Contents

1 Link Between the Digit Sum Function and Auto-Convolutions **6**
1.1 A new type of string operators . . . 6
1.1.1 String class . . . 6
1.1.2 Truncation, number representation, and convergence . . . 6
1.1.3 String convolution and square root . . . 7
1.1.4 Well-balanced strings . . . 7
1.2 Infinite sequences of iterated auto-convoluted strings . . . 7
1.2.1 Visualizing iterated auto-convoluted string sequences . . . 8
1.2.2 Fundamental result about the number of zeros and ones . . . 9
1.3 Solving one of the greatest mathematical mysteries . . . 12
1.3.1 Testing different seeds . . . 13
1.3.2 Another interesting sequence . . . 15
1.3.3 Surprising, unpublished results about the digit distribution . . . 15
1.3.4 Application to cryptography . . . 15
1.3.5 Python code . . . 16

2 Petabyte Challenge to Help Crack the Famous Math Conjecture **19**
2.1 Introduction . . . 19
2.2 Spectacular behavior of the digit sum function . . . 20
2.2.1 Potential scenarios . . . 20
2.2.2 Dynamics of the unbroken bifurcation process . . . 21
2.3 Case studies . . . 22
2.4 An extreme case . . . 23
2.5 Applications and AI Challenge with petabytes dataset . . . 24
2.5.1 AI challenge . . . 24
2.5.2 The dataset . . . 25
2.6 Python code . . . 26

3 From Digit Sum to Universal Dataset and Benchmarking AI Algorithms **29**
3.1 Introduction . . . 29
3.2 Deep dive into the digit sum function . . . 30
3.2.1 Digit sum function: examples . . . 30
3.2.2 Spectacular behavior of digit sum with primorials . . . 31
3.2.3 Future research . . . 32
3.2.4 References . . . 33
3.2.5 Comparison with standard methodology . . . 33
3.3 Infinite dataset and applications . . . 34
3.4 Python code . . . 34
3.4.1 Forward iterations . . . 35
3.4.2 Backward iterations . . . 38

4 Quantum Dynamics, Logistic Map, and Digit Distribution of Special Constants **41**
4.1 Introduction . . . 41
4.2 Logistic map and the digit sum function . . . 42
4.2.1 Model comparison, with illustrations . . . 42
4.2.2 Normality of special math constants . . . 45
4.2.3 Applications and references . . . 45
4.3 Re-balancing an uneven digit distribution . . . 46
4.3.1 Digit-balancing transforms . . . 46

4.3.2 Digit block balancing . . . 48
4.4 Conclusion . . . 52
4.5 Main Python code . . . 53

5 Test of Normality and Digit Distribution of Algebraic Numbers 56
5.1 Simple normality test with application to PRNGs . . . 56
5.1.1 High-performance computing with Chebyshev polynomials . . . 57
5.1.2 Application with test of randomness and Python code . . . 58
5.1.3 Problem and solution . . . 60
5.2 Another interesting discrete quadratic dynamical system . . . 60
5.2.1 Case with multiple limits . . . 61
5.2.2 Case with single limit . . . 61
5.3 Surprising results about the digit distribution . . . 62
5.3.1 Python code for the computer-assisted proof of the main theorem . . . 64
5.3.2 Python code for the deeper theorem . . . 66
5.4 Strong patterns found in the digits of algebraic numbers . . . 67
5.4.1 Python code to compute the digits . . . 69
5.5 Correlated bit strings: seminal result and applications . . . 70
5.5.1 Autocorrelations in related sequences . . . 72
5.5.2 Python code . . . 72

6 Quantum States and the Riemann Zeta Function 74
6.1 Synthetic primes, quantum states, and the Riemann Hypothesis . . . 74
6.1.1 Definitions . . . 74
6.1.2 Building a Beurling eta function by deletion . . . 75
6.1.3 Building a Beurling eta function by swapping . . . 75
6.1.4 Applications and Python code . . . 76
6.2 Quantum derivatives, GenAI, and the Riemann Hypothesis . . . 82
6.2.1 Cornerstone result to bypass the roadblocks . . . 83
6.2.2 Quantum derivative of functions nowhere differentiable . . . 84
6.2.3 Project and solution . . . 85
6.2.4 Python code . . . 89

7 Quantum, chaotic and fractal types of algorithmic convergence 94
7.1 Digit count generating function and fractal convergence . . . 94
7.2 Other examples of chaotic convergence . . . 96
7.2.1 Smooth convergence but with multiple branches . . . 96
7.2.2 Chaotic convergence with multiple branches . . . 96
7.2.3 Deep dive into the chaotic case . . . 100
7.2.4 Interesting connection between 3^n and the digits of $\sqrt{2}$ . . . 102
7.3 Random polynomials and spectral analysis of digit distributions . . . 105
7.3.1 Orbits of the digit generating function . . . 105
7.3.2 Connection to Littlewood, Shapiro, and Newman Polynomials . . . 107
7.3.3 Spectral signature of even versus uneven bit strings . . . 108
7.3.4 Mesmerizing video featuring 1000 cases, with Python code . . . 110

8 Towards Deterministic AI: The NPG Random Generator 115
8.1 NPG: fast non-periodic pseudo-random number generator . . . 115
8.1.1 NPG non-standard fundamental examples . . . 115
8.1.2 Deterministic vs cryptographically secure NPG vs other PRNGs . . . 116
8.1.3 NPG success vs PCG64 and PCG64DXSM statistical failures . . . 117
8.2 Compute time and Python code . . . 119
8.2.1 NPG source code . . . 120
8.2.2 PRNG tests of randomness . . . 123

Appendix A The Pi Day xLLM agent 127

Appendix B Convolution, Approximations, and Signal Processing 130
B.1 Approximations to mathematical function . . . 130
B.1.1 Finding the roots of ζ with fast-converging series . . . 130
B.1.2 Approximation based on quantization . . . 131
B.2 Non-causal discrete convolution with Gaussian kernel . . . 132

Let ν be the length of the bit string x_{N-1} concatenated with x_N. It depends on a, b and N. I generated $M = 8$ bit strings $B_0, \ldots, B_7$ of length ν, for K different combinations of a and b. For a specific a, b, the first one B_0 comes from my system, the other ones are generated sequentially by PCG64 with the Numpy seed set by `rng_seed`. For now, assume that $K = 1$ and $N = 1000$. In the code, K is implicitly detected as the number of elements (parameter sets a, b) in the `params` array. Also, the first and last few bits of each bit string are discarded, by setting `skip_flag=True`. Of course, a, b have no direct impact on the success or not of PCG64, but ν (a function of a, b) does. The failures are listed in Table 8.1, with ν denoted as "Bits" and `rng_seed` denoted as "Seed". The latter is used for PCG64, not in my system. The "seed" in my system (in the classical sense) is $x_0 = 1$, the lowest possible value, and not listed in the table. Instead, a, b play that role.

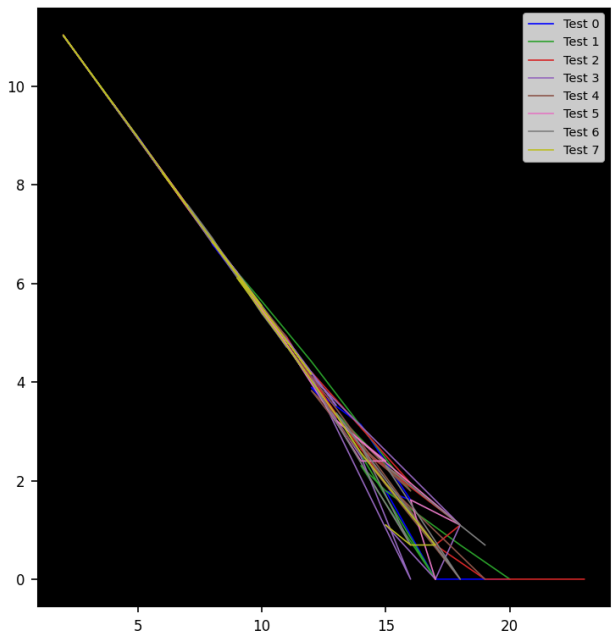

Figure 8.1: Runs of length L; L on X-axis, count (log scale) on Y-axis. Blue for NPG, non-blue for PCG64.

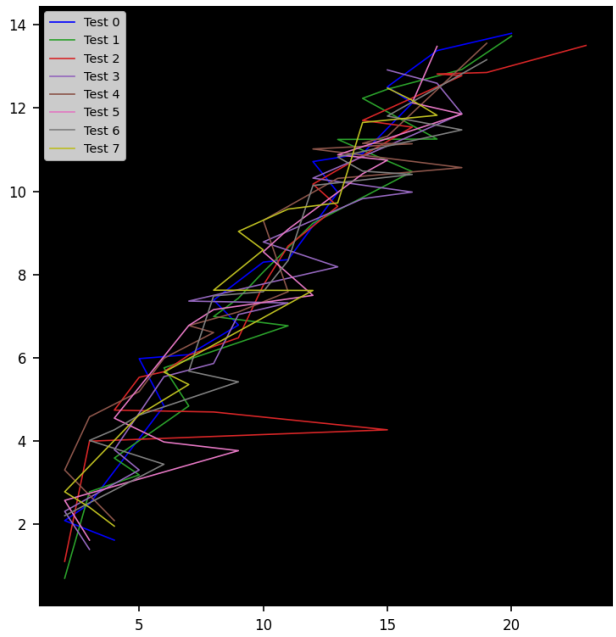

Figure 8.2: Same as Fig. 8.1 but with first arrival of run of length L on Y-axis (log scale), L on X-axis

Failures, while relatively rare, happen frequently enough to be a concern for PCG64. But no failure was found for my system (the absent B_0 string in the table). The Python code in section 8.2 allows for full replication. In Table 8.1, the 'max_hits' issue is part of the collision test while 'records' (Figure 8.2) is part of the run test. Both are described below.

The statistical tests of randomness used here cover a small portion of battery suites such as BigCrush, DieHarder, PractRand or NIST. I selected them for their generality, covering multiple tests at once while being relatively independent. I also developed several strong tests not found in these suites, for instance the Weyl test described in chapter 5. Now, here are the tests that I chose to produce Table 8.1, where only failures are reported:

- The frequency test computes the proportion of any substring of length at most `Lmax` in a bit string. Thus it acts as a test of independence in dimensions up to `Lmax`. It is highly correlated to the entropy test. Note that in the string '10001', the number of '00' is 2 according to my test (correct), while the default Python count is 1. This test, in its form, is also known as the equidistribution or block test.
- The autocorrelation test computes the maximum of the absolute value of all autocorrelations up to lag `maxLag`, in any bit string. It summarizes the autocorrelation function and is thus a spectral test. However, I also included the NIST spectral test based on Fourier transforms. PCG64 failed that test a few times.
- The compression test compresses a bit string with the powerful LZMA compressor. It did not manage to compress any of the bit strings tested, whether coming from PCG64 or NPG. It shows that those strings have high entropy, compatible with good randomness.
- The collision test computes the number of substrings of length `L` occurring more than once in a bit string, especially when `L` is large and few collisions are expected. It also returns max_hits, the maximum number of times one of those rare substrings occurs. That's where PCG64 showed some failures.
- The run test computes all the runs of 0 of various lengths (Figure 8.1) and their first arrival in the bit string (Figure 8.2). Test k ($k = 0, \ldots, 7$) corresponds to the bit string B_k in Table 8.1, with B_0 for NPG and the other indices for PCG64, here with $a = 1$, $b = 1991$ and the PCG64 seed set to 42. In most cases,

no anomaly was found, neither for NPG nor PCG64. The case featured here is an exception with B_2 (in red, PCG64) out of range. A future test could compute the correlation between the lengths of successive runs.

- The next-bit test checks if you can predict the next bit in a bit sequence, fast enough while outperforming pure chance. I use deep neural networks (DNNs) for this task, in a way similar to how they are used to predict the next token in large language models. In my tests, DNNs cannot do better than random guessing for bit streams generated by NPG. By contrast PCG64 shows weaknesses on occasions depending on the test configuration

It is easy to build a bit sequence that not only passes all equidistribution tests [Wiki], but also mathematically proved to do so. This is the case for the Mersenne Twister PRNG, a precursor to PCG64, itself a precursor to my NPG system. It is proved to be 623-dimensional equidistributed up to 32 bits accuracy. Even worse, the Champernowne constant [Wiki] is proved to be equidistributed in any dimension for any number of bits. Yet its base-10 digits sequence is 12345678910111213... and it is a normal number in base 10. The equivalent exists in base 2. In base 2, it would fail the pair-correlation test. In base 2 or 10, it fails the Kolmogorov complexity test [Wiki], which has no implementation because it is undecidable. The Copeland–Erdős constant [Wiki] where the numbers 1, 2, 3 and so on are replaced by the prime numbers, is also fully equidistributed. Yet, it would make for a terrible PRNG. In short, the fact that a PRNG is proved to be equidistributed, has little value.

Finally, some claim that the binary digits of π have no consecutive 0 between positions n and $8n$ for n large enough, and thus its digits do not mimic randomness, see here. This argument is wrong: any random bit sequence has no consecutive 0 between positions n and $n+\log_2 n$. In fact, I proved (and I am not the only one) that the binary digits of $\sqrt{2}$ cannot have consecutive 0 between positions n and $2n$. You expect that from a random sequence when n is large enough. And if not satisfied, it would prove that the sequence is not random. Much of the pseudo-science linked to random bits is spread by people who know little about the topic, despite advanced degrees in computer science and similar fields.

8.2 Compute time and Python code

The compute time is summarized in Table 8.2. A good benchmark is when $N = 2000$, generating about 10^9 bits in 0.10 seconds with PCG64, a standard also reported elsewhere. Unlike PCG64 (linear in time), there is an optimum for NPG. On my laptop, producing 8 bit strings each with $N = 2000$ is much faster than producing a single string with $N = 4000$ even though the total number of bits generated is the same in both cases. Also, I optimized PCG64 for speed, as follows:

```
size = total_bits
block_size = 32
nsamples = 20
start = time.perf_counter()
chunk_size = size // (nsamples * block_size)
for sample in range(nsamples):
   bits = rng.integers(0, 2**block_size, size = chunk_size, dtype=np.uint32)
end = time.perf_counter()
```

I know discuss the code. First, if you perform a large number of tests, some will lead to false positives, just by chance: a truly random sequence erroneously flagged as abnormal. The issue comes from using the same high p-value too many times. This phenomenon is described in most statistical textbooks, along with solutions to address it. Anyway, this is not the case here.

N	Total bits	a	b	NPG	PCG64
1000	166,725,233	1	1991	0.005550	0.016639
1000	166,668,496	1	1	0.003406	0.013890
1000	169,528,867	53	31	0.007111	0.017117
2000	1,333,336,996	1	1	0.022635	0.106265
3000	4,500,005,496	1	1	0.115812	0.338585

Table 8.2: Compute time in seconds, NPG vs PCG64

It is worth checking if the PCG64 failures are linked to the odd number of bits in Table 8.1, dictated by the choice of a and b. Those who built PCG64 probably tested it mostly with more standard values. The spectral

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]. 13, 15, 24, 33, 46

[2] Christoph Aistleitner et al. Normal numbers: Arithmetic, computational and probabilistic aspects. 2016. Workshop [Link]. 13, 24, 33, 46

[3] Adel Alamadhi, Michel Planat, and Patrick Solé. Chebyshev's bias and generalized Riemann hypothesis. *Preprint*, pages 1–9, 2011. arXiv:1112.2398 [Link]. 85

[4] Gökalp Alpan and Maxim Zinchenko. Lower bounds for weighted Chebyshev and orthogonal polynomials. *Preprint*, 2024. arXiv:2408.11496v [Link]. 57

[5] David Bailey, Jonathan Borwein, and Neil Calkin. *Experimental Mathematics in Action*. A K Peters, 2007. 12, 24, 33, 46, 62

[6] Verónica Becher, A. Marchionna, and G. Tenenbaum. Simply normal numbers with digit dependencies. *Mathematika*, 69:988–991, 2023. arXiv:2304.06850 [Link]. 13, 24, 33, 46

[7] Frederik Broucke. On zero-density estimates for Beurling zeta functions. *Preprint*, pages 1–24, 2024. arXiv:2409:1051v1 [Link]. 78

[8] James Dolan. Carrying is a 2-cocycle. *Preprint*, pages 1–9, 2023. [Link]. 13, 24, 33, 46

[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]. 71

[10] Taylor Dupuy and David E. Weirich. Bits of in binary, Wieferich primes and a conjecture of Erdős. *Journal of Number Theory*, 158:268–280, 2016. [Link]. 103

[11] 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]. 13, 24, 33, 46

[12] P. M. Gauthier. Approximating the Riemann zeta-function by polynomials with restricted zeros. *Canadian Mathematical Bulletin*, 62(3):475–478, 2018. [Link]. 132

[13] Vincent Granville. *Stochastic Processes and Simulations: A Machine Learning Perspective*. MLT, 2022. [Link]. 78, 131

[14] Vincent Granville. *Synthetic Data and Generative AI*. MLT, 2022. [Link]. 78

[15] Vincent Granville. *Gentle Introduction To Chaotic Dynamical Systems*. MLT, 2023. [Link]. 8, 11, 12, 13, 15, 16, 19, 24, 33, 45, 46, 58, 62, 70, 71, 78, 79, 84, 85, 98, 101

[16] Vincent Granville. *Building Disruptive AI & LLM Technology from Scratch*. MLT, 2024. [Link]. 16, 24, 33, 46, 127

[17] Vincent Granville. *State of the Art GenAI & LLMs, Creative Projects & Solutions*. MLT, 2024. [Link]. 21, 85

[18] Vincent Granville. *Statistical Optimization for AI and Machine Learning*. MLT, 2024. [Link]. 79

[19] Vincent Granville. *Synthetic Data and Generative AI*. Elsevier, 2024. [Link]. 83

[20] Vincent Granville. *Blueprint: Next-Gen Enterprise RAG & LLM 2.0 – Nvidia PDFs Use Case*. 2025. MLT [Link]. 127

[21] Vincent Granville. Simple, efficient, secure, accurate enterprise AI xLLM 2.0 architecture & operating system. 2025. BondingAI internal report `bdai-scores.pdf`, July 2025. 127

[22] Vincent Granville. *No-Blackbox, Secure, Efficient AI and xLLM Solutions*. MLT, 2026. [Link]. 96, 101, 127, 132

[23] Vincent Granville and Richard L Smith. Disaggregation of rainfall time series via Gibbs sampling. *NISS Technical Report*, pages 1–21, 1996. [Link]. 131

[24] Emil Grosswald. Oscillation theorems of arithmetical functions. *Transactions of the American Mathematical Society*, 126:1–28, 1967. [Link]. 85

[25] Adam J. Harper. Moments of random multiplicative functions, II: High moments. *Algebra and Number Theory*, 13(10):2277–2321, 2019. [Link]. 86

[26] 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]. 86

[27] Adam J. Harper. Almost sure large fluctuations of random multiplicative functions. *Preprint*, pages 1–38, 2021. arXiv [Link]. 86

[28] C. P. Hughes and A. Nikeghbali. The zeros of random polynomials cluster uniformly near the unit circle. *Compositio Mathematica*, 144:734–746, 2008. [Link]. 109

[29] 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]. 72

[30] 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]. 33, 46

[31] Marcus Michelen and Oren Yakir. Limit law for root separation in random polynomials. *Preprint*, pages 1–77, 2025. arXiv:2505.02723 [Link]. 109

[32] 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]. 13, 24, 33, 46

[33] Mohammadamin Moradi et al. Data-driven model discovery with Kolmogorov-Arnold networks. *Preprint*, pages 1–6, 2024. arXiv:2409.15167 [Link]. 25, 33, 46

[34] 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]. 24, 33, 46

[35] Alan Oppenheim and Ronald W. Schafer. *Discrete–Time Signal Processing*. Prentice Hall, 1999. 134

[36] Michel Planat and Patrick Solé. Efficient prime counting and the Chebyshev primes. *Preprint*, pages 1–15, 2011. arXiv:1109.6489 [Link]. 85

[37] Eric S. Rowland. Regularity versus complexity in the binary representation of 3^n. *Complex Systems*, 18:367–377, 2009. [Link]. 103

[38] Frank Ruskey. Generating functions for the digital sum and other digit counting sequences. *Journal of Integer Sequences*, 12:1–9, 2009. [Link]. 95

[39] Klaus Schiefermayr and Maxim Zinchenko. Norm estimates for Chebyshev polynomials, i. *Journal of Approximation Theory*, 265, 2021. [Link]. 57

[40] Jan-Christoph Schlage-Putcha and Jasson Vindas. The prime number theorem for Beurlings generalized numbers – new cases. pages 1–26, 2011. [Link]. 78

[41] Maxwell Schneider and Robert Schneider. Digit sums and generating functions. *Preprint*, pages 1–10, 2018. arXiv:1807.06710 [Link]. 95

[42] Terence Tao. Limit law for root separation in random polynomials. *Tao's blog*, 2013. [Link]. 109

[43] Terence Tao. Biases between consecutive primes. *Tao's blog*, 2016. [Link]. 85

[44] Van Vu Terence Tao. Limit law for root separation in random polynomials. *Preprint*, pages 1–56, 2014. arXiv:1307.4357 [Link]. 109

[45] Yury V. Tiumentsev and Mikhail V. Egorchev. *Neural Network Modeling and Identification of Dynamical Systems*. Elsevier, 2019. 24, 33, 46

[46] Chukwudubem Umeano and Oleksandr Kyriienko. Ground state-based quantum feature maps. *Preprint*, pages 1–8, 2024. arXiv:2024.07174 [Link]. 13, 24, 33, 46

[47] Joseph Vandehey. On the binary digits of $\sqrt{2}$. *Preprint*, pages 1–6, 2017. arXiv:1711.01722 [Link]. 12, 24, 33, 46, 62

[48] Troy Vasiga and Jeffrey Shallit. On the iteration of certain quadratic maps over GF(p). *Discrete Mathematics*, 277:219–240, 2004. [Link]. 24, 33, 45

[49] Henry S. Warren. *Hacker's Delight*. Addison-Wesley Professional, second edition, 2012. 13, 24, 33, 46

[50] Nhan D. V. Nguyen Yen Q. Do. Limit law for root separation in random polynomials. *Electronic Journal of Probability*, 30:1–37, 2025. [Link]. 109

[51] 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]. 24, 33, 46

Index

p-adic numbers, 6, 115
p-adic valuation, 75
p-value, 119

agent (artificial intelligence), 25
agent-based modeling, 34
AI–generated art, 111
algebraic number, 33
analytic continuation, 83
analytic functions, 82
asymptotic periodicity, 96
attractor distribution, 12
autoconvolution (see self-convolution), 19
autocorrelation, 70
autocorrelation function, 34, 97, 118
    time lag, 43

basin of attraction, 79
Benford's law, 103
Bernoulli convolution, 15
Bernoulli distribution, 95
Bernoulli process, 45
beta distribution, 12
Beurling prime, 74
bifurcation phase, 20
binomial coefficients, 25
    central coefficient, 15
blancmange curve, 13, 24, 33, 46
BQ (base quadratic map), 42
Brownian motion, 20, 34, 84, 98

canonical form (strings), 6
Cantor set, 46
carry digit function, 13, 24, 33, 46
Catalan numbers, 15
causal model
    non-causal, 132
central-limit theorem, 20
Champernowne constant, 119
chaotic convergence, 94
chaotic phase, 20, 42
characteristic polynomial, 96
Chebyshev polynomials, 56, 57
Chebyshev's bias, 83, 85, 86
checksum, 111
class (string or number), 7
cocycle, 13, 24, 33, 46
Collatz conjecture, 96
collision test, 118
combinatorial complexity, 117
compression, 118
computational intelligence, 25
congruential class, 21, 31, 43
conjugate maps, 19
convergence
    absolute, 74
    acceleration, 78
    chaotic convergence, 94
    conditional, 83
    fractal convergence, 94
convolution, 7
    auto-convolution, 7
    iterated self-convolution, 29
    self-convolution, 19, 29
convolution product
    deconvolution, 133
    discrete Gaussian, 132
Copeland–Erdős constant, 119
coprime integers, 15
correlation, 71
    cross-correlations, 15, 34
    empirical correlation, 15, 70
Cramér's conjecture, 78
cryptographically secure, 117
cubic equation, 96
curve fitting, 86, 134

deconvolution, 133
deep neural network, 24, 33, 46, 119
deterministic AI, 115
deterministic PRNG, 117
digit count, 13, 19
digit sum function, 13, 15, 19, 24, 33, 42, 46, 94, 105
    adjusted digit sum, 30, 38, 42
Dirichlet L-function, 83
Dirichlet character, 83
Dirichlet eta function, 74, 132
Dirichlet series, 83
Dirichlet's theorem, 87
dyadic map, 11, 12, 15, 19, 29, 56
dyadic rational, 32, 33, 102, 106
dynamical systems, 41, 79
    chaotic, 52, 79, 100
    quadratic map, 58
    state space, 58

empirical distribution function, 101
empirical probability density function, 98
entropy test, 118
equidistribution, 58, 118, 119
ergodic mean, 101
ergodicity, 12, 19, 100

ergodic mean, 100
Euler product, 74, 82
Euler's transform, 131
Euler-Mascheroni constant, 130
experimental math, 82

filter
blurring, 134
fixed point, 33, 52
flat polynomial, 107
fractal, 45
fractal convergence, 94
fractional part function, 15
functional equation, 12, 101

generating function, 15, 48, 95, 105
generative AI (GenAI), 87
geometric mean
complex numbers, 57
goodness-of-fit, 86
gradient descent
stochastic, 96

Hamming weight, 13, 24, 33, 46, 94
high-performance computing, 57, 116
homeomorphism, 11, 19

invariant measure, 12, 19, 29, 72, 100, 101
inverse transform, 29
irrational number, 15

kernel method
Gaussian kernel, 132
Kolmogorov complexity, 119

Laurent series, 134
law of the iterated logarithm, 20
leading digits, 103
Littlewood polynomial, 107
Littlewood's oscillation theorem, 85
LLM (large language model), 24, 34
logarithmic capacity, 57
logistic map, 12, 19, 24, 29, 33, 41, 45
loss function
adaptive, 96
LZMA compression, 118

Mandelbrot set, 41
map (dynamical systems), 12
Mersenne Twister, 117, 119
Mertens' theorem, 130
multi-branch function, 21
multiplicative function (random), 86
multiplicative order, 13, 21
Möbius function, 108

Newman polynomial, 107
next-bit test, 119
normal number, 12, 19, 24, 33, 34, 46, 49, 56, 103, 107, 109, 119
simply normal, 45
strongly normal, 58

NPG (non-periodic PRNG), 115
number representation, 6

odometer map, 15
orbit (dynamical systems), 52, 105

partition function, 116
PCG64, 117
PCG64DXSM, 117
period (rational numbers), 15
periodicity
asymptotic, 96
Perron-Frobenius operator, 101
pigeonhole principle, 49, 101
Pisot number, 107
prime race, 85
primorial, 30, 32, 35
PRNG (pseudo-random generator), 24, 33, 34, 46, 115
cryptographically secure (CSPRNG), 117
deterministic, 117
Mersenne Twister, 117
non-periodic, 115
NPG, 115
PCG64, 117
PCG64DXSM, 117
PRNG state, 116
strong PRNG, 15, 71, 117
tests of randomness, 118
Python library
Gmpy2, 34
MPmath, 82, 86
PrimePy, 86
Scipy, 86

quadratic convergence, 33
quadratic dynamical systems, 24, 29, 33, 45
quadratic irrational, 15
quadratic map, 41, 58
base quadratic map, 41
quantization, 103
quantum cryptography, 13, 24, 33, 46
quantum derivative, 21, 84
quantum dynamics, 31, 34, 42
quantum function, 21, 30
quantum map, 13, 24, 33, 46
quantum state, 24, 43, 78, 85, 96
quantum system
quantum convergence, 77
sub-quantum states, 74

R-squared, 86
Rademacher distribution, 86
Rademacher function (random), 86
random number generation (see PRNG), 58
random polynomial, 108
random walk, 20, 84
randomness test, 71
reciprocal distribution, 12, 19, 29
replicability, 71, 111, 115, 116
residue class, 30, 96
Riemann Hypothesis, 78, 82, 131
Riemann zeta function, 74, 82, 105

root–finding algorithm, 111
run (of same digit), 15
run test, 118

Salem number, 107
scaling factor, 86
seed (dynamical systems), 12
seed (PRNG), 72, 115, 116
seed string, 8, 19, 29, 42
    reverse seed, 30
Shapiro polynomial, 107
signal processing, 132
slow growth function, 48
spectral radius, 58
spectral test, 118
spectral view, 43
square root (of a string), 7
square root operator, 29
state (PRNG), 116
state space, 58
stationary process
    non-stationary, 43
Stern's diatomic series, 96
Stirling numbers, 95
stochastic gradient descent, 96
string class, 6
string convergence, 7
swarm optimization, 96
symbolic mathematics, 96
synthetic data, 34
synthetic function, 87
synthetic numbers, 74, 78

Thue-Morse sequence, 107
time series
    disaggregation, 131
transfer function, 134
trinomial coefficients, 132
truncation, 6, 19

undecidable, 119
universal property, 111

Weyl criterion (normality), 56, 58
Weyl's equidistribution theorem, 102

Z-transform, 134