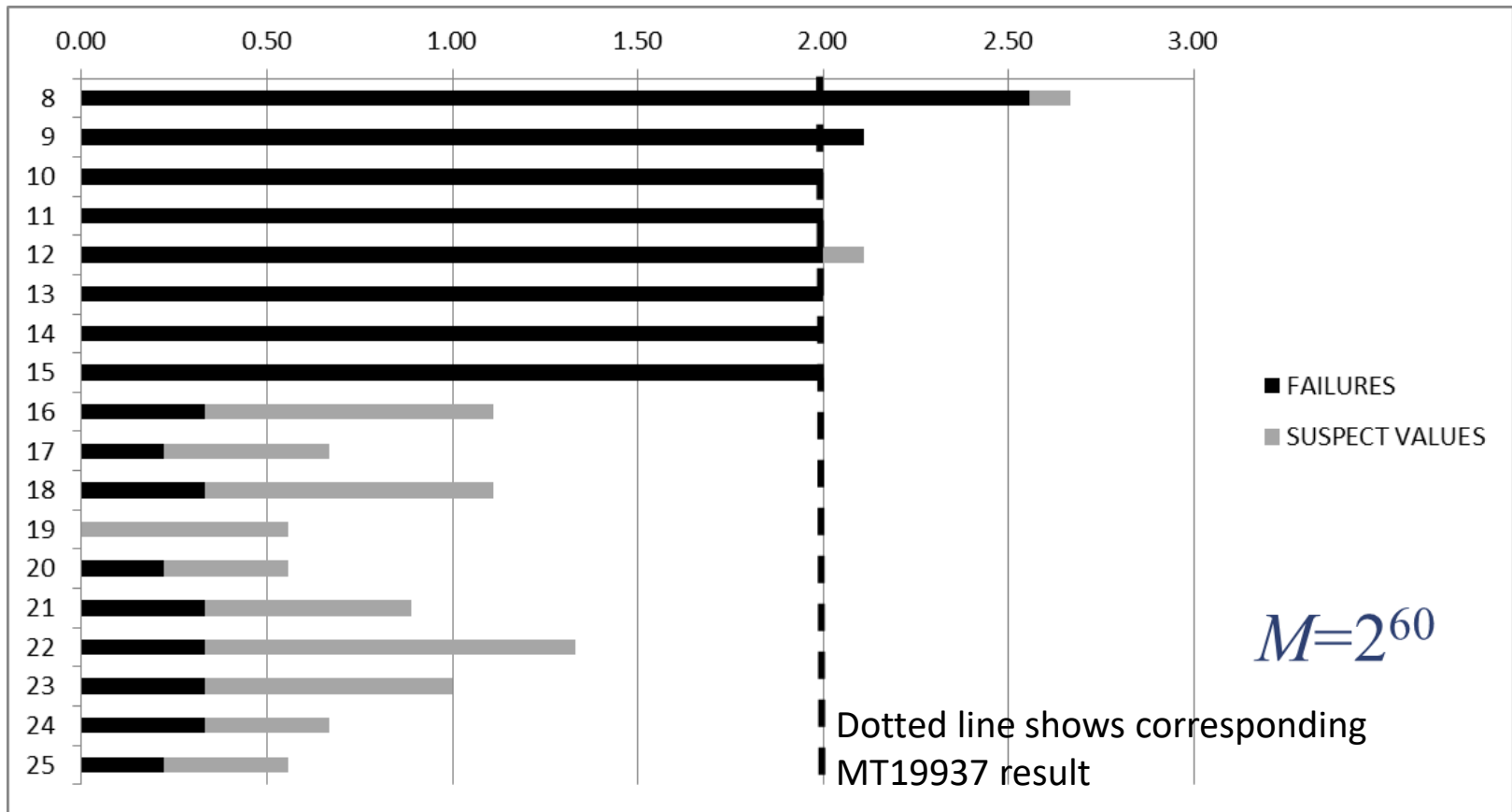
TestU01 package has been described by L'Ecuyer and Simard [10]. Considered application of empirical tests of uniformity and randomness to sequences generated by a wide range of algorithms - developed a comprehensive set of empirical tests designed to detect undesirable characteristics in such sequences. L'Ecuyer and Simard present results of applying the TestU01 tests to a large number of different sequences, identifying generators that pass all the tests (collectively called the BigCrush test battery), as well as identifying many generators (including some that are widely used) having serious deficiencies in respect of certain specific tests. Results presented are 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  $[10^{-10}, 1-10^{-10}]$  with a “suspect” value falling in one of the ranges  $[10^{-10}, 10^{-4}]$  or  $[1-10^{-4}, 1-10^{-10}]$ .

[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.

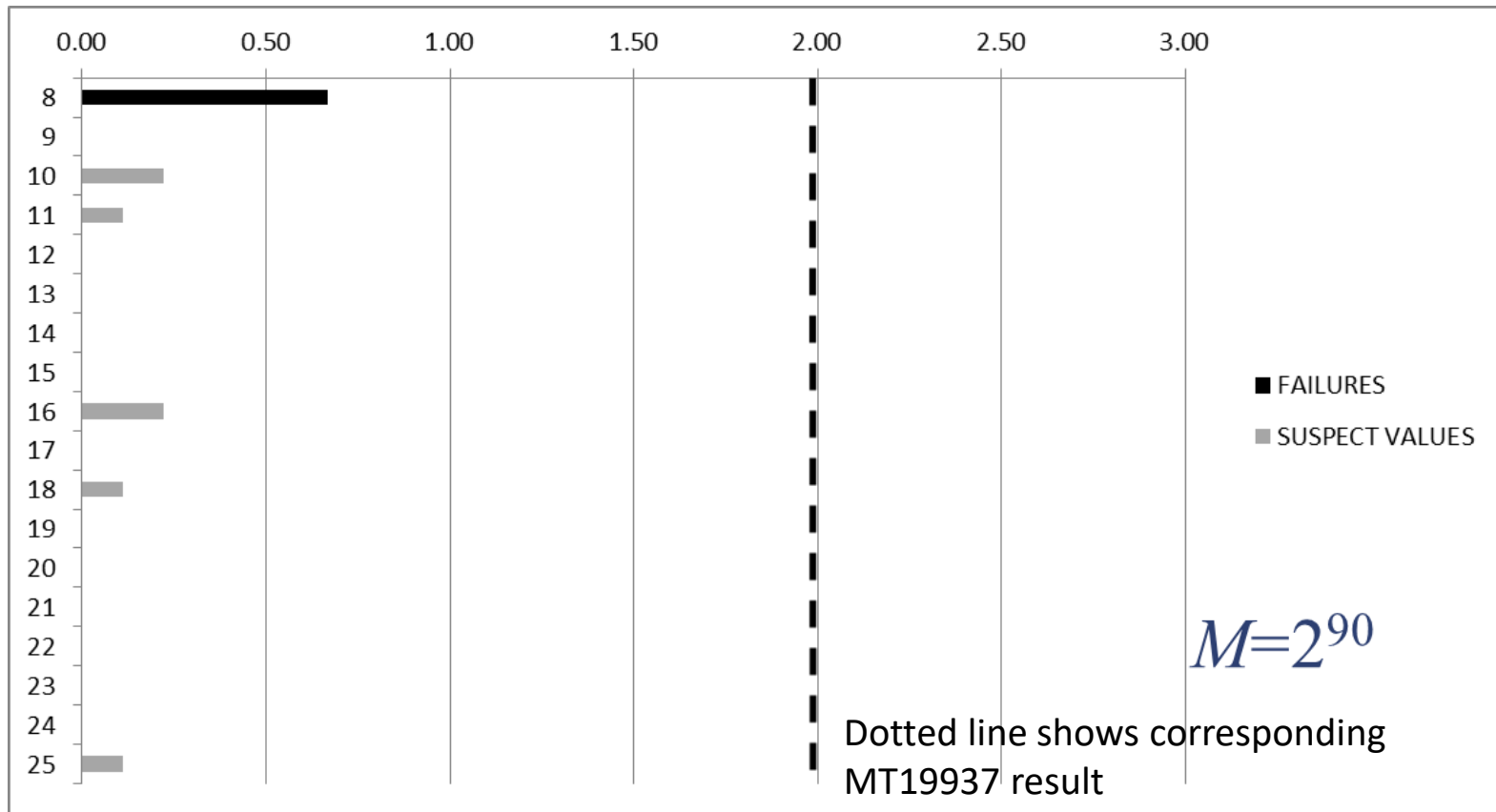
# Results with TestU01

- Results shown are the average number of ‘failures’ (black bars) and ‘suspect values’ (grey bars) obtained with nine different seeds and initialisations (plotted on the x-axis) for ACORN generators with order  $8 \leq k \leq 25$  (plotted on the y-axis) and three values of modulus  $M=2^{60}$ ,  $M=2^{90}$  and  $M=2^{120}$ .
- In effect, the **only constraint** on the initialisation for the nine cases selected for testing was that in each case the seed be a different odd integer less than the modulus; initial values were assigned arbitrarily, in general not all zeroes. Results of testing show improvement of the overall results with increasing modulus.
  - Some very straightforward and simple constraints on initialisation appear to “guarantee” passing all the tests – this is work in progress
- Results for the first seven cases were included in reference [2]; the final two cases represent new results added in the last 2 months.
- Each run of BigCrush requires some  $2^{38}$  pseudo-random variates and takes about 6 cpu hours to run; nine cases (initialisations) times 18 orders (8 to 25 inclusive) requires about 1000 cpu hours to generate each figure on the next three slides
  - Fortunately my laptop has four cores and is happy to work overtime while I am sleeping!
- Corresponding results obtained for the Mersenne Twister MT19937 generator, are shown by the dotted line on the figures.

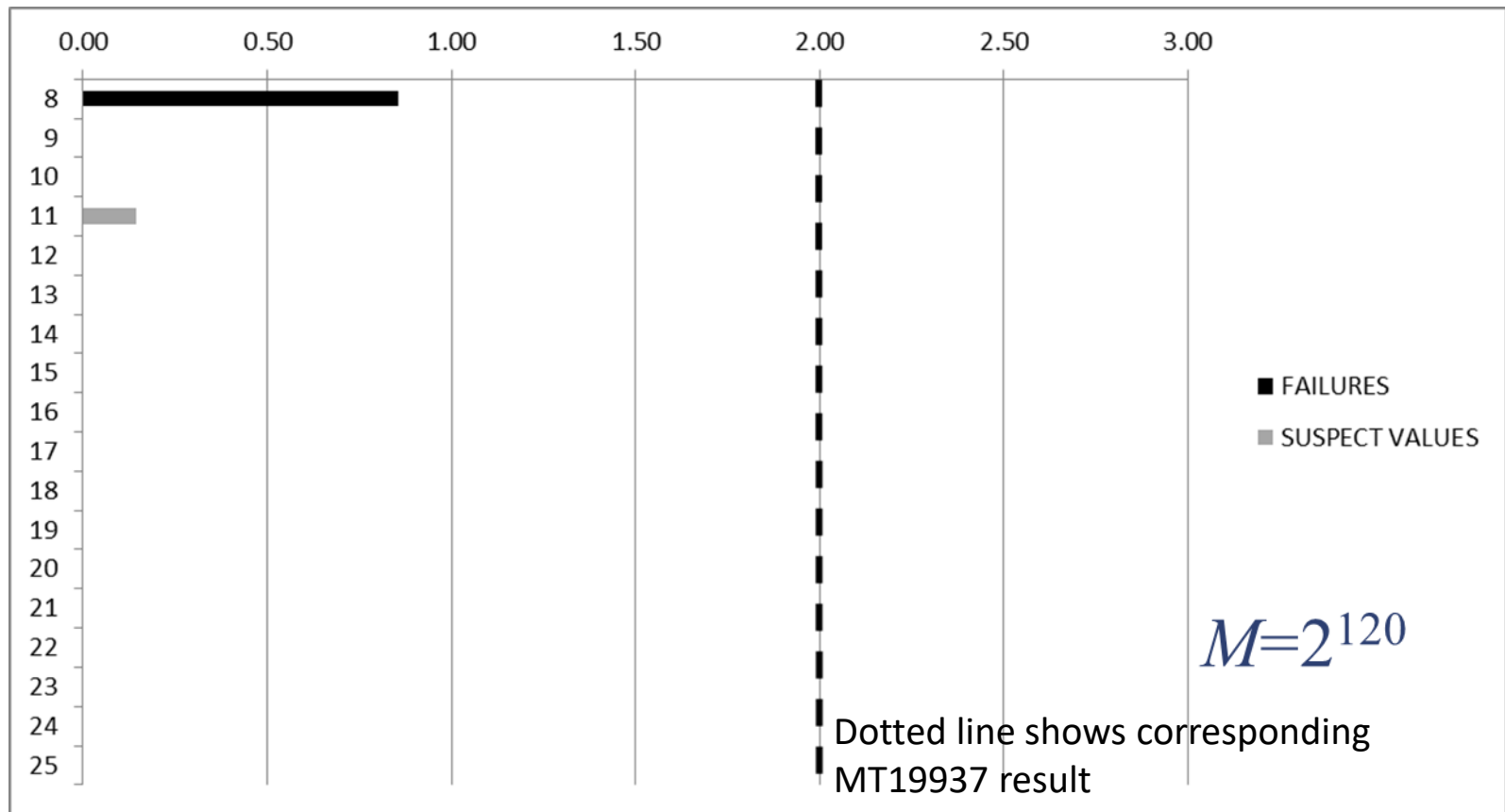
# TestU01 BigCrush Results – average for 9 different ACORN generators with modulus $2^{60}$



# TestU01 BigCrush Results – average for 9 different ACORN generators with modulus $2^{90}$



# TestU01 BigCrush Results – average for 9 different ACORN generators with modulus $2^{120}$



# Discussion

- With  $M=2^{120}$  and  $k \geq 9$ , ACORN generators passed all the BigCrush tests for each of the 9 different seeds and initialisations tested, with only very occasional ‘suspect values’
- With  $M=2^{90}$  and  $k \geq 9$ , ACORN generators passed all the BigCrush tests for each of the 9 different seeds and initialisations tested, only marginally more ‘suspect values’ than for  $M=2^{120}$
- With  $M=2^{60}$  and  $k \geq 9$ , ACORN generators failed on average no more than two of the BigCrush tests across the 9 different seeds and initialisations tested
- This contrasts with corresponding results obtained for the widely-used Mersenne Twister MT19937 generator, which consistently failed on (at least) two of the tests in both the Crush and BigCrush test suites, regardless of starting point.

# Comparison of ACORN and MT19937

## Mersenne Twister, MT19937

- Execution time  $\sim 10^{-8}$  seconds per call
- Period  $2^{19937-1}$
- Single sequence; consistently fails 2 of the TestU01 BigCrush tests, regardless of starting point chosen (some starting points are significantly worse)
- PROBLEM: What to do in the future if application requires “better quality” pseudo random numbers

## ACORN, Order 9 (or more), Modulus $2^{120}$

- Execution time  $\sim 10^{-8}$  seconds per call
- Period greater than  $2^{120}$  (a known multiple of modulus)
  - this is more than enough to outlast any conceivable processor
  - time to make  $2^{30}$  calls is  $\sim 8$  seconds; to make  $2^{60}$  calls would be  $\sim 300$  years
- Choice of more than  $2^{100}$  different sequences of each order which might “reasonably be expected” to pass all the TestU01 BigCrush tests –work in progress to demonstrate this
  - More than enough choices to use a different ACORN sequence that passes all TestU01 tests on every processor in existence now and on every conceivable future processor
- SOLUTION: If future applications require “better quality” pseudo random numbers, can increase order and increase modulus as require to give  $k$ -distributed prn, for any desired  $k$ 
  - Execution time scales as  $\sim k \log_2 M$

# Conclusions

- ACORN generators have been shown to reliably pass all the tests in the TestU01 BigCrush test suite for order between 9 and 25, modulus  $2^{120}$  and a very wide range of different initialisations; every odd value for the seed gives rise to a different sequence with the same statistical properties
- Statistical performance is better than some very widely used generators (including the Mersenne Twister MT19937) and comparable to the best currently available methods
- ACORN generators are amenable to theoretical analysis, and have been shown to give rise to some very interesting mathematics
  - A  $k$ -th order ACORN generator can be proved to approximate to being  $k$ -distributed; this approximation improves with increasing modulus (in a sense it “converges” to  $k$ -distributed)
  - Have recently demonstrated a surprising link between ACORN sequences and Pascal’s triangle
- ACORN sequences are worthy of further study!
- Happy to discuss further with anybody who is interested