# Normal Numbers and Pseudorandom Generators*

David H. Bailey† Jonathan M. Borwein‡

December 5, 2011

**Abstract**

For an integer $b \geq 2$ a real number $\alpha$ is *b-normal* if, for all $m > 0$, every $m$-long string of digits in the base-$b$ expansion of $\alpha$ appears, in the limit, with frequency $b^{-m}$. Although almost all reals in $[0, 1]$ are $b$-normal for every $b$, it has been rather difficult to exhibit explicit examples. No results whatsoever are known, one way or the other, for the class of "natural" mathematical constants, such as $\pi$, $e$, $\sqrt{2}$ and $\log 2$. In this paper, we summarize some previous normality results for a certain class of explicit reals, and then show that a specific member of this class, while provably 2-normal, is provably *not* 6-normal. We then show that a practical and reasonably effective pseudorandom number generator can be defined based on the binary digits of this constant, and conclude by sketching out some directions for further research.

*Submitted to Heinz Bauschke, ed., *Proceedings of the Workshop on Computational and Analytical Mathematics in Honour of Jonathan Borwein's 60th Birthday*, Springer, 2011.

†Lawrence Berkeley National Laboratory, Berkeley, CA 94720, DHBailey@lbl.gov. Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231.

‡Laureate Professor and Director Centre for Computer Assisted Research Mathematics and its Applications (CARMA), University of Newcastle, Callaghan, NSW 2308, Australia. Distinguished Professor, King Abdulaziz University, Jeddah 80200, Saudi Arabia. Email: jonathan.borwein@newcastle.edu.au.

# 1 Introduction

For an integer $b \geq 2$ we say that a real number $\alpha$ is *b-normal* (or *normal base b*) if, for all $m > 0$, every $m$-long string of digits in the base-$b$ expansion of $\alpha$ appears, in the limit, with frequency $b^{-m}$, or, in other words, with exactly the frequency one would expect if the digits appeared completely "at random." It follows from basic probability theory that, for any integer $b \geq 2$, almost all reals in the interval $(0, 1)$ are $b$-normal. What's more, almost all reals in the unit interval are simultaneously $b$-normal for all integers $b \geq 2$.

Yet identifying even a single explicitly given real number that is $b$-normal for some $b$ has proven frustratingly difficult. The first constant proven 10-normal was the Champernowne constant [7], namely $0.12345678910111213\ldots$, produced by concatenating the natural numbers in decimal format. This was extended to base-$b$ normality (for base-$b$ versions of the Champernowne constant). In 1946, Copeland and Erdös established that the corresponding concatenation of primes $0.23571113171923\ldots$ and also the concatenation of composites $0.46891012141516\ldots$, among others, are also 10-normal [8]. In general they proved:

**Theorem 1 ([8])** *If $a_1, a_2, \cdots$ is an increasing sequence of integers such that for every $\theta < 1$ the number of $a$'s up to $N$ exceeds $N^\theta$ provided $N$ is sufficiently large, then the infinite decimal*

$$0.a_1a_2a_3\cdots$$

*is normal with respect to the base $\beta$ in which these integers are expressed.*

This clearly applies to the primes of the form $ak + c$ with $a$ and $c$ relatively prime in any given base and to the integers which are the sum of two squares (since every prime of the form $4k + 1$ is included).

Some related results were established by Schmidt, including the following [15]. Write $p \sim q$ if there are positive integers $r$ and $s$ such that $p^r = q^s$. Then

**Theorem 2** *If $p \sim q$, then any real number that is $p$-normal is also $q$-normal. However, if $p \not\sim q$, then there are uncountably many $p$-normal reals that are not $q$-normal.*

In a recent survey, Queffelec [14] described the above result and also presented the following, which he ascribed to Korobov:

**Theorem 3** *Numbers of the form* $\sum_k p^{-2^k} q^{-p^{2^k}}$, *where* $p$ *and* $q$ *are relatively prime, are* $q$*-normal.*

Nonetheless, we are still completely in the dark as to the $b$-normality of "natural" constants of mathematics. Borel was the first to conjecture that *all* irrational algebraic numbers are $b$-normal for *every* integer $b \geq 2$. Yet not a single instance of this conjecture has ever been proven. We do not even know for certain whether or not the limiting frequency of zeroes in the binary expansion of $\sqrt{2}$ is one-half, although numerous large statistical analyses have failed to show any significant deviation from statistical normals. The same can be said for $\pi$ and other basic constants, such as $e, \log 2$ and $\zeta(3)$. Clearly any result (one way or the other) for one of these constants would be a mathematical development of the first magnitude.

In the case of an algebraic number of degree $d$, it is now known that the number of ones in the binary expansion through bit position $n$ must exceed $Cn^{1/d}$ for a positive number $C$ (depending on the constant) and all sufficiently large $n$ [1]. In particular, there must be at least $\sqrt{n}$ ones in the first $n$ bits of $\sqrt{2}$. But this is clearly a relatively weak result, because, barring an enormous mathematical surprise, the correct limiting frequency of ones in the binary expansion of $\sqrt{2}$ is, almost certainly, one-half.

In this paper, we briefly summarize some previously published normality results for a certain class of real constants, prove an interesting *non-normality* result, and then demonstrate how these normality results can be parlayed into producing a practical pseudorandom number generator. This generator can be implemented quite easily, is reasonably fast-running, and, in initial tests, seems to produce results of satisfactory "randomness." In addition, we show how all of this suggests a future direction to the long sought proof of normality for "natural" mathematical constants.

# 2 Normality of a class of generalized BBP-type constants

In [3], Richard Crandall and one of the present authors (Bailey) analyzed the class of constants

$$\alpha_{b,c}(r) \;=\; \sum_{k=1}^{\infty} \frac{1}{c^k b^{c^k + r_k}}, \tag{1}$$

where the integers $b > 1$ and $c > 1$ are co-prime, where $r$ is any real in $[0, 1]$, and where $r_k$ is the $k$th binary digit of $r$. These constants qualify as "generalized BBP-type constants," because the $n$-th base-$b$ digit can be calculated directly, without needing to compute any of the first $n-1$ digits, by a simple and efficient algorithm similar to that first applied to $\pi$ and $\log 2$ in the paper by Bailey, P. Borwein and Plouffe [2].

Bailey and Crandall were able to establish:

**Theorem 4** *Every real constant of the class (1) is b-normal.*

Subsequently, Bailey and Misieurwicz were able to establish this same result (at least in a simple demonstrative case) via a much simpler argument, utilizing a "hot spot" lemma proven by ergodic theory techniques [5] (see also [6, pg. 155]).

Fix integers $b$ and $c$ satisfying the above criteria, and let $r$ and $s$ be any reals in $[0, 1]$. If $r \neq s$, then $\alpha_{b,c}(r) \neq \alpha_{b,c}(s)$, so that the class $A_{b,c} = \{\alpha_{b,c}(r),\ 0 \leq r \leq 1\}$ has uncountably many distinct elements (this was shown by Bailey and Crandall). However, it is not known whether the class $A_{b,c}$ contains any constants of mathematical significance, such as $\pi$ or $e$.

In this paper we will focus on the constant $\alpha_{2,3}(0)$, which we will denote as $\alpha$ for short:

$$\begin{aligned} \alpha &= \alpha_{2,3}(0) = \sum_{k=1}^{\infty} \frac{1}{3^k 2^{3^k}} \qquad (2) \\ &= 0.041883680831502985071252898624571682426096758465485 7\ldots_{10} \\ &= \texttt{0.0AB8E38F684BDA12F684BF35BA781948B0FCD6E9E06522C3F35B}\ldots_{16}. \end{aligned}$$

Although its 2-normality follows from the results in either of the two papers mentioned above ([3] and [5]), this particular constant was first proved 2-normal by Stoneham back in 1973 [16].

# 3 A non-normality result

It should be emphasized that just because a real constant is $b$-normal for some integer $b > 1$, it does not follow that it is $c$-normal for any other integer $c$, except in the case where $b^r = c^s$ for positive integers $r$ and $s$ (see Theorem 2). In other words, if a constant is 8-normal, it is clearly 16-normal (since base-16 digits can be written as four binary digits and base-8 digits can be

written as three binary digits), but nothing can be said *a priori* about that constant's normality in any base that is not a power of two.

As mentioned above, there are very few normality results, and none is known for well-known constants of mathematics. But the same can be said about specific non-normality results, provided we exclude rationals (which repeat and thus are not normal) and examples, such as 1.0101000100000001 ... (i.e., ones appear in position $2^m$), that are constructed specifically not to be normal but otherwise have relatively little mathematical interest (although Liouville's class of transcendental numbers is an exception). In particular, none of the well-known "natural" constants of mathematics have ever been proven *not* to be $b$-normal for some $b$. Indeed, such a result, say for $\pi$, $\log 2$ or $\sqrt{2}$, would be even more interesting than a proof of normality for that constant.

In that vein, here is an intriguing result regarding the $\alpha$ constant mentioned above:

**Theorem 5** *$\alpha$ is* not *6-normal.*

**Discussion:** Let the notation $\{\cdot\}$ denote fractional part. Note that the base-6 digits immediately following position $n$ in the base-6 expansion of $\alpha$ can be obtained by computing $\{6^n\alpha\}$, which can be written as follows:

$$\{6^n\alpha\} \;=\; \left\{ \sum_{m=1}^{\lfloor \log_3 n \rfloor} 3^{n-m} 2^{n-3^m} \right\} + \left\{ \sum_{m=\lfloor \log_3 n \rfloor + 1}^{\infty} 3^{n-m} 2^{n-3^m} \right\}. \qquad (3)$$

Now note that the first portion of this expression is *zero*, since all terms of the summation are integers. That leaves the second expression.

Consider the case when $n = 3^M$, where $M \geq 1$ is an integer, and examine just the first term of the second summation. We see that this expression is

$$3^{3^M-(M+1)} 2^{3^M-3^{M+1}} \;=\; 3^{3^M-M-1} 2^{-2\cdot 3^M} \;=\; (3/4)^{3^M} / 3^{M+1}. \qquad (4)$$

We can generously bound the sum of all terms of the second summation by 1.00001 times this amount, for all $M \geq 1$, and by many times closer to unity for all $M \geq 2$, etc. Thus we have

$$\{6^{3^m}\alpha\} \;\approx\; \frac{\left(\frac{3}{4}\right)^{3^m}}{3^{m+1}}, \qquad (5)$$

0.
0130140430003334251130502130000001243555045432233011500243525320551352
3435410104300000000000000005141130054040555455303144250433435101241345
2351125142125134505503545015053522052044340452151505102411552500425130
0511244540010441311500324203032130000000000000000000000000000000000000
0000014212034311121452013525445342113412240220525301054204423552411055
4150155204350414555400310145303033532002534340401301240104453254343502
1420204324150255551010040433000455441145010313314511510144514123443342
3412400551313335045423530553151153501533452435450250055521453054234342
1530350125024205404135451231323245353031534552304115020154242121145201
5422225343403404505301233255344404431033324453321414150142334545424124
3203125340050134150245514404300000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000003133505424444311110555341410520145402134123130014243331331 15
...

Table 1: Base-6 expansion of $\alpha$.

and this approximation is as accurate as one wishes (in ratio) for all sufficiently large $m$.

Given the very small size of the expression $(3/4)^{3^m}/3^{m+1}$ for even moderate-sized $m$, it is clear the base-6 expansion will have very long stretches of zeroes beginning at positions $3^m + 1$. For example, by explicitly computing $\alpha$ to high precision, one can produce the counts of consecutive zeroes $Z_m$ that immediately follow position $3^m$ in the base-6 expansion of $\alpha$—see Tables 1 and 2.

In total, there 14256 zeroes in these ten segments, which, including the last segment, span the first $59049 + 9487 = 68536$ base-6 digits of $\alpha$. In this tabulation we have of course ignored the many zeroes in the large "random" segments of the expansion. Thus the fraction of the first 68536 digits that are zero is at least $14256/68536 = 0.20800747\ldots$, which is significantly more than the expected value $1/6 = 0.166666\ldots$.

A more careful analysis shows that this limiting ratio

$$\lim_{m\to\infty} \frac{\sum_{m\geq 1} Z_m}{3^m + Z_m} = \frac{3}{2} \cdot \frac{\log_6(4/3)}{1 + \log_6(4/3)} \tag{6}$$

$$= \frac{1}{2}\log_2(4/3) = 0.2075187496\ldots \tag{7}$$

| $m$ | $3^m$ | $Z_m$ |
|---|---|---|
| 1 | 3 | 1 |
| 2 | 9 | 3 |
| 3 | 27 | 6 |
| 4 | 81 | 16 |
| 5 | 243 | 42 |
| 6 | 729 | 121 |
| 7 | 2187 | 356 |
| 8 | 6561 | 1058 |
| 9 | 19683 | 3166 |
| 10 | 59049 | 9487 |

Table 2: Counts $Z_m$ of consecutive zeroes immediately following position $3^m$ in base-6 expansion of $\alpha$.

Complete details are given in the Appendix. Also included in the Appendix is a proof of this generalization of Theorem 5:

**Theorem 6** *Given coprime integers $b \geq 2$ and $c \geq 2$, the constant $\alpha_{b,c} = \sum_{k\geq 1} 1/(c^k b^{c^k})$ is* not *bc-normal.*

These results thus constitute simple and concrete counter-examples to the question of whether normality in one base $b$ implies normality in another base $c$ (except in simple cases covered by the first part of Theorem 2). In particular, these results are explicit examples of part two of Theorem 2.

It is worth pointing out that Cassels proved that for almost real $x$ in the unit interval, $x$ is 2-normal but not 3-normal, although he did not present any explicit example of such $x$ [4]. Above we have presented an explicit real that is 2-normal but not 6-normal, which is almost but not quite such an example. Some related discussion is given in [13, 15, 17].

# 4 Alpha as a pseudorandom generator

The normality result for $\alpha$ (Theorem 4) suggests that the binary digits of $\alpha$ (certainly not its base-6 digits) could be used to fashion a practical pseudorandom number generator. Indeed, this was suggested in [3] and [6, pg.

169–170]. We will show here how this can be done. The result is a generator that is both efficient on single-processor systems and also well-suited for parallel processing: each processor can quickly and independently calculate the starting seed for its section of the resulting global sequence, which global sequence is the same as the sequence produced on a single-processor system (subject to some reasonable conditions). However, it is acknowledged that before such a generator is used in a "practical" application, it must be subjected to significant checking and testing. It should also be noted that just because a number is normal does not guarantee its suitability for pseudo-random generation (e.g., the convergence of the limiting frequencies might be very slow), although this particular scheme does appear to be reasonably well-behaved.

## 4.1 Background

Define $x_n$ to be the binary expansion of $\alpha$ starting with position $n+1$. Note that $x_n = \{2^n\alpha\}$, where $\{\cdot\}$ means the fractional part of the argument. First consider the case $n = 3^m$ for some integer $m$. In this case one can write

$$x_{3^m} \;=\; \{2^{3^m}\alpha\} \;=\; \left\{\sum_{k=1}^{m} \frac{2^{3^m-3^k}}{3^k}\right\} + \sum_{k=m+1}^{\infty} \frac{2^{3^m-3^k}}{3^k}. \tag{8}$$

Observe that the "tail" term (i.e., the second term) in this expression is exceedingly small once $m$ is even moderately large—for example, when $m = 10$, this term is only about $10^{-35551}$. This term will hereafter be abbreviated as $\epsilon_m$. By expanding the first term, one obtains

$$\begin{aligned} x_{3^m} \;&=\; \frac{(3^{m-1}2^{3^m-3} + 3^{m-2}2^{3^m-3^2} + \cdots + 3\cdot 2^{3^m-3^{m-1}} + 1) \bmod 3^m}{3^m} \\ &\quad +\epsilon_m \end{aligned} \tag{9}$$

The numerator is taken modulo $3^m$, since only the remainder when divided by $3^m$ is of interest when finding the fractional part. By Euler's totient theorem, the next-to-last term in the numerator, when reduced modulo $3^m$, is three. Similarly, it can be seen that every other term in the numerator, when reduced modulo $3^m$, is equivalent to itself without the power-of-two

part. In other words, the expression above reduces to

$$x_{3^m} = \frac{(3^{m-1} + 3^{m-2} + \cdots + 3 + 1) \bmod 3^m}{3^m} + \epsilon_m \tag{10}$$

$$= \frac{3^m - 1}{2 \cdot 3^m} + \epsilon_m = \frac{\lfloor 3^m/2 \rfloor}{3^m} + \epsilon_m \tag{11}$$

(The authors are indebted to Helaman Ferguson for a key idea in this proof.) More generally, for $n$ that is not a power of three, one can write

$$x_n = \frac{(2^{n-3^m} \lfloor 3^m/2 \rfloor) \bmod 3^m}{3^m} + \epsilon, \tag{12}$$

where $m$ is chosen so that $3^m$ is the largest power of three less than or equal to $n$. In this case, one can be assured that $\epsilon < 10^{-30}$ provided $n$ is not within 100 of any power of three.

## 4.2 Algorithm

With this explicit expression in mind, an algorithm can be given for generating pseudorandom deviates, in the form of a sequence of IEEE 64-bit floating-point numbers in $(0, 1)$. These deviates contain, in their mantissas, successive 53-bit segments of the binary expansion of $\alpha$, beginning at some given starting position.

*Initialization.* First select a starting index $a$ in the range $3^{33} + 100 = 5559060566555623 \leq a \leq 2^{53} = 9007199254740992$. The value of $a$ can be thought of as the "seed" of the generator. Then calculate

$$z_0 = 2^{a-3^{33}} \cdot \lfloor 3^{33}/2 \rfloor \bmod 3^{33} \tag{13}$$

*Generate iterates.* Successive iterates of the generator can then be recursively computed by iterating

$$z_k = 2^{53} \cdot z_{k-1} \bmod 3^{33} \tag{14}$$

and then returning the values $z_k 3^{-33}$, which are 64-bit IEEE floating-point results in the unit interval.

*Arithmetic.* Several of the operations used in this scheme must be done with an accuracy of at least 106 mantissa bits. This can be done using

"double-double" arithmetic. A double-double datum is represented by a pair of IEEE double-precision floating-point numbers: the first word is the closest 64-bit IEEE value to the double-double value, and the second word is the difference. Algorithms for performing basic double-double arithmetic algorithms, using only rounded 64-bit IEEE floating-point operations, are given in [9] or [6, pg. 218-220]. These have been implemented in C++ and Fortran-90 double-double computation software packages, which include both basic-level arithmetic functions as well as common algebraic and transcendental functions, available from first author's website: `http://crd.lbl.gov/~dhbailey/mpdist`.

On other other hand, one could also use 128-bit integer or 128-bit IEEE floating-point arithmetic to do these operations, if these operations are available in hardware (software implementations tend to be relatively slow).

*Implementation details.* The operation $2^{53} \cdot z_{k-1} \bmod 3^{33}$ can be performed efficiently as follows: (1) multiply $2^{53}$ by $z_{k-1}$ (double times double yielding a double-double or 128-bit result); (2) multiply the result of step 1 (just the high-order portion will do) by $3^{-33}$ and take the greatest integer; (3) multiply the result of step 2 by $3^{33}$ (double times double yielding a double-double or 128-bit result); and (4) subtract the result of step 3 from the result of step 1 (using double-double or 128-bit arithmetic). It is possible that the result of step 2 might be one unit too high, or one too low, so that the result of step 4 may need to be adjusted accordingly: if it is negative, add $3^{33}$; if it exceeds $3^{33}$, subtract $3^{33}$.

*Exponentiation.* The exponentiation required in the initialization may be done efficiently using the binary algorithm for exponentiation. This is merely the formal name for the observation that exponentiation can be economically performed by means of a factorization based on the binary expansion of the exponent. For example, one can write $3^{17} = ((((3^2)^2)^2)^2) \cdot 3$, thus producing the result in only five multiplications, instead of the usual 16. According to Knuth, this technique dates back at least to 200 BCE [10, pg. 461]. In this application, the exponentiation result is required modulo a positive integer $k$. This can be done very efficiently by reducing modulo $k$ the intermediate multiplication result at each step of the exponentiation algorithm. A formal statement of this scheme is as follows:

To compute $r = b^n \bmod k$, where $r, b, n$ and $k$ are positive integers: First set

$t$ to be the largest power of two such that $t \le n$, and set $r = 1$. Then

A: if $n \ge t$ then $r \leftarrow br \bmod k$; $\quad n \leftarrow n - t$; $\quad$ endif
$t \leftarrow t/2$
if $t \ge 1$ then $r \leftarrow r^2 \bmod k$; $\quad$ go to A; $\quad$ endif

Note that the above algorithm is performed entirely with positive integers that do not exceed $k^2$ in size.

A full implementation of the entire pseudorandom scheme, which runs on any computer system with IEEE 64-bit arithmetic and a Fortran-90 compiler, can be obtained from the first author's website: `http://crd.lbl.gov/~dhbailey/mpdist`. The code is straightforward and can easily be converted to other languages, such as C or Java.

## 4.3 Analysis

It can be seen from the above that the recursive sequence generating iterates, which contain successive 53-long segments of binary digits from the expansion of $\alpha$, is nothing more than a special type of linear congruential pseudorandom number generator, a class that has been studied extensively by computer scientists and others [10, pg. 10–26]. In other words, the binary digits of $\alpha$ are "locally" (within a range of indices spanned by successive powers of three) given by a linear congruential generator, with a modulus that is a large power of three.

This observation makes it an easy matter to determine the period $P$ of the resulting generator [10, pg. 17]: as specified above, $P = 2 \cdot 3^{32} \approx 3.706 \cdot 10^{15}$. Note, however, that the binary digits of the resulting sequence will match that of $\alpha$ only if $[a, a+53n]$, where $a$ is the starting index and $n$ is the number of floating-point results generated, does not include a power of three or come within 100 of a power of three. If one can utilize 128-bit integer arithmetic, one could use a larger modulus, say $3^{40}$, which would yield a period that is 2187 times larger.

This scheme has one significant advantage over conventional linear congruential generators that use a power-of-two modulus: it cleanly avoids anomalies that sometimes arise in large scientific codes, when arrays with dimensions that are large powers of two are filled with pseudorandom data and then accessed both by row and by column (or plane), or which otherwise are accessed by large power-of-two data strides (as in a power-of-two FFT). This is because the pseudorandom data sequence accessed in this manner has

a reduced period, and thus may be not as "random" as desired. The usage of a modulus that is a large power of three is immune to these problems. The authors are not aware of any major scientific calculation that involves data access strides that are large powers of three.

## 4.4 Performance

As mentioned above, a Fortran-90 implementation of the scheme described above is available on the first author's website. For comparison purposes, the conventional linear congruential generator

$$z_n \;=\; 5^{21} \cdot z_{n-1} \bmod 2^{53}, \tag{15}$$

was implemented using the same software and programming style. These two codes were then tested on an 2.8 GHz Apple MacPro workstation, using the gfortran compiler (and running only on one of the eight cores). The program implementing the normal-number-based scheme required 3.553 seconds to generate an array of 100 million double-precision deviates. The conventional linear congruential system required essentially the same time.

By the way, the above program also is self-checking, in that it computes 100 million iterates using (14), then checks that the same value is produced by jumping ahead 100 million steps, by using formula (13). The present authors have used this program to check computational and data integrity on various computer systems. In at least one instance, the program disclosed intermittent memory errors.

## 4.5 Parallel Implementation

The scheme described above is very well suited for parallel processing, a trait not shared by a number of other commonly used pseudorandom schemes. Consider, for example, an implementation of the above pseudorandom scheme on a distributed memory system. Suppose that $k$ is the processor number and $p$ is the total number of processors used. Assume that a total of $n$ pseudorandom deviates are to be generated, and assume that $n$ is evenly divisible by $p$. Then each processor generates $n/p$ results, with processor $p$ using as a starting value $a + nk/p$. Note that each processor can quickly and independently generate its own value of $z_0$ by using formula (13).

In this way, the collective sequence generated by all processors coincides precisely with the sequence that is generated on a single processor system.

This feature is crucially important in parallel processing, permitting one can verify that a parallel program produces the same answers (to within reasonable numerical round-off error) as the single-processor version. It is also important, for the same reason, to permit one to compare results, say, between a run on 64 CPUs of a given system with one on 128 CPUs.

This scheme has been used to generate data for the fast Fourier transform (FFT) benchmark that is part of the benchmark suite for the High Productivity Computing Systems (HPCS) program, funded by the U.S. Defense Advanced Research Projects Agency (DARPA) and the U.S. Department of Energy.

## 4.6 Variations

Some initial tests, conducted by Nelson Beebe of the University of Utah, found that if by chance one iterate is rather small, it will include as its trailing bits a few of the leading bits of the next result (this is a natural consequence of the construction). While the authors are not aware of any application for which this feature would have significant impact, it can be virtually eliminated by advancing the sequence by more than 53 bits—say by 64 bits—from iterate to iterate.

This can be done by simply altering formula (14) above to read:

$$z_k \;=\; 2^{64} \cdot z_{k-1} \bmod 3^{33}. \tag{16}$$

This can be implemented as is, if one is using 128-bit integer or 128-bit IEEE floating-point arithmetic, but does not work correctly if one is using double-double arithmetic, because the product $2^{64} \cdot z_{k-1}$ could exceed $2^{106}$, which is the maximum size of an integer that can be represented exactly as a double-double operand. When using double-double arithmetic, one can compute each iterate using the following:

$$z_k \;=\; 2^{11} \cdot (2^{53} \cdot z_{k-1} \bmod 3^{33}) \bmod 3^{33}. \tag{17}$$

Tests by the present authors, advancing 64 bits per result, showed no significant correlation to the leading bits of the next iterate. And, of course, the additional "skip" here could be more than 11; it could be any value up to 53.

Finally, there is no reason that other constants from this class could not also be used in a similar way. For example, a very similar generator could be

constructed based on $\alpha_{2,5}$. One could also construct pseudorandom generators based on constants that are 3-normal or 5-normal, although one would lose the property that successive digits are precisely retained in consecutive computer words (which are based on binary arithmetic). The specific choice of multiplier and modulus can be made based on application requirements and the type of high-precision arithmetic that is available (e.g., double-double or 128-bit integer).

However, as we noted above, it is important to recognize that any proposed pseudorandom number generator, including this one, must be subjected to lengthy and rigorous testing [10, 11, 12]. Along this line, as noted above, generators of the general linear congruential family have problems, and it is not yet certain whether some variation or combination of generators in this class can be fashioned into a robust, reliable scheme that is both efficient and practical. But we do believe that these schemes are worthy of further study.

# 5 Conclusion and directions for further work

In this paper, we have shown how the constant $\alpha = \sum_{n\geq 1} 1/(3^n 2^{3^n})$, which is provably 2-normal, is *not* 6-normal, as well as some generalizations. These results thus constitute simple and concrete counter-examples to the question of whether normality in one base $b$ implies normality in another base $c$ (except in simple cases covered by the first part of Theorem 2). In particular, these results are explicit examples of the second part of Theorem 2. We have also shown how a practical pseudorandom number generator can be constructed based on the binary digits of $\alpha$, where each generated word consists of successive sections of its binary expansion.

Perhaps the most significant implication of the algorithm we have presented is not for its practical utility, but instead for the insight it provides to the fundamental question of normality. In particular, the pseudorandom number construction implies that the digit expansions of one particular class of provably normal numbers consist of successive segments of exponentially growing length, and within each segment the digits are given by a specific type of linear congruential generator, with a period that also grows exponentially. From this perspective, the 2-normality of $\alpha$ is entirely plausible.

Now consider what this implies, say, for the normality of a constant such

as $\log 2$. First recall the classical formula

$$\log 2 \;=\; \sum_{n=1}^{\infty} \frac{1}{n2^n}. \tag{18}$$

Thus, following the well-known BBP approach (see [2] or [6, Chap. 4]), we can write

$$\{2^d \log 2\} \;=\; \left\{ \sum_{n=1}^{d} \frac{2^{d-n} \bmod n}{n} \right\} + \left\{ \sum_{n=d+1}^{\infty} \frac{2^{d-n}}{n} \right\}. \tag{19}$$

This leads immediately to the BBP algorithm for computing the binary digits of $\log 2$ beginning after position $d$, since each term of the first summation can be computed very rapidly by means of the binary algorithm for exponentiation, and the second summation quickly converges.

But we can also view (19) for its insight on normality. Note that the binary expansion of $\log 2$ following position $d$ can be seen as a sum of normalized linear congruential pseudorandom number generators, with periods (at least in some terms) that grow steadily with $n$ (since the period of a linear congruential generator depends on the factorization of the modulus). But with increasing $n$, at least some terms will have prime moduli, resulting in relatively long periods. In fact, some will be primitive primes modulo two, which give the maximal period $(n-1)/2$. Note that the sum of normalized linear congruential generators can be rewritten as a single linear congruential generator. Thus it is plausible that the period of the sum of generators in the first portion of (19) increases without bound, resulting in a highly "random" expansion (although all of this needs to be worked out in detail).

We have attempted to develop these notions further, but so far we have not made a great deal of progress. But, at the least, this approach may be effective for constants such as

$$\beta \;=\; \sum_{n \in W} \frac{1}{n2^n}, \tag{20}$$

where $W$ is the set of primitive primes modulo two, which as mentioned above give rise to a maximal periods when used as a linear congruential modulus. Only time will tell.

# 6 Appendix

**Proof of Theorem 5**: $\alpha_{2,3}$ is not 6-normal.

Let $Q_m$ be the base-6 expansion of $\alpha_{2,3}$ immediately following position $3^m$ (i.e., after the "decimal" point has been shifted to the right $3^m$ digits). We can write

$$\begin{aligned} Q_m &= 6^{3^m}\alpha_{2,3} \bmod 1 \\ &= \left(\sum_{k=1}^{m} 3^{3^m-k}2^{3^m-3^k}\right) \bmod 1 + \sum_{k=m+1}^{\infty} 3^{3^m-k}2^{3^m-3^k}. \end{aligned} \tag{21}$$

The first portion of this expression is zero, since all terms in the summation are integers. The small second portion is very accurately approximated by the first term of the series, namely $(3/4)^{3^m}/3^{m+1}$. In fact, for all $m \geq 1$,

$$\frac{(3/4)^{3^m}}{3^{m+1}} < Q_m < \frac{(3/4)^{3^m}}{3^{m+1}}(1 + 2 \cdot 10^{-6}). \tag{22}$$

Let $Z_m = \lfloor \log_6 1/Q_m \rfloor$ be the number of zeroes in the base-6 expansion of $\alpha$ that immediately follow position $3^m$. Then for all $m \geq 1$, (22) can be rewritten

$$\begin{aligned} & 3^m \log_6\left(\frac{4}{3}\right) + (m+1)\log_6 3 - 2 \\ < \; Z_m \; < \; & 3^m \log_6\left(\frac{4}{3}\right) + (m+1)\log_6 3. \end{aligned} \tag{23}$$

Now let $F_m$ be the fraction of zeroes in the base-6 expansion of $\alpha$ up to position $3^m + Z_m$ (i.e., up to the end of the block of zeroes that immediately follows position $3^m$). Clearly

$$F_m > \frac{\sum_{k=1}^{m} Z_k}{3^m + Z_m}, \tag{24}$$

since the numerator only counts zeroes in the long stretches. The summation in the numerator satisfies, for all sufficiently large $m$,

$$\begin{aligned} \sum_{k=1}^{m} Z_k &> \frac{3}{2}\left(3^m - \frac{1}{3}\right)\log_6\left(\frac{4}{3}\right) + \frac{m(m+3)}{2}\log_6 3 - 2m \\ &> \frac{3}{2}\cdot 3^m \log_6\left(\frac{4}{3}\right) - \frac{1}{2}\log_6\left(\frac{4}{3}\right) - 2m. \end{aligned} \tag{25}$$

Now given any $\epsilon > 0$, we can write, for all sufficiently large $m$,

$$\begin{aligned} F_m &> \frac{\frac{3}{2} \cdot 3^m \log_6\left(\frac{4}{3}\right) - \frac{1}{2}\log_6\left(\frac{4}{3}\right) - 2m}{3^m + 3^m \log_6\left(\frac{4}{3}\right) + (m+1)\log_6 3} \\ &= \frac{\frac{3}{2}\log_6\left(\frac{4}{3}\right) - \frac{1}{3^m}\left(\frac{1}{2}\log_6\left(\frac{4}{3}\right) + 2m\right)}{1 + \log_6\left(\frac{4}{3}\right) + \frac{(m+1)\log_6 3}{3^m}} \\ &\geq \frac{\frac{3}{2}\log_6\left(\frac{4}{3}\right) - \epsilon}{1 + \log_6\left(\frac{4}{3}\right) + \epsilon} \;\geq\; \frac{1}{2}\log_2\left(\frac{4}{3}\right) - 2\epsilon. \end{aligned} \tag{26}$$

But $\beta = \frac{1}{2}\log_2(4/3)$ (which has numerical value $0.2075187496\ldots$) is clearly greater than $1/6$, since $(4/3)^3 = 64/27 > 2$. This means that infinitely often (namely, whenever $n = 3^m + Z_m$) the fraction of zeroes in the base-6 expansion of $\alpha$ up to position $n$ exceeds $\frac{1}{2}(1/6 + \beta) > 1/6$. Thus $\alpha$ is not 6-normal. **QED**.

**Proof of Theorem 6**: Given co-prime integers $b \geq 2$ and $c \geq 2$, the constant $\alpha_{b,c} = \sum_{k\geq 1} 1/(c^k b^{c^k})$ is not $bc$-normal.

Let $Q_m(b,c)$ be the base-$bc$ expansion of $\alpha_{b,c}$ immediately following position $c^m$. Then

$$\begin{aligned} Q_m(b,c) &= (bc)^{c^m}\alpha_{b,c} \bmod 1 \\ &= \left(\sum_{k=1}^{m} c^{c^m - k} b^{c^m - c^k}\right) \bmod 1 + \sum_{k=m+1}^{\infty} c^{c^m - k} b^{c^m - c^k}. \end{aligned} \tag{27}$$

As above, the first portion of this expression is zero, since all terms in the summation are integers, and the second portion is very accurately approximated by the first term of the series, namely $[\frac{c}{b(c-1)}]^{c^m}/c^{m+1}$. In fact, for any choice of $b$ and $c$ as above, and for all $m \geq 1$,

$$\frac{1}{c^{m+1}}\left[\frac{c}{b(c-1)}\right]^{c^m} < Q_m(b,c) < \frac{1}{c^{m+1}}\left[\frac{c}{b(c-1)}\right]^{c^m} \cdot (1 + 1/10). \tag{28}$$

Let $Z_m(b,c) = \lfloor \log_{bc} 1/Q_m(b,c) \rfloor$ be the number of zeroes that immediately follow position $c^m$. Then for all $m \geq 1$, (28) can be rewritten as

$$\begin{aligned} & c^m \log_{bc}\left[\frac{b(c-1)}{c}\right] + (m+1)\log_{bc} c - 2 \\ < \; & Z_m(b,c) \; < \; c^m \log_{bc}\left[\frac{b(c-1)}{c}\right] + (m+1)\log_{bc} c. \end{aligned} \tag{29}$$

Now let $F_m(b,c)$ be the fraction of zeroes up to position $c^m + Z_m(b,c)$. Clearly

$$F_m(b,c) \;>\; \frac{\sum_{k=1}^{m} Z_k(b,c)}{c^m + Z_m(b,c)}, \tag{30}$$

since the numerator only counts zeroes in the long stretches. The summation in the numerator of $F_m(b,c)$ satisfies

$$\begin{aligned}\sum_{k=1}^{m} Z_k(b,c) &> \frac{c}{c-1}\left(c^m - \frac{1}{c}\right)\log_{bc}\left[\frac{b(c-1)}{c}\right] + \frac{m(m+3)}{2}\log_{bc} c - 2m \\ &> \frac{c^{m+1}}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] - \frac{1}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] - 2m. \end{aligned} \tag{31}$$

Thus given any $\epsilon > 0$, we can write, for all sufficiently large $m$,

$$\begin{aligned} F_m(b,c) &> \frac{\frac{c^{m+1}}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] - \frac{1}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] - 2m}{c^m + c^m\log_{bc}\left(\frac{b(c-1)}{c}\right) + (m+1)\log_{bc} c} \qquad (32) \\ &= \frac{\frac{c}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] - \frac{1}{c^m}\left(\frac{1}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] + 2m\right)}{1 + \log_{bc}\left[\frac{b(c-1)}{c}\right] + \frac{(m+1)\log_{bc} c}{c^m}} \\ &\geq \frac{\frac{c}{c-1}\log_{bc}\left[\frac{b(c-1)}{c}\right] - \epsilon}{1 + \log_{bc}\left[\frac{b(c-1)}{c}\right] + \epsilon} \\ &\geq \frac{c}{c-1}\cdot\frac{\log_{bc}\left[\frac{b(c-1)}{c}\right]}{1 + \log_{bc}\left[\frac{b(c-1)}{c}\right]} - 2\epsilon. \\ &= T(b,c) - 2\epsilon, \qquad (33) \end{aligned}$$

where

$$T(b,c) \;=\; \frac{c}{c-1}\cdot\frac{\log_{bc}\left[\frac{b(c-1)}{c}\right]}{1 + \log_{bc}\left[\frac{b(c-1)}{c}\right]}. \tag{34}$$

To establish the desired result that $T(b,c) > 1/(bc)$, first note that

$$T(b,c) \;>\; \frac{1}{2}\log_{bc}\left[\frac{b(c-1)}{c}\right] \;\geq\; \frac{1}{2}\log_{bc}\left(\frac{b}{2}\right). \tag{35}$$

Raise $bc$ to the power of the right-hand side, and also to the power $1/(bc)$. Then it suffices to demonstrate that

$$\frac{b}{2} \;>\; \left[(bc)^{1/(bc)}\right]^2. \tag{36}$$

The right-hand side is bounded above by $(e^{1/e})^2 = 2.0870652286\ldots$. Thus this inequality is clearly satisfied whenever $b \geq 5$.

If we also presume that $c \geq 5$, then by examining the middle of (35) it suffices to demonstrate that

$$\frac{1}{2}\log_{bc}\frac{4b}{5} \;>\; \frac{1}{bc} \tag{37}$$

or

$$\frac{4b}{5} \;>\; \left(e^{1/e}\right)^2. \tag{38}$$

But this is clearly satisfied whenever $b \geq 3$. For the case $b = 2$ and $c \geq 5$, we can write

$$T(b,c) \;=\; \frac{c}{c-1}\cdot\frac{\log_{2c}\left[\frac{2(c-1)}{c}\right]}{1+\log_{2c}\left[\frac{2(c-1)}{c}\right]} \;\geq\; \frac{\log_{2c}\left[\frac{2(c-1)}{c}\right]}{1+\log_{10}2}, \tag{39}$$

so by similar reasoning it suffices to demonstrate that

$$\frac{2(c-1)}{c} \;>\; \left(e^{1/e}\right)^{1+\log_{10}2} \;=\; 1.61384928833\ldots. \tag{40}$$

But this is clearly satisfied whenever $c \geq 6$.

The five remaining cases, namely $(2,3), (2,5), (3,2), (3,4), (4,3)$, are easily verified by explicitly computing numerical values of $T(b,c)$ using (34). As it turns out, the simple case that we worked out in detail above, namely $b = 2$ and $c = 3$, is the worst case, in the sense that for all other $(b,c)$, the fraction $T(b,c)$ exceeds the natural frequency $1/(bc)$ by greater margins. **QED**.

# References


[1] David H. Bailey, Jonathan M. Borwein, Richard E. Crandall and Carl Pomerance, "On the binary expansions of algebraic numbers," *Journal of Number Theory Bordeaux*, vol. 16 (2004), pg. 487–518.

[2] David H. Bailey, Peter B. Borwein and Simon Plouffe, "On the rapid computation of various polylogarithmic constants," *Mathematics of Computation*, vol. 66, no. 218 (Apr 1997), pg. 903–913.

[3] David H. Bailey and Richard E. Crandall, "Random generators and normal numbers," *Experimental Mathematics*, vol. 11 (2002), no. 4, pg. 527–546.

[4] J. W. S. Cassels, "On a problem of Steinhaus about normal numbers," *Colloquium Mathematicum*, voo. 7 (1959), pg. 95–101.

[5] David H. Bailey and Michal Misiurewicz, "A strong hot spot theorem," *Proceedings of the American Mathematical Society*, vol. 134 (2006), no. 9, pg. 2495–2501.

[6] Jonathan M. Borwein and David H. Bailey, *Mathematics by Experiment: Plausible Reasoning in the 21st Century*, AK Peters, Natick, MA, and ed., 2008.

[7] D. G. Champernowne, "The construction of decimals normal in the scale of ten," *Journal of the London Mathematical Society*, vol. 8 (1933), pg. 254–260.

[8] A. H. Copeland and P. Erdös, "Note on normal numbers," *Bulletin of the American Mathematical Society*, vol. 52 (1946), pg. 857–860.

[9] Yozo Hida, Xiaoye S. Li and David H. Bailey, "Algorithms for quad-double precision floating point arithmetic," *15th IEEE Symposium on Computer Arithmetic*, IEEE Computer Society, 2001, pg. 155–162.

[10] Donald E. Knuth, *The Art of Computer Programming*, vol. 2, Addison-Wesley, Boston, 1998.

[11] P. L'Ecuyer, "Random number generation," in *Handbook of Computational Statistics*, J. E. Gentle, W. Haerdle and Y. Mori, ed., Springer-Verlag, Berlin, 2004, Chap. II.2.

[12] P. L'Ecuyer and R. Simard, "TestU01: A C Library for empirical testing of random number generators," *ACM Transactions on Mathematical Software*, vol. 33 (Aug. 2007).

[13] A. D. Pollington, "The Hausdorff dimension of a set of normal numbers," *Pacific Journal of Mathematics*, vol. 95 (1981), pg. 193–204.

[14] Martine Queffelec, "Old and new results on normality," *Lecture Notes – Monograph Series*, vol. 48, *Dynamics and Stochastics*, 2006, Institute of Mathematical Statistics, pg. 225–236.

[15] W. Schmidt, "On normal numbers," *Pacific Journal of Mathematics*, vol. 10 (1960), pg. 661–672.

[16] R. Stoneham, "On absolute $(j, \epsilon)$-normality in the rational fractions with applications to normal numbers," *Acta Arithmetica*, vol. 22 (1973), 277–286.

[17] H. Weyl, "Uber die Gleichverteilung von Zahlen mod. Eins," *Mathematische Annalen*, vol. 77 (1916), pg. 313–352.