```
      else:
         break
   return(count)

def phi(p, q, nu):
   while p * 2**nu > q:
       p = p // 2
   return(p)

#--- 1. Main

for k in range(N):

   p = (p+q)**2
   q = 2**mu *q**2

   str_p = gmpy2.digits(p, 2)
   str_p = str_p[0:ndigits]
   p = gmpy2.mpz(str_p, 2)

   str_q = gmpy2.digits(q, 2)
   str_q = str_q[0:ndigits]
   q = gmpy2.mpz(str_q, 2)

   old_p = p
   p = phi(p, q, nu)

   old_nmatch = nmatch
   nmatch = count_matching_prefix_chars(str_p, str_lim)
   old_newdigits = newdigits
   newdigits = str_p[old_nmatch:nmatch]
   pair = (old_newdigits, newdigits)
   update_hash(hash_pairs, pair, 1)
   x = p/q
   update_hash(hash_newdigits, newdigits, 1)
   if k % 1000 == 0:
      print("New digits:", k, nmatch, newdigits)

#--- 2. Summary results

print("\n\n")
hash_newdigits = dict(sorted(hash_newdigits.items(), key=lambda item: item[1], reverse=True))
c1 = 0 # counts digits equal to 1
c0 = 0 # counts digits equal to 0
for key in hash_newdigits:
   noccur = hash_newdigits[key]
   c1 += noccur * key.count('1')
   c0 += noccur * key.count('0')
   print("New digits hash summary:",c0, c1, noccur, key)

print("\n\n")
hash_pairs = dict(sorted(hash_pairs.items(), key=lambda item: item[1], reverse=True))
for pair in hash_pairs:
   cnt = hash_pairs[pair]
   if cnt > 5:
      print("Pair:", cnt, pair)
```

5.5 Correlated bit strings: seminal result and applications

So far, I looked at individual digit sequences separately. Now I focus on comparing sequences, and more specifically, identifying cross-correlations (or their absense) to build other tests of randomness, and with cryptographic applications in mind. Let $x \in [0, 1[$ be a real number. Its digits d_0, d_1 and so on in integer base $b > 1$ are obtained using the following recursion:

$$x_n = \{bx_{n-1}\} = \{b^n x_0\}, \quad d_n = \lfloor bx_n \rfloor \tag{5.28}$$

where $x_0 = x$. The brackets $\{\cdot\}$ denote the fractional part while $\lfloor \cdot \rfloor$ denotes the integer part function. If x is a normal number, the lag-k autocorrelation in the sequence (x_n), that is, the correlation between the sequences (x_n) and (x_{n+k}), is equal to b^{-k}. However, the autocorrelations of any lag in the digit sequence (d_n) are all zero. See section 3.2 entitled "Probabilistic properties of numeration systems" in [14]. Correlation should be interpreted as the limit of the empirical correlation based on the first n terms in the sequence, as $n \to \infty$. The limit exists if x is a normal number. For the exact formulation and computation, see the code section 5.5.2.

A less well-known result is the following.

Theorem 5.5.1 *If $x > 0$ is a normal in base 2 and p, q are odd coprime positive integers, then px, qx are also normal in base 2. The correlation between the binary digits of px and qx is equal to*

$$\rho\Big(px, qx\Big) = \frac{1}{pq} = \rho\Big(x, \frac{qx}{p}\Big) = \rho\Big(x, \frac{px}{q}\Big) \tag{5.29}$$

Proof

As a starting point, it is easy to show that for two normal numbers x, y, the correlation between their binary digit sequences $(d_k(x))$ and $(d_k(y))$ satisfies

$$\rho_n(x, y) := -1 + \frac{4}{n}\sum_{k=0}^{n-1} d_k(x)d_k(y) \;\to\; \rho(x, y), \quad \text{as } n \to \infty. \tag{5.30}$$

Also, a normal number multiplied by a non-zero rational is normal in the same base (Wall's theorem, 1949; see also [9]). Thus, the binary digit distributions of x, px and qx have the same mean $\frac{1}{2}$ and same variance $\frac{1}{4}$. Not all the equalities in (5.29) need to be proved separately, as we have trivial equivalences, using a change of variables preserving normality, and the fact that $\rho(\cdot\,,\cdot)$ is symmetric. For instance, thanks to the substitution $x \mapsto x/p$, we have

$$\rho\Big(px, qx\Big) = \frac{1}{pq} \implies \rho\Big(x, \frac{qx}{p}\Big) = \frac{1}{pq}.$$

Also, by a symmetry argument, swapping p and q, we have:

$$\rho\Big(x, \frac{px}{q}\Big) = \frac{1}{pq} \iff \rho\Big(x, \frac{qx}{p}\Big) = \frac{1}{pq}.$$

A proof that $\rho(x, px/q) = (pq)^{-1}$ was first published by William Huber on CrossValidated.com in 2019, see here and here. The proof assumes that the binary digits of x are randomly distributed as an infinite Bernoulli trial with 50% of 0 and 1, a stronger assumption than normality in base 2 ∎

Now, if x is a normal number, $\alpha \neq \beta$ are positive integers and p, q are odd coprime positive integers, then we have the following (the proof is left as an exercise):

$$\rho\Big(2^\alpha px, 2^\beta qx\Big) = 0. \tag{5.31}$$

I now can state and prove the following deep result with important implications, for instance the fact that the binary digit sequences of $\sqrt{2}$ and $\sqrt{3}$ are uncorrelated, that is $\rho(\sqrt{2}, \sqrt{3}) = 0$, if both are normal numbers.

Theorem 5.5.2 *Let x, y be normal numbers in base 2, linearly independent over $\mathbb{Q}$. Then $\rho(x, y) = 0$.*

Proof

Linear independence over $\mathbb{Q}$ means that if $\alpha x = \beta y$ for some integers α, β, we must have $\alpha = \beta = 0$. There are infinitely many pairs of sequences $(\alpha_t), (\beta_t)$ such that $y_t = \alpha_t x/\beta_t \to y$ as $t \to \infty$, with α_t, β_t being positive odd coprimes for all t. By virtue of theorem 5.5.1, we have

$$\rho(x, y_t) = \frac{1}{\alpha_t \beta_t}.$$

As t increases, the number of identical digits on the left in x and y_t increases, and thus $\rho(x, y_t) \to \rho(x, y)$. At the same time, $\alpha_t, \beta_t \to \infty$ because there is no rational number r such that $x = ry$ due to x, y being linearly independent over $\mathbb{Q}$. Thus, $\rho(x, y) = 0$. ∎

Formulas (5.29) and (5.31) lead to a new test of randomness. For instance, to check if the binary digits of a number x are random enough, you first compute $\lambda_n(p, q) = \rho_n(px, qx)$, the empirical correlation on the first n digits, for various values of n, p, q. Then run N simulations, replacing the digits of x by N sequences of random bits. Now let $L_n(p, q)$ and $U_n(p, q)$ be the lower and upper correlations computed over the N samples with fixed p, q. If $\lambda_n(p, q) \notin [L_n(p, q), U_n(p, q]$ far more often than expected by chance, then the digits of x are presumed non-random. Also, if for some p, q, the value $(pq)^{-1}$ is not within these two bounds, it means that your random number generator is defective.

Theorem 5.5.2 is particularly useful in the context of strong PRNGs (pseudo-random number generators), with applications to cryptography where replicability is mandatory, or to test the strength of other PRNGs. In particular, the PRNG discussed in section 4.4 in [14] relies on a large number of quadratic irrationals: the square roots of square-free integers. Random bits are generated by

- choosing (say) 10^6 such numbers,
- for each of them extracting 10^4 binary digits starting at a random location in the digit expansion,
- then concatenate all the collected digits to produce 10^{10} random bits.

This PRNG offers up to 10^6 distinct seed pairs. Each pair consists of (1) a square-free integer and (2) the location or index where the binary digit sequence must start in the associated quadratic irrational. Then theorem 5.5.2 guarantees that the 10^6 digit sequences are uncorrelated, if the underlying square root numbers are normal.

5.5.1 Autocorrelations in related sequences

The sequence $x_n = \{\beta + x_{n-1}\} = \{\beta n + x_0\}$ where $\beta > 0$ is an irrational number, behaves very differently from that defined by (5.28). Again, the n-th "digit" in base b is defined as $d_n = \lfloor bx_n \rfloor$. Now the invariant measure is unique and applies to all x_0. Again, it is the uniform distribution on $[0, 1]$, but the k-lag autocorrelations are much stronger and do not decay as k increases. Likewise, the digits have strong long-range autocorrelations, a big contrast with (5.28) where they are uncorrelated. This generalizes to the sequence $x_n = \{\beta n^\theta\}$. For details, see [27].

5.5.2 Python code

Now I share my Python code to compute the correlation between the binary digits of px and qx, where x is a real number in $[0, 1]$ and p, q are positive integers, coprime or not, odd or even. The function `vg_correl` computes the digits backward with carry-over and returns the correlation, while `gmpy2_correl` uses the gmpy2 library to compute the correlation between the digts of x and those of px/q. The code is also on GitHub, here.

The current version of `gmpy2_correl` has a bug caused by `w_offset` not correct when $p > q$. For instance, `gmpy2_correl(z,p,1)` and `vg_correl(z,p,1)` should obviously return the same correlation equal to $1/p$, but only the latter does, due to digits misalignment in the former when $p > q$ (in this example, $q = 1$).

```
# Compute binary digits of X, p*X, q*X backwards (assuming X is random)
# Only digits after the decimal point (on the right) are computed
# Compute correlations between digits of p*X and q*X
# Include carry-over when performing grammar school multiplication

import numpy as np
import gmpy2

kmax = 10000000
ctx = gmpy2.get_context()
ctx.precision = kmax
ndigits = ctx.precision
z = gmpy2.sqrt(2) # in the article, z is denoted as x

# main parameters
seed = 195
np.random.seed(seed)
# p, q odd integers, coprime
p = 3
q = 5

def gmpy2_correl(z, p, q):

    # correl b/w binary digits of z and pz/q (needs p < q)
    zstri = gmpy2.digits(z, 2)[0] # get binary digits of z as a string
    zoff = gmpy2.digits(z, 2)[1]

    w = gmpy2.mpfr(z*p)/gmpy2.mpz(q)
    woff = gmpy2.digits(w, 2)[1]
    w_offset = '0' * (zoff - woff) # works only if p < q
    wstri = w_offset + gmpy2.digits(w, 2)[0]

    prod = 0
    for k in range(kmax):
        d1 = int(zstri[k])
        d2 = int(wstri[k])
        prod += d1*d2
        correl = 4*prod/(k+1) - 1
        if k % 100000 == 0 and k > 100:
            checksum = correl * p * q # should be close to 1
            print("gmpy2> k: %7d correl: %9.7f check: %9.7f" %(k, correl, checksum))
    return(correl)
```

```
43
44
45  def vg_correl(z, p, q):
46
47      # correl b/w binary digits of pz and qz
48      mode = 'constant' # options: 'random', 'constant'
49
50      # local variables
51      zstri = gmpy2.digits(z, 2)[0] # get binary digits of z as a string
52      X, pX, qX = 0, 0, 0
53      d1, d2, e1, e2 = 0, 0, 0, 0
54      prod, count = 0, 0
55      sum1 = 0
56      sum2 = 0
57
58      # loop over digits in reverse order
59      for k in range(kmax):
60
61          # b is a digit of X
62          if mode == 'random':
63              b = np.random.randint(0, 2)
64          else:
65              b = int(zstri[kmax-k-1])
66          X = b + X/2
67
68          c1 = p*b
69          old_d1 = d1
70          old_e1 = e1
71          d1 = (c1 + old_e1//2) %2 # digit of pX
72          e1 = (old_e1//2) + c1 - d1
73          pX = d1 + pX/2
74
75          c2 = q*b
76          old_d2 = d2
77          old_e2 = e2
78          d2 = (c2 + old_e2//2) %2 #digit of qX
79          e2 = (old_e2//2) + c2 - d2
80          qX = d2 + qX/2
81
82          prod += d1*d2
83          count += 1
84          sum1 += d1
85          sum2 += d2
86          mean1 = sum1/count
87          mean2 = sum2/count
88          std1 = (mean1 * (1 - mean1))**0.5
89          std2 = (mean2 * (1 - mean2))**0.5
90          covar = prod/count - mean1*mean2
91          if count > 100:
92              correl = covar/(std1*std2)
93          else:
94              correl = 0
95          #correl = 4*prod/count - 1
96
97          if k% 100000 == 0:
98              checksum = p*q*correl # should be close to 1
99              print("vg>k = %7d, correl = %9.6f checksum = %9.6f" % (k, correl, checksum))
100
101     print("\np = %3d, q = %3d" %(p, q))
102     print("X = %12.9f, pX = %12.9f, qX = %12.9f" % (X, pX, qX))
103     print("X = %12.9f, p*X = %12.9f, q*X = %12.9f" % (X, p*X, q*X))
104     print("Correl = %7.4f, 1/(p*q) = %7.4f" % (correl, 1/(p*q)))
105     return(correl)
106
107 #--- Main
108
109 correl1 = gmpy2_correl(z, p, q)
110 correl2 = vg_correl(z, p, q)
```

Bibliography

[1] Franklin T. Adams-Watters and Frank Ruskey. Generating functions for the digital sum and other digit counting sequences. *Journal of Integer Sequences*, 12:1–9, 2009. [Link]. 12, 14, 23, 32, 45

[2] Christoph Aistleitner et al. Normal numbers: Arithmetic, computational and probabilistic aspects. 2016. Workshop [Link]. 12, 23, 32, 45

[3] Adel Alamadhi, Michel Planat, and Patrick Solé. Chebyshev's bias and generalized Riemann hypothesis. *Preprint*, pages 1–9, 2011. arXiv:1112.2398 [Link]. 84

[4] Gökalp Alpan and Maxim Zinchenko. Lower bounds for weighted Chebyshev and orthogonal polynomials. *Preprint*, 2024. arXiv:2408.11496v [Link]. 56

[5] David Bailey, Jonathan Borwein, and Neil Calkin. *Experimental Mathematics in Action*. A K Peters, 2007. 11, 23, 32, 45, 61

[6] Verónica Becher, A. Marchionna, and G. Tenenbaum. Simply normal numbers with digit dependencies. *Mathematika*, 69:988–991, 2023. arXiv:2304.06850 [Link]. 12, 23, 32, 45

[7] Frederik Broucke. On zero-density estimates for Beurling zeta functions. *Preprint*, pages 1–24, 2024. arXiv:2409:1051v1 [Link]. 77

[8] James Dolan. Carrying is a 2-cocycle. *Preprint*, pages 1–9, 2023. [Link]. 12, 23, 32, 45

[9] David Doty, Jack H. Lutz, and Satyadev Nandakumar. Finite-state dimension and real arithmetic. *Information and Computation*, 205:1640–1651, 2007. arXiv:cs/0602032 [Link]. 70

[10] Faiza Firdousi, Syeda Iram Batool, and Muhammad Amin. A novel construction scheme for nonlinear component based on quantum map. *International Journal of Theoretical Physics*, 58:3871–3898, 2019. [Link]. 12, 23, 32, 45

[11] P. M. Gauthier. Approximating the Riemann zeta-function by polynomials with restricted zeros. *Canadian Mathematical Bulletin*, 62(3):475–478, 2018. [Link]. 95

[12] Vincent Granville. *Stochastic Processes and Simulations: A Machine Learning Perspective*. MLT, 2022. [Link]. 77, 94

[13] Vincent Granville. *Synthetic Data and Generative AI*. MLT, 2022. [Link]. 77

[14] Vincent Granville. *Gentle Introduction To Chaotic Dynamical Systems*. MLT, 2023. [Link]. 7, 10, 11, 12, 14, 15, 18, 23, 32, 44, 45, 57, 61, 69, 70, 77, 78, 83, 84

[15] Vincent Granville. *Building Disruptive AI & LLM Technology from Scratch*. MLT, 2024. [Link]. 15, 23, 32, 45, 100

[16] Vincent Granville. *State of the Art GenAI & LLMs, Creative Projects & Solutions*. MLT, 2024. [Link]. 20, 84

[17] Vincent Granville. *Statistical Optimization for AI and Machine Learning*. MLT, 2024. [Link]. 78

[18] Vincent Granville. *Synthetic Data and Generative AI*. Elsevier, 2024. [Link]. 82

[19] Vincent Granville. *Blueprint: Next-Gen Enterprise RAG & LLM 2.0 – Nvidia PDFs Use Case*. 2025. MLT [Link]. 100

[20] Vincent Granville. Simple, efficient, secure, accurate enterprise AI xLLM 2.0 architecture & operating system. 2025. BondingAI internal report `bdai-scores.pdf`, July 2025. 100

[21] Vincent Granville. *No-Blackbox, Secure, Efficient AI and xLLM Solutions*. MLT, 2026. [Link]. 95, 100

[22] Vincent Granville and Richard L Smith. Disaggregation of rainfall time series via Gibbs sampling. *NISS Technical Report*, pages 1–21, 1996. [Link]. 94

[23] Emil Grosswald. Oscillation theorems of arithmetical functions. *Transactions of the American Mathematical Society*, 126:1–28, 1967. [Link]. 84

[24] Adam J. Harper. Moments of random multiplicative functions, II: High moments. *Algebra and Number Theory*, 13(10):2277–2321, 2019. [Link]. 85

[25] Adam J. Harper. Moments of random multiplicative functions, I: Low moments, better than squareroot cancellation, and critical multiplicative chaos. *Forum of Mathematics, Pi*, 8:1–95, 2020. [Link]. 85

[26] Adam J. Harper. Almost sure large fluctuations of random multiplicative functions. *Preprint*, pages 1–38, 2021. arXiv [Link]. 85

[27] Christopher Lutsko, Athanasios Sourmelidis, and Niclas Technau. Pair correlation of the fractional parts of αn^{θ}. *Journal of the European Mathematical Society*, 27:4069–4082, 2025. arXiv:2106.09800 [Link]. 71

[28] M. Madritsch and J. Thuswaldner. The level of distribution of the sum-of-digits function of linear recurrence number systems. *Journal de Théorie des Nombres de Bordeaux*, 34:449–482, 2022. MLT [Link]. 32, 45

[29] Vaibhav Mohanty et al. Maximum mutational robustness in genotype–phenotype maps follows a self-similar blancmange-like curve. *The Royal Society Publishing*, pages 1–16, 2023. [Link]. 12, 23, 32, 45

[30] Mohammadamin Moradi et al. Data-driven model discovery with Kolmogorov-Arnold networks. *Preprint*, pages 1–6, 2024. arXiv:2409.15167 [Link]. 24, 32, 45

[31] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1:4–27, 1990. [Link]. 23, 32, 45

[32] Alan Oppenheim and Ronald W. Schafer. *Discrete–Time Signal Processing*. Prentice Hall, 1999. 97

[33] Michel Planat and Patrick Solé. Efficient prime counting and the Chebyshev primes. *Preprint*, pages 1–15, 2011. arXiv:1109.6489 [Link]. 84

[34] Klaus Schiefermayr and Maxim Zinchenko. Norm estimates for Chebyshev polynomials, i. *Journal of Approximation Theory*, 265, 2021. [Link]. 56

[35] Jan-Christoph Schlage-Putcha and Jasson Vindas. The prime number theorem for Beurlings generalized numbers – new cases. pages 1–26, 2011. [Link]. 77

[36] Terence Tao. Biases between consecutive primes. *Tao's blog*, 2016. [Link]. 84

[37] Yury V. Tiumentsev and Mikhail V. Egorchev. *Neural Network Modeling and Identification of Dynamical Systems*. Elsevier, 2019. 23, 32, 45

[38] Chukwudubem Umeano and Oleksandr Kyriienko. Ground state-based quantum feature maps. *Preprint*, pages 1–8, 2024. arXiv:2024.07174 [Link]. 12, 23, 32, 45

[39] Joseph Vandehey. On the binary digits of $\sqrt{2}$. *Preprint*, pages 1–6, 2017. arXiv:1711.01722 [Link]. 11, 23, 32, 45, 61

[40] Troy Vasiga and Jeffrey Shallit. On the iteration of certain quadratic maps over GF(p). *Discrete Mathematics*, 277:219–240, 2004. [Link]. 23, 32, 44

[41] Henry S. Warren. *Hacker's Delight*. Addison-Wesley Professional, second edition, 2012. 12, 23, 32, 45

[42] Rose Yu and Rui Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, 121, 2024. [Link]. 23, 32, 45