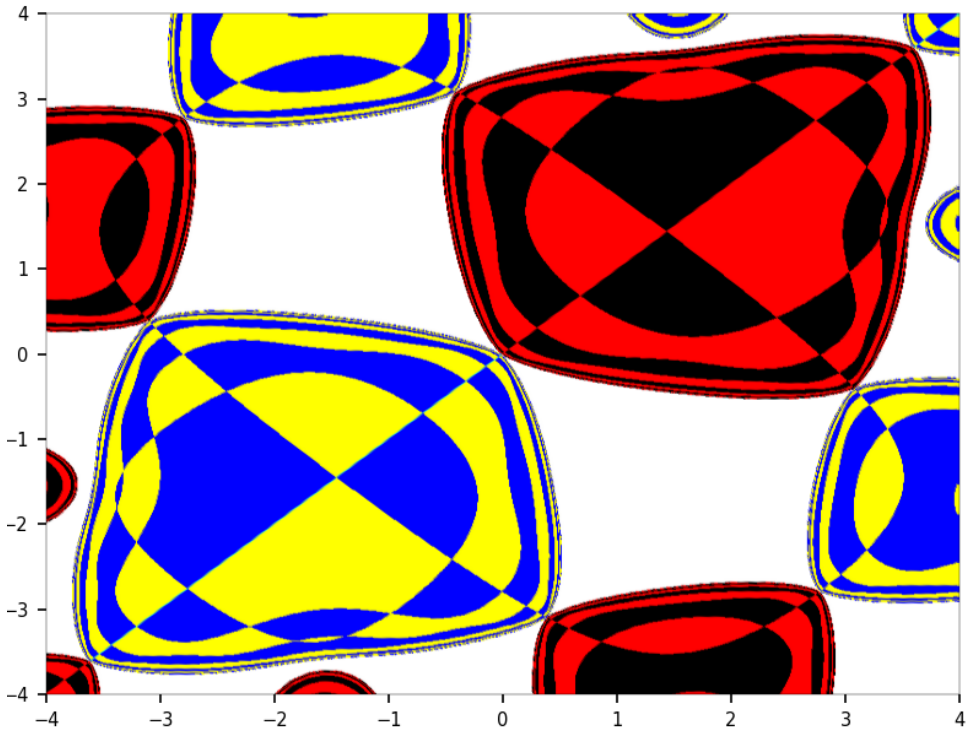
Gentle Introduction To Chaotic Dynamical Systems



Vincent Granville, Ph.D. | www.MLTechniques.com | Version 1.0, February 2023

Contents

Chapter 2

Introduction to Discrete Chaotic Dynamical Systems

In chapter 1, I introduced the concept of invariant measure satisfying a stochastic integral equation, in the context of stochastic discrete dynamical systems such as reflected random walks. In this chapter I establish a general formula to find the invariant measure – also called invariant or equilibrium distribution – for a class of well-known models. I then discuss several examples where an exact solution is known. I also discuss the inverse problem: finding a dynamical system satisfying a given invariant measure. The last section deals with the two-dimensional case.

2.1 Definitions, properties, and examples

One fascinating problem is the following. Given a random number, what is the probability distribution of the coefficients of its continued fraction expansion? For instance, we have

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1 + \cfrac{1}{292 + \cfrac{1}{1 + \ldots}}}}}$$

In this case the first five coefficients are $7, 15, 1, 292, 1$. The method presented here provides an immediate solution to this historical problem: the Gauss–Kuzmin distribution. In particular, the frequency of the value $n = 1, 2$ and so on among the coefficients in the continued fraction expansion is

$$\pi_n = 2\log_2(n+1) - \log_2(n+2) - \log_2 n > 0. \tag{2.1}$$

The expectation is infinite, but the logarithm of these coefficients have a finite expectation. Note that using the first three coefficients, you get the very good approximation $\pi \approx 355/113 \approx 3.15159292$. This is due to the fact that the fourth coefficient 292 is unusually large. The same method allows you to solve many other problems not related to continued fractions, with various applications.

I will dig deeper into this example later in this section. But first, I need to introduce the general framework, some terminology, and more examples. I start with the one-dimensional case. A discrete dynamical system [Wiki] is an iteration $X_{k+1} = g(X_k)$, where the initial value X_0 is called the seed. The index k is usually referred to as the time or the evolution parameter. The system maps a domain $D \in \mathbb{R}$ (typically an interval) onto itself. That is, if $x \in D$, then $g(x) \in D$. Thus X_0, X_1 and so on are all in D, and $g(D) = D$. Typically, the function g is not invertible to allow for mixing. The domain D is called the state space, and the function g is called a mapping.

One of the most well-known examples is the logistic map [Wiki]. In this case, $g(x) = rx(1-x)$ where $0 < r \leq 1$ is a parameter and $D = [0, 1]$. There is a time-continuous version of it, governed by a differential equation rather than a recurrence. It is fully chaotic – the case we are interested in – when $r = 4$. It has properties shared by all the systems studied in this book.

2.1.1 Properties of discrete chaotic dynamical systems

The systems studied in this book have the following properties. They are stated in the context of the logistic map as an illustration.

- For the vast majority of seeds X_0, the sequence $\{X_k\}$ is dense in D: it approaches any value in D arbitrarily closely if k is large enough. However, some seeds (infinitely many yet very rare) produce a periodic sequence and thus result in non-chaotic behavior. An example is $X_0 = \frac{1}{2}$ for the logistic map. Such seeds are called bad seeds. Their set is small enough that it has zero Lebesgue measure [Wiki].
- The sequence can not be inverted: given an observed value X_k, there are always two values X_{k-1} and X'_{k-1} such that $X_k = g(X_{k-1}) = g(X'_{k-1})$. In some cases other than the logistic map, you actually have more than two potential previous iterates leading to the same X_k.
- The sequence $\{X_k\}$ is ergodic [Wiki]: the distribution and other statistical properties (mean, variance, autocorrelations) attached to the X_k's can be approximated with arbitrary precision using one single infinite sequence (called realization or instance) starting with a good seed X_0, or using infinitely many realizations, each starting with a good seed – say a different random seed for each realization – but using only the first few iterates in each realization.
- The limiting distribution F for the X_k's (previously referred to as the invariant measure) is solution to the following functional equation: $F(x) = F(g(x))$. Here $F(x) = P(X_k < x)$ if you were to start with a random seed X_0 uniformly distributed on D, and let $k \to \infty$. This is true only if you start with a good seed. The probability that a random seed is a good one is 100%. Good seeds lead to a non-periodic sequence $\{X_k\}$, chaotic yet well behaved.
- Computation of X_k based on the recurrence $X_k = g(X_{k-1})$ is bound to generate totally wrong values after as little as 45 iterations. Due to the ergodicity of the sequence, it is not a problem in many cases: it amounts to starting with a brand new seed every 45 iterations or so. It is an issue when computing long-range autocorrelations.
- Two sequences $\{X_k\}, \{X'_k\}$ starting with two different good seeds X_0, X'_0 that are very close to each other, will over time completely diverge. This property is sometimes used to define chaos. It also explains the previous point.

One important result to find an exact solution (assuming there is one) to the functional equation $F(x) = F(g(x))$, and thus directly identify the main invariant measure, is the following. Here I assume that $g(x) = p(x) - \lfloor p(x) \rfloor$ where the brackets represent the integer part function. In addition, $p(x)$ is assumed to be positive, monotonic, continuous and decreasing (thus bijective and invertible) with $p(1) = 1$ and $p(0) = \infty$. Thus, $D = [0, 1]$. Despite the restrictions, it covers a large class of important dynamical systems, for instance the Gauss map [Wiki] if $p(x) = 1/x$ leading to the continued fractions [Wiki] discussed in the introduction. In this case, $g(0) = 0$.

In a number of important and interesting cases, there is a function $r(\cdot)$ such that the invariant distribution F with support domain $D = [0, 1]$ can be written as $F(x) = r(x + 1) - r(1)$. Thus $r(x)$ must be increasing on $[1, 2]$ and $r(2) = 1 + r(1)$. Not any function can be an invariant distribution, also called attractor distribution in probability theory. Define $R(x) = r(x + 1) - r(x)$. Then we can retrieve $p(x)$ (under some conditions) using the formula

$$p(x) = R^{-1}\Big(r(x + 1) - r(1)\Big) = R^{-1}(F(x)). \tag{2.2}$$

The approach here is backward: we solve the inverse problem where F is known and we try to retrieve $p(x)$. But in the end, it helps build a table of dynamical systems with known solution, in the same way that a table of integrals helps solve a number of integrals.

If you only know $p(x)$, to retrieve $F(x)$ or its derivative $f(x)$, you need to solve the following functional equation, whose unknown is the function f.

$$f(x) = \sum_{k=1}^{\infty} \left| \frac{dF(q(x + k))}{dx} \right| = \sum_{k=1}^{\infty} |q'(x + k)| f(q(x + k)). \tag{2.3}$$

Here $q(x) = p^{-1}(x)$. Note that $R(x) = F(q(x))$ or alternatively, $R(p(x)) = F(x)$, with $p(q(x)) = q(p(x)) = x$. Also, $0 \le x \le 1$.

In practice, you get a good approximation if you use the first 1000 terms in the sum. Typically, the invariant density f is bounded, and the weights $|q'(x + k)|$ are decaying relatively fast as k increases. The theory behind this is based on the transfer operator [Wiki], and related to the Perron-Frobenius theorem [Wiki]. The invariant density is the eigenfunction of the transfer operator, corresponding to the eigenvalue 1. A much simpler approach is discussed in Exercise 4. Also, if x is replaced by a vector (for instance, if working with bivariate dynamical systems), the above formula can be generalized, involving two variables x, y, and the derivative of the (joint) distribution is replaced by a Jacobian.

Exercise 4 *Establishing the functional equation* – Assume $p(x) = \lambda/x^{\alpha}$ with $\alpha, \lambda > 0$. Prove that the invariant density f satisfies (2.3).

Solution
Let x' be a back image of x, in the sense that $g(x') = x$. For any $x \in [0,1]$, there are infinitely many x'_k such that $g(x'_k) = x$. I denote them as

$$q_k(x) = q(x+k) = \left(\frac{\lambda}{x+k}\right)^{1/\alpha}, \quad k = 1, 2, \cdots$$

where $q(x) = p^{-1}(x)$ and $q_k(x) = q(x+k)$ by definition. Now the back image of the whole interval $[x, x+\epsilon]$ with $\epsilon \approx 0$ is obtained as

$$T_\epsilon(x) = \bigcup_{k=1}^{\infty} \Big[q_k(x+\epsilon), q_k(x)\Big].$$

The length of the k-th interval on the right-hand side is approximately $\epsilon|q'_k(x)|$, where q' denotes the derivative with respect to x. Thus, since $F([x, x+\epsilon]) = F(g([x, x+\epsilon])) = F(T_\epsilon(x))$, dividing by ϵ and letting $\epsilon \to 0$, we have the following fundamental functional equation:

$$f(x) = \sum_{k=1}^{\infty} |q'_k(x)| \cdot f(q_k(x))$$

where f is the derivative of F. ∎

In particular, if $\alpha = \lambda = 1$ (the standard continued fraction), it is easy to verify that $F(x) = \log_2(1+x)$ satisfies the functional equation. This is a well known result, and the proof provided here is probably one of the shortest ones. An immediate consequence is Formula (2.1) involving the Gauss–Kuzmin distribution [Wiki]. This discrete distribution represents the probability that $\lfloor p(X_k) \rfloor = n$ for $n = 1, 2$ and so on, starting with a random seed X_0, and $k \to \infty$. It has an infinite expectation. If $\alpha < 1$ the resulting distribution has a finite expectation, and if $\alpha \leq \frac{1}{2}$, its variance is finite. The strictly positive integer $\lfloor p(X_k) \rfloor$ representing the k-th coefficient in the continued fraction expansion of X_0, is also called the k-th digit in the system in question.

2.1.2 Shortlist of dynamical systems with known solution

In section 1.4, I discussed a system with an exact solution to the functional equation: reflected random walks. I discuss more systems with an exact solution in the next chapters. The logistic map is one of them. Here I focus on systems where the exact solution can be derived from the results in section 2.1.1. Before starting, I introduce the concept of digits and numeration system attached to the chaotic dynamical systems in question. It is important and used throughout this book, with interesting applications in cryptography and random number generation.

2.1.2.1 Digits and numeration system attached to a dynamical system

The numeration systems described here are a generalization of the classic ones. Traditional digits in base b are themselves attached to the the map $g(x) = bx - \lfloor bx \rfloor$, known as the ten-fold map if $b = 10$, and the dyadic map [Wiki] if $b = 2$. I discuss theses cases, including when b is a bivariate or irrational base, later in this book. The full infinite sequence of digits (also called codes) allows you to retrieve the seed X_0, that is, the number that it represents.

Let us now focus on the dynamical systems described in section 2.1.1 using the map $g(x) = p(x) - \lfloor p(x) \rfloor$. The k-th digit a_k attached to the seed $X_0 = x_0$ is defined as $a_k = \lfloor p(X_k) \rfloor$ for $k = 0, 1$ and so on. Even though the dynamical systems discussed here are not invertible, it is possible to compute x_0 if you only known its digits, thanks to the fact that $p(x)$ is invertible. To compute x_0 based on its digits, start with N large (say $N = 20$), $u_N = 0$, and proceed iteratively backwards starting with $k = N-1$, back down to $k = 0$, using the recursion

$$u_k = q(u_{k+1} + a_k) - \lfloor q(u_{k+1} + a_k) \rfloor.$$

Here $q(x) = p^{-1}(x)$. If you start with $N = \infty$, then $x_0 = u_0$. For the Gauss map, the digits are simply the coefficients that appear in the continued fraction expansion $x_0 = 1/(a_0 + 1/(a_1 + 1/(a_2 + \dots)))$. Note that $x_0 \in [0,1]$ and the digits can take on any integer value greater than 1. For the Gauss map, numbers x_0 that are rational are excluded. Similar restrictions apply to other maps. The expectation of the digits (the average of all the digits of x_0 for any x_0 but a set of Lebesgue measure zero), is finite if and only if $\int_0^\epsilon p(x)f(x)dx < \infty$ for any $\epsilon \in]0,1]$. Here $f(x)$ is the invariant density, that is, the derivative of the invariant distribution. Likewise, the variance of the digits is finite if and only if $\int_0^\epsilon p^2(x)f(x)dx < \infty$ for any $\epsilon \in]0,1]$. If the expectation and

variance are finite, then it is possible to compute the autocorrelation between two successive digits. Obviously, both the expectation and variance are infinite for the digits of the Gauss map (continued fractions). Below we discuss a few dynamical systems, some with finite and some with infinite expectation / variance for the digits.

2.1.2.2 Examples with exact solution in closed form

Besides the Gauss map associated to continued fractions, I provide 5 examples with exact solution in closed form for the invariant distribution F. Again, $a_k = \lfloor p(X_k) \rfloor$ is the k-th digit attached to the system in question, as discussed in section 2.1.2.1. I use the functions $r(x)$, $R(x)$ and $q(x) = p^{-1}(x)$ introduced in section 2.1.1.

Example 1

Let $r(x) = -2(x+1)/x$. Then

$$F(x) = \frac{2x}{x+1}, \quad R(x) = \frac{2}{x(x+1)}, \quad p(x) = \frac{-1+\sqrt{5+4/x}}{2}, \quad q(x) = \frac{4}{(2x+1)^2-5}.$$

In this case

$$P(a_k = n) = F(q(n)) - F(q(n+1)) = \frac{4}{n(n+1)(n+2)}, \quad \mathrm{E}[a_k] = \sum_{n=1}^{\infty} kP(a_k = n) = 2.$$

Example 2

Let

$$r(x) = \lambda \log \frac{\alpha+x}{\alpha+1} \text{ with } \lambda = \left(\log \frac{\alpha+2}{\alpha+1}\right)^{-1}.$$

Then

$$F(x) = \lambda \log\left(\frac{\alpha+x+1}{\alpha+1}\right), \quad p(x) = \frac{\alpha+1}{x} - \alpha, \quad q(x) = \frac{\alpha+1}{\alpha+x}.$$

Here $\alpha \geq 0$. If $\alpha = 0$, then this is just the Gauss map and we are dealing with standard continued fractions. Thus this example deals with generalized continued fractions.

Example 3

Let

$$r(x) = \lambda \log \frac{x(x+1)}{2} \text{ with } \lambda = \frac{1}{\log 3}.$$

Then

$$F(x) = \lambda \log \frac{(x+1)(x+2)}{2}, \quad p(x) = \frac{4}{x(x+3)}, \quad q(x) = \frac{3}{2}\left(-1+\sqrt{1+\frac{16}{9x}}\right).$$

Example 4

Let

$$r(x) = \lambda \alpha^{w(x)} \text{ with } \lambda = \frac{1}{\alpha^4 - \alpha^2}, \text{ and } w(x) = 2^x.$$

Then

$$F(x) = \lambda(\alpha^{w(x+1)} - \alpha^2), \quad p(x) = \log_2 \log_\alpha \left(\frac{1-\sqrt{1+4(w^2-\alpha^2)}}{2}\right).$$

Here $0 < \alpha < \sqrt{2}/2$. Other values of α don't work. The more simple case $r(x) = \lambda\alpha^x$ leads to nowhere.

Example 5

Let ψ be a positive, monotonic decreasing function with $\psi(0) = \infty$. Let

$$r(x) = -\frac{1}{\psi(1)} \cdot \sum_{k=0}^{\infty} \psi(x+k), \quad \int_1^{\infty} \psi(x)dx < \infty.$$

Then

$$F(x) = \frac{1}{\psi(1)} \cdot \sum_{k=1}^{\infty} \Big(\psi(k) - \psi(x+k)\Big), \quad R(x) = \frac{\psi(x)}{\psi(1)}, \quad p(x) = \psi^{-1}\left(\sum_{k=1}^{\infty} \Big[\psi(k) - \psi(x+k)\Big]\right).$$

Also, $P(a_k = n) = (\psi(n) - \psi(n+1))/\psi(1)$, for $n = 1, 2$ and so on. An interesting case involving the Hurwitz and Riemann zeta functions [Wiki] is when $\psi(x) = x^{-s}$ with $s > 1$. It corresponds to a non-standard version of the Hurwitz map.

2.1.2.3 Gauss map: expectation, variance and autocorrelation

For the Gauss map $X_{k+1} = 1/X_k - \lfloor 1/X_k \rfloor$, the invariant distribution obtained in Exercise 4 is $F(x) = \log_2(1+x)$ with $0 \le x \le 1$. The invariant density $f(x)$ is the derivative of the distribution in question. The expectation and variance of X_k are easy to compute, for instance $\mathrm{E}[X_k] = \int_0^1 x f(x)dx$. The covariance $\mathrm{Cov}[X_k, X_{k+1}] = E[X_k X_{k+1}] - E[X_k]E[X_{k+1}]$ is a bit more delicate:

$$\mathrm{E}[X_k X_{k+1}] = \int_0^1 x\left(\frac{1}{x} - \left\lfloor \frac{1}{x} \right\rfloor\right) f(x)dx = 1 - \int_0^1 x \left\lfloor \frac{1}{x} \right\rfloor f(x)dx.$$

I tried Mathematica to evaluate the rightmost integral, to non-avail. Manual computations lead to

$$\int_0^1 x \left\lfloor \frac{1}{x} \right\rfloor f(x)dx = \frac{1}{\log 2} \sum_{n=1}^{\infty} \left[\frac{1}{n+1} - n \log\left(\frac{(n+1)^2}{n(n+2)} \right) \right] = \frac{\gamma}{\log 2}$$

where γ is the Euler–Mascheroni constant. Putting everything together (I spare you the details), the lag-1 autocorrelation between X_k and X_{k+1}, that is the limit of the empirical lag-1 autocorrelation computed on the first k iterates as $k \to \infty$, is

$$\rho = 2 \cdot \frac{(2-\gamma)\log 2 - 1}{3\log 2 - 2} \approx -0.347.$$

This is confirmed by empirical evidence, and it is true for almost all $x_0 \in [0, 1]$ (in the Lebesgue sense). Exceptions to the rule include numbers such as $x_0 = \sqrt{2} - 1$, the golden ratio, rational and quadratic irrational numbers. By comparison, the lag-n autocorrelation for the multiplicative system $X_{k+1} = bX_k - \lfloor bX_k \rfloor$ attached the the numeration system in integer base $b > 1$, is b^{-n}. The additive system $X_{k+1} = X_k + \alpha - \lfloor X_k + \alpha \rfloor$ with α irrational, exhibits very strong long-range autocorrelations, making it one of the least chaotic dynamical systems among the chaotic ones. Unlike the Gauss map or the dyatic map ($b = 2$), it does not have exceptions (that is, bad seeds or numbers x_0 not following the dominant invariant distribution) because its invariant distribution is unique. The exceptions (bad seeds) to the dyatic map are all the numbers x_0 that are not normal [Wiki] in base 2.

2.1.3 Exercises: Gauss map and continued fractions

The exercises in this section focus on addressing the numerical instability of the sequence $\{X_k\}$, the bivariate digits of a bivariate dynamical system, and how Khinchin's constant related to the coefficients of continued fractions, is obtained.

Exercise 5 *A bivariate numeration system* – How would you define the digits (a_k, b_k) of a seed $(X_0, Y_0) = (x_0, y_0)$ in the following bivariate dynamical system, and reconstruct (x_0, y_0) based on its digits?

$$X_{k+1} = \frac{1}{Y_k} - \left\lfloor \frac{1}{Y_k} \right\rfloor, Y_{k+1} = \frac{1}{X_k} - \left\lfloor \frac{1}{X_k} \right\rfloor.$$

Solution
The orbit (Y_k, Y_k) is dense on the unit square for almost all initial conditions (X_0, Y_0). Of cause the marginals of the invariant joint distribution are just the invariant distributions of the 1D case for such a basic system. Yet it is not obvious to obtain the full invariant joint distribution and understand its intricacies. In short, $F(x, y) \neq F(x)F(y)$.

The digits of (x_0, y_0) are (a_k, b_k) for $k = 0, 1$ and so on, defined by $a_k = \lfloor 1/Y_k \rfloor$ and $b_k = \lfloor 1/X_k \rfloor$. If you know the digits, you can reconstruct (x_0, y_0) as follows. Start with N large enough, say $N = 20$ if you want about 20 digits of accuracy. Set $(u_N, v_N) = (0, 0)$. Proceed backwards as follows, from $k = N - 1$ down to $k = 0$:

$$u_k = \frac{1}{v_{k+1} + b_k} - \left\lfloor \frac{1}{v_{k+1} + b_k} \right\rfloor, \quad v_k = \frac{1}{u_{k+1} + a_k} - \left\lfloor \frac{1}{u_{k+1} + a_k} \right\rfloor.$$

If $N = \infty$ then $(u_0, v_0) = (x_0, y_0)$. Another bi-dimensional numeration system (an extension of the standard base-b numeration system) is discussed in section 2.1.4.

Exercise 6 *Numerical instability of chaotic dynamical systems* — As discussed in section 2.1.1, the computation of X_k using the iterative map $X_{k+1} = g(X_k)$ quickly results in total loss of accuracy after as little as 45

iterations, as rounding errors propagate exponentially fast. Indeed, the problem is similar to computing the first 1000 binary digits of $\sqrt{2}$ if your machine precision is limited to 32 bits. How would you test the accuracy of your computations, especially when working with very high (yet finite) precision?

Solution
One way assess numerical instability is to pretend – by truncating and rounding the values of X_k – that your machine has a precision of only 3 digits. Then do the same, pretending the precision is 6 digits, then 12 digits. Comparing the results obtained with 3-digit versus 6-digit or 12-digit precision, you can tell how far (that is, how many iterations) you can go to preserve the desired accuracy in the sequence $\{X_k\}$. The idea is to incrementally increase the precision until you see no more improvements. This is the actual limit of your machine or library function that you use. In my case, I found in some older system that increasing the precision from 180 to 200 or 300 digits had no effect. This meant that no matter what is advertised in the documentation (or possibly due to some glitches of my own), the accuracy of my results was limited to 180 digits.

Exercise 7 *Khinchin's constant* – Let a_1, a_2 and so on be the coefficients or digits of the continued fraction expansion of any good seed $x_0 \in [0, 1]$. How do you determine the value of $(a_1 a_2 \cdots a_k)^{1/k}$ as $k \to \infty$?

Solution
The existence and determination of Khinchin's constant [Wiki] is a well-know and fascinating problem. It is a painfully slow process to obtain an empirical estimation and assess its accuracy, but the theory helps solve this problem. Here $P(a_k = n)$ is easy to obtain from the invariant distribution, see (2.1). Let K_0 be the constant in question. We have:

$$\log K_0 = \lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} \mathrm{E}\Big[\log a_i\Big] = \lim_{k\to\infty} \mathrm{E}\Big[\log a_k\Big] = \sum_{n=1}^{\infty} (\log n) P(a_k = n).$$

From there, it is easy to obtain

$$K_0 = \prod_{n=1}^{\infty} \left(1 + \frac{1}{n(n+2)}\right)^{\log_2 n} \approx 2.6854520010.$$

The values of $\log a_k = \log\lfloor p(X_k)\rfloor$ for k between 1 and 20,000 are plotted in Figure 2.1, for $X_0 = \pi$. You can download these values from the "Online Encyclopedia of Integer Sequences" (OEIS), here. The first few ones are shown at the very beginning of section 2.1.

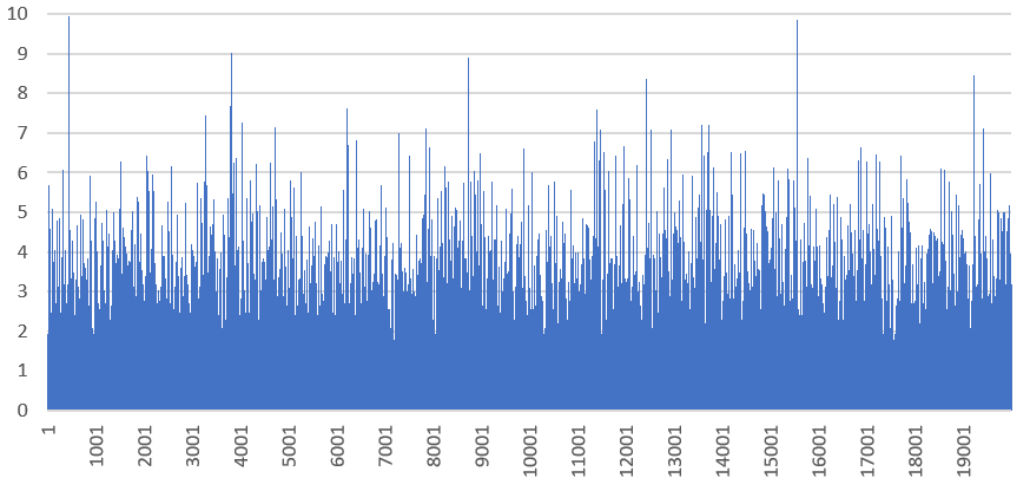


Figure 2.1: First 20,000 coefficients (their logarithm) in the continued fraction of π

2.1.4 Random numbers based on a bivariate numeration system

The following sequence is rather interesting. The initial condition is a bivariate seed (X_0, X_1) and (b_0, b_1) is a vector of integer numbers, positive or negative, called bivariate base. The iterated map is defined as follows:

$$X_{k+2} = b_1 X_{k+1} + b_0 X_k - \lfloor b_1 X_{k+1} + b_0 X_k \rfloor, \quad k = 0, 1, \ldots$$

Here $X_0 \in [0, 1]$ is a constant, and X_1 is uniformly distributed on $[0, 1]$. So in all cases $X_k \in [0, 1]$. If $b_0 = 0$, particular cases include the ten-fold map ($b_1 = 10$) and the dyatic map ($b_1 = 2$) discussed earlier. Non-integer

and irrational bases are discussed later in this book. Also, if $b_0 = 0$, then $a_k = \lfloor b_2 X_k \rfloor$ is the k-th digit of X_1 in base b_1.

In addition, if $b_1 = 0$, the lag-m autocorrelation in the sequence $\{X_k\}$ is b_1^{-m} for $m = 1, 2$ and so on. Furthermore, all vectors (X_k, X_{k+1}) lie in a small, finite number of parallel lines. Thus, unless b_1 is large in absolute value, this sequence is not a good candidate to generate random numbers on $[0, 1]$. The goal here is to show that with two bases b_0, b_1 (instead of one), we can reduce the interdependencies between successive X_k's, of course without completely eliminating them. We can do even better with 3 or more bases, but this is not discussed here. See this post for an example with deeper recursions.

2.1.4.1 Properties of the bivariate numeration system

Before exploring how to increase randomness in the sequence $\{X_k\}$, I want to share a few properties. Let $b_0, b_1 > 0$ and $X_0, X_1 \in [0, 1]$. Define $d_k = b_0 X_k + b_1 X_{k+1} - X_{k+2}$ as the k-th digit of (X_0, X_1) in base (b_0, b_1). The main properties are:

- Two different vectors (X_0, X_1) and (X_0', X_1') can not have the same digits in base (b_0, b_1) if $b_0 > b_1 + 1$ and $b_0, b_1 > 0$. See proof here.
- Assume $b_0 > b_1 + 1$ and $b_0, b_1 > 0$. Let $b = b_0 + b_1$. Then to each $(X_0, X_1) \in [0, 1] \times [0, 1]$ corresponds a unique number $h(X_0, X_1)$ defined by its expansion in base b as follows:

$$h(X_0, X_1) = \sum_{k=0}^{\infty} \frac{d_k}{b^k} \in [0, b].$$

  This creates an order on $[0, 1] \times [0, 1]$: $(X_0, X_1) < (X_0', X_1')$ if and only if $h(X_0, X_1) < h(X_0', X_1')$.
- Unlike in univariate base systems, the distribution of the digits of a random number in base (b_0, b_1) may not be uniformly distributed. For instance, if $b_0 = b_1 = 3$ then the digits 0 and 5 of a random number (X_0, X_1) appear with a frequency of 1/18, the digits 1 and 4 with a frequency of 3/18, and the digits 2 and 3 with a frequency of 5/18.
- Another interesting property, assuming $X_1 \neq 0$, is the following: $X_k = A_k X_1 - \lfloor A_k X_1 \rfloor$ with $A_0 = X_0/X_1$, $A_1 = 1$, and $A_{k+2} = b_1 A_{k+1} + b_0 A_k$. This result is based on empirical evidence. The proof or disproof is left to the reader.
- Finally, all triplets (X_k, X_{k+1}, X_{k+2}) lie on a finite number of parallel planes. The parametric equation for this family of planes is $b_0 x + b_1 y - z = d$, where the parameter d is an integer. This parameter can only take a few different values, depending on b_0 and b_1. The values in question correspond to the potential values that the digits of any arbitrary (X_0, X_1) can take.

It is interesting to note that the system described here with second-order recurrence can be re-written as a bivariate dynamical system with first-order recurrence, as follows. In this case, the seed is (X_0, Y_0). The technique used here for the transformation works with all recurrences of any order: the reduction to first order is achieved by increasing the dimension of the system.

$$\begin{cases} X_{k+1} = Y_k, \\ Y_{k+1} = b_1 Y_k + b_0 X_k - \lfloor b_1 Y_k + b_0 X_k \rfloor. \end{cases}$$

2.1.4.2 Increasing point spread and randomness

As discussed in section 2.1.4.1, given a base (b_0, b_1), each triplet (X_k, X_{k+1}, X_{k+2}) lies in a family of parallel planes. Thus, these triplets are not randomly scattered in the entire 3-dimensional unit cube, regardless of the seed. Indeed the number of parallel planes can be quite small. This means that we are far from a uniform distribution on the unit cube. Using very large integers (in absolute value) for b_0 and b_1 significantly increases the number of parallel planes in the family, and thus the randomness.

The same problem exists with congruential random number generators (PRNG) [Wiki], though usually involving more than 3 dimensions and possibly more hyperplanes (thus more randomness) depending on the PRNG and its seed. Unlike PRNG sequences though, the sequence $\{X_k\}$ is typically non-periodic. Better random number generators are discussed in my book on synthetic data [8], in particular a very fast PRNG involving digits of millions of quadratic irrational numbers.

Now, let's get back to the sequence $\{X_k\}$ and assume that $b_0, b_1 > 0$. There is a simple technique to dramatically boost the number of planes, and thus the randomness: work with the sequence $\{Y_k\}$ instead of $\{X_k\}$, where $Y_k = X_{3k}$. For the sequence $\{Y_k\}$, the number of potential parallel planes where any (Y_k, Y_{k+1}, Y_{k+2}) can be found, is $M = b_0^3 + 3b_0 b_1 + b_1^3$. These planes satisfy the equation $b_0^3 x + b_1(b_1^2 + 3b_0)y - z = d$, where the

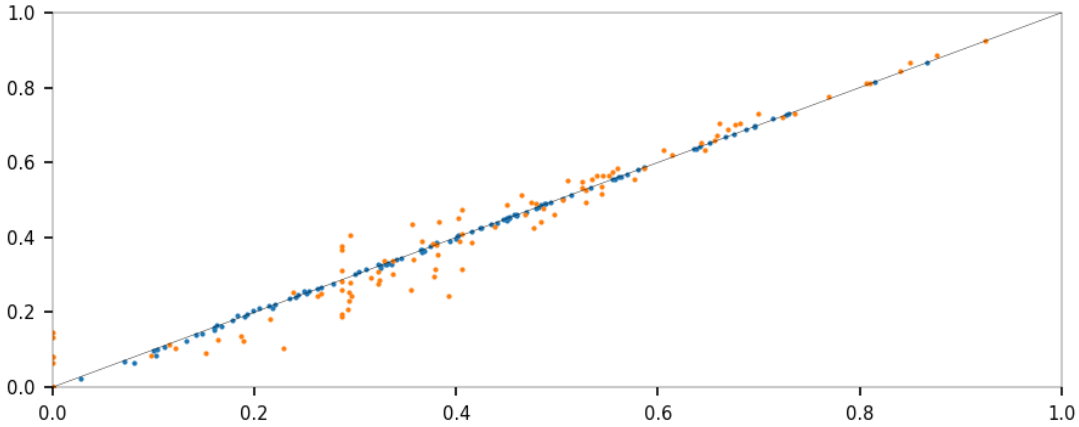


Figure 2.2: Base $(-5, 4)$ leads to better randomness [blue fit] than $(3, -2)$ [orange fit]

parameter d can take on M distinct values: $d = 0, 1, \ldots, M-1$. The first step to prove this result is to realize that due to the recurrence relation defining X_k and consequently Y_k, we always have

$$b_0^3 Y_k + b_1(b_1^2 + 3b_0)Y_{k+1} - Y_{k+2} = d,$$

where d is an integer between 0 and $M-1$. This fact assumes that $b_0, b_1 > 0$. Another solution is to use a deeper recurrence, say

$$X_{k+3} = b_2 X_{k+2} + b_1 X_{k+1} + b_0 X_k - \lfloor b_2 X_{k+2} + b_1 X_{k+1} + b_0 X_k \rfloor, \quad k = 0, 1, \ldots$$

where (X_0, X_1, X_2) is the seed, (b_0, b_1, b_2) is the base, and $Y_k = X_{4k}$. Now instead of parallel planes, we have parallel hyperplanes in 4 dimensions, and a much larger number of them. In short, a lot more randomness. And of course, this can be extended to higher dimensions, further boosting the randomness.

There is another way to look at the lack of randomness, and how to increase it. For the sequence $\{X_k\}$, not all bases are created equal. Some lead to more randomness than others. The following Kolmogorov-Smirnov statistic [Wiki] measures the distance to randomness:

$$\Delta = \max_{\alpha} \Delta(\alpha), \quad \text{with } \Delta(\alpha) = \left| \Big(\pi(\alpha)\Big)^{1/3} - \Big(\pi_0(\alpha)\Big)^{1/3} \right| \tag{2.4}$$

where

- In the 3-D version of the test, $\alpha = (\alpha_1, \alpha_2, \alpha_3) \in [0, 1]^3$, though the test is more revealing the higher the dimension; the maximum in (2.4) is over all $\alpha \in [0, 1]^3$. In d dimension, $1/3$ is replaced by $1/d$.
- $\pi(\alpha) = \pi(\alpha_1, \alpha_2, \alpha_3)$ is the empirical distribution [Wiki] of $P(X_k < \alpha_1, X_{k+1} < \alpha_2, X_{k+2} < \alpha_3)$, computed on many k's over a large number of good seeds (X_0, X_1), for a large number n of random triplets $(\alpha_1, \alpha_2, \alpha_3)$ in the unit cube.
- $\pi_0(\alpha) = \pi_0(\alpha_1, \alpha_2, \alpha_3) = \alpha_1\alpha_2\alpha_3$ is the exact distribution (CDF) if the X_k's were independently and uniformly distributed on $[0, 1]$. Here we known that they aren't.

The larger Δ, the less random the sequence $\{X_k\}$ is. The number of k's is denoted as `m` in the Python code in sectionpulkh. The fractional part of a real number x positive or negative, that is $x - \lfloor x \rfloor$, is denoted as `x%1` in the code. In other words, x modulo 1. It is between 0 and 1. Also, despite the appearance of using only one seed in the Python code, the fact that errors grow exponentially in the iterative computation of X_k means that the sequence $\{X_k\}$ is implicitly and randomly re-seeded every 45 iterations or so, for a total of $m = 10^5$ iterations per base (b_0, b_1).

Figure 2.2 produced by the code in section 2.1.4.3 shows $\Delta(\alpha)$ computed for $n = 100$ triplets α. Thus each set (blue, orange) consists of 100 dots. The further away the dots are from the main diagonal, the less random the sequence $\{X_k\}$ is on average. Orange corresponds to $(b_0, b_1) = (3, -2)$, blue to $(b_0, b_1) = (-5, 4)$. The Python code is also on my GitHub repository, here. Special cases such as $(b_0, b_1) = (0, 2)$ don't work not because of a glitch in the code, but because of the way hardware architecture (related to arithmetic operations) is designed. In this case, replace 2 by 2 ± 2^{-30}, and it will fix the issue even though the base no longer consists of integers.

2.1.4.3 Python implementation of distance to randomness

This code described in section 2.1.4.2 is also on GitHub, here. It is used to produce Figure 2.2. The purpose is to measure the quality of a base (b_0, b_1), in terms of its ability to generate randomness in the sequence $\{X_k\}$, regardless of the (good) seed (X_0, X_1).

```
# chaos_2D_base.py | compute Kolmogorov-Smirnov Delta distance in test of independence

import numpy as np
import matplotlib.pyplot as plt

def simulation(b0,b1,n,m):

   Delta  = 0.0; # Kolmogorov-Smirnov distance
   arr_pi = [0] * n
   arr_pi_0 = [0] * n

   for iter in range(0, n):
      alpha1 = np.random.uniform(0.0, 1.0)
      alpha2 = np.random.uniform(0.0, 1.0)
      alpha3 = np.random.uniform(0.0, 1.0)
      count = 0
      for k in range (2, m):
         x[k] = (b1 * x[k-1] + b0* x[k-2]) % 1
         if x[k-2]<alpha1 and x[k-1]<alpha2 and x[k]<alpha3:
            count += 1
      pi  = (count/(m-1))**(1/3)
      pi_0 = (alpha1 * alpha2 * alpha3)**(1/3)
      arr_pi.append(pi)
      arr_pi_0.append(pi_0)
      diff = abs(pi - pi_0)
      if diff > Delta:
         Delta = diff
      OUT.write("%3d\t%3d\t%.4f\t%.4f\t%.4f\t%6.5f\t%6.5f\n" \
          % (b0,b1,alpha1,alpha2,alpha3,pi,pi_0))

   return(Delta, arr_pi, arr_pi_0)


n = 100
m = 100000
seed = 543
np.random.seed(seed)
x = [0] * m # initialize array of size m
x[0] = 0.5   # seed: X_0
x[1] = np.log(2) # seed: X_1

OUT=open("base2d.txt","w")
OUT.write("b0\tb1\talpha0\talpha1\talpha2\tpi\tpi_0\n")
axes = plt.axes()
[axx.set_linewidth(0.2) for axx in axes.spines.values()]
axes.tick_params(axis='both', which='major', labelsize=7)
axes.tick_params(axis='both', which='minor', labelsize=7)
axes.set_xlim(0.0, 1.0)
axes.set_ylim(0.0, 1.0)
plt.plot([0,1], [0,1], 'k-', linewidth=0.2)

b0 = -5; b1 =4
(Delta, arr_pi, arr_pi_0) = simulation(b0, b1, n, m)
print("base = (%2d, %3d) | Kolmogorov-Smirnov distance: %6.4f" % (b0, b1, Delta))
plt.scatter(arr_pi, arr_pi_0,s=1.0)

b0 = 3; b1 =-2
(Delta, arr_pi, arr_pi_0) = simulation(b0, b1, n, m)
print("base = (%2d, %3d) | Kolmogorov-Smirnov distance: %6.4f" % (b0, b1, Delta))
plt.scatter(arr_pi, arr_pi_0,s=1.0)
```

```
plt.show()
OUT.close()
```

2.2 Iterative method to find the invariant distribution

Now I discuss an iterative algorithm to solve the stochastic integral equation (2.3) to find the invariant distribution F. Usually, there is no direct, exact solution. Even when there is one as in the preceding examples, the ability to retrieve it via an iterative algorithm tells you how fast your algorithm converges, and what the error looks like. Remember that here, the unknown is a function F or its derivative (the density) f. Nevertheless, the fixed-point algorithm [Wiki] still works to get a very accurate approximation of this function. It is implemented as follows.

We start with a rough approximation f_0 of the invariant density. Then we build successive approximations using the fixed-point iteration

$$f_{n+1}(x) = \sum_{k=1}^{\infty} |q'(x+k)| f_n(q(x+k)). \tag{2.5}$$

The algorithm is a direct application of Formula (2.3). As an illustration, I use it to find the invariant distribution (a good approximation) for the dynamical system defined by $g(x) = p(x) - \lfloor p(x) \rfloor$, where $q(x) = p^{-1}(x)$ and $p(x) = \lambda/x^{\alpha}$. Here $\lambda, \alpha > 0$ and $\alpha \leq 1$. The exact solution is known at least when $\alpha = \lambda = 1$. In this case, we are dealing with the Gauss map studied earlier. In all the cases, the domain of the invariant density is $[0, 1]$. I start with the uniform density $f_0(x) = 1$ for $x \in [0, 1]$. The implementation is in the Python code in section 2.2.2.

2.2.1 Results and discussion about convergence

Figure 2.3 shows the successive approximations f_1, f_2 and so on, converging very fast towards the exact solution: the invariant density $f(x) = \kappa/(1+x)$ with $\kappa = 1/\log 2$ (in red). This is true even when starting with the uniform density $f_0(x) = 1$. The function f_1 is in blue, f_2 in orange, f_3 in green, and f_4 is already indistinguishable from the exact solution. I used the first 500 terms in the right-hand side of formula (2.5), with 5000 evenly spaced locations (interpolating nodes) on the X-axis for the approximations. You can do very well with much fewer terms and nodes. The Python implementation is in section 2.2.2.

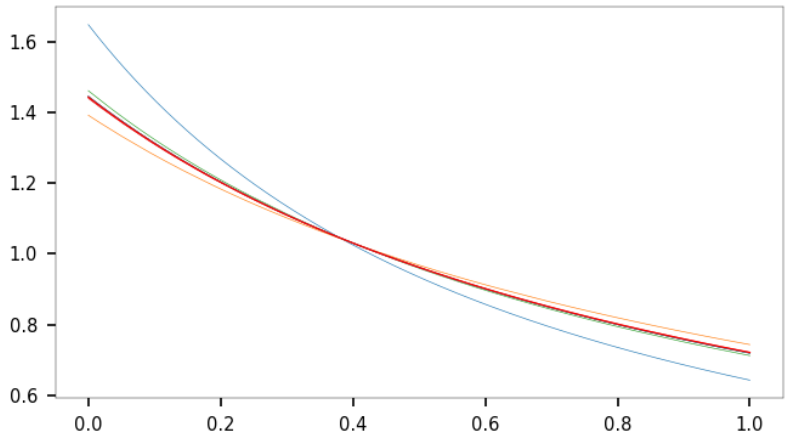


Figure 2.3: Convergence of iterated functions towards the invariant density in red

Later in this book, I discuss other dynamical systems where the exact solution to the functional equation is known in closed form. It leads to interesting and sometimes awkward, non-standard probability distributions. We already established the invariant distribution attached to the Gauss map, as well as the related Gauss-Kuzmin distribution attached to the digits or coefficients of continued fractions. We will see that we also have a special, simple invariant distribution attached to the "digits" of nested radicals [Wiki]. We will also discover that the well-known invariant measure of the logistic map defined by $g(x) = 4x(1-x)$ is a beta distribution. Even more surprising, we will find (in closed form) the invariant measure of the square-root logistic map defined by $g(x) = \sqrt{4x(1-x)}$.

2.2.2 Python code to numerically solve the functional equation

The code is also available on GitHub, here. I use the first `N` terms in formula (2.5), with `M` evenly spaced locations (interpolating nodes) on the X-axis for the approximations. The function f_{n+1} is stored in the array `f_new` with `M+1` entries, and computed based on values of f_n stored in `f_old`. These arrays are updated at each iteration. The exact solution is stored in `f_exact`.

```
# chaos_solveFunctional.py | fixed-point iteration to compute invariant density f
# f_exact corresponds to alpha = llambda = 1: f(x) = (1/log(2)) * 1/(1+x)

import numpy as np
import matplotlib.pyplot as plt

M = 5000  # granularity
N = 500   # number of terms in the sum for functional equation
llambda = 1 # must be <= 1
alpha = 1
beta  = 1/alpha

OUT=open("solve.txt","w")

f_old = []
f_new = []
f_exact = []
x = []
for k in range(M+1):
   f_old.append(1.0) # initialize f_0 to uniform[0,1]
   f_new.append(0.0)
   f_exact.append( (1/np.log(2)) / (1+k/M) )
   x.append(k/M)  # locations on X-axis
axes = plt.axes()
[axx.set_linewidth(0.2) for axx in axes.spines.values()]
axes.tick_params(axis='both', which='major', labelsize=7)
axes.tick_params(axis='both', which='minor', labelsize=7)

for iter in range(0,10):

   print("Fixed-point iteration:",iter)
   sum=0

   for k in range(M+1):
      # loop over equally spaced arguments of f
      f_new[k]=0
      for n in range(1,N+1):
         # loop over the terms in functional equation
         y=(llambda/(x[k] + n))**beta # x[k] = k/M
         y=int(0.5 + M*y)       # must have: 0 <= y <= M
         if y <= M+1:
            f_new[k] += f_old[y]*beta*(llambda**beta)/((x[k]+n)**(1+beta))
         else:
            print("Out of range error (ignored):",iter,k,y)
      sum += f_new[k]

   for k in range(0,M+1):
      f_new[k] = M*f_new[k]/sum # must integrate to 1 (it's a density)
      f_old[k] = f_new[k]

   for k in range(M+1):
      OUT.write("%6d\t%6d\t%6.4f\t%6.4f\n" % (iter,k,x[k],f_new[k]))
   plt.plot(x,f_new,linewidth=0.3)

OUT.close()

plt.plot(x,f_new,linewidth=0.3)
plt.plot(x,f_exact,color='red',linewidth=0.6)
plt.show()
```

2.2.3 Curious, very accurate approximation of π

The dynamical system discussed here, namely with $p(x) = \lambda/x^{\alpha}$, leads to a curious approximation of π, in just one iteration:

$$\pi - 3 \approx (6189766)^{-1/8}$$

correct to 9 decimal places: the error is about 5.19×10^{-11}. It is obtained with $\lambda = 1$ and $\alpha = 1/8$. The digits of such systems are defined in section 2.1.2.1, and 6189766 turns out to be the first digit of $\pi - 3$ in that system.

By contrast, the approximation $\pi \approx 355/113$ is obtained using 3 digits of $\pi - 3$ in the continued fraction system (the Gauss map, with $p(x) = 1/x$). It was already considered a pretty spectacular approximation, correct to 6 decimal places, with the error around 2.67×10^{-7}. Its history dates back to the 5th century when it was named the Milü number (see here). Note that the Gauss map is a particular case of the more general dynamical system in question, with $\lambda = \alpha = 1$.

2.3 Measuring the amount of chaos

Before discussing chaos, I want to emphasize the fact that many dynamical systems have a non-chaotic version depending on the parameter. For instance, for the logistic map $g(x) = rx(1-x)$ discussed earlier, there is no chaos in the sequence $\{X_k\}$ if $r < 3$. Chaos starts to kick-in at $r \geq 3$. The sequence is fully chaotic when $r = 4$. In this book, I only consider fully chaotic sequences.

In the logistic map, when $r \geq 3$, the sequence exhibits a number of bifurcations [Wiki]: 2 when $r = r_1 = 3$, then 4 starting at $r = r_2 \approx 3.449$, then 8 starting at $r = r_3 \approx 3.544$, then 16 starting at $r = r_4 \approx 3.564$ and so on, up to an infinite number at $r = 4$. Successive ratios $(r_{n-1} - r_{n-2})/(r_n - r_{n-1})$ tend to some constant when $n \to \infty$. That constant, also found in many other dynamical systems, is the universal constant of chaos. It is known as Feigenbaum's constant [Wiki], and its value is approximately 4.669.

2.3.1 Hurst exponent and related metrics

In the context of Brownian motions, random walks and related chaotic processes discussed in chapter 1, chaos is often measured using some kind of moving average. Here I illustrate some of these metrics applied to different types of stochastic processes ranging from very smooth (barely chaotic) to highly unpredictable (very chaotic). These processes are built using the following method:

Step 1: Use an autocorrelated stationary process $\{X_k\}$, with $k = 0, 1$ and so on.

Step 2: The standardized $\{X_k\}$ is $\{Y_k\}$ with mean and variance equal to 0 and 1 respectively.

Step 3: $\{Z_k\}$ is the final process with $Z_k = (Y_0 + \cdots + Y_k)/\sqrt{k+1}$.

In practice, we might be more interested in time-continuous processes than in the discrete version. So a time series featuring $\{Z_k\}$ for $k = 0, 1$ and so on up to $k = N$ may be re-scaled (say) with $t = k/750$ so that it looks time-continuous with the time now ranging from $t = 0.00$ to $t = N/750$ (with time increments of $1/750$). This technique is described in section 1.1.

Now I provide 4 examples of such processes, each featuring a class of patterns in terms of chaotic behavior. Details about the construction is found in my spreadsheet `Chaos_BrownianFractional.xlsx` available on GitHub, here. The spreadsheet illustrates the construction of $\{X_k\}, \{Y_k\}, \{Z_k\}$ for series A, as well as the computation of smoothness metrics. I used it to produce Figure 2.4. In the spreadsheet, $N = 22{,}212$.

Series A: $X_{k+1} = bX_k - \lfloor bX_k \rfloor$ with $X_0 = \log 2$ and $b = (1 + \sqrt{5})/2$.

Series B: X_k is either 0 or 1 with probability $\frac{1}{2}$. The X_k's are independent.

Series C: $X_{k+1} = b + X_k - \lfloor b + X_k \rfloor$, with $X_0 = \log 2$, $b = (1 + \sqrt{5})/2$.

Series D: for $\{X_k\}$, use $\{Z_k\}$ from series C.

Interestingly, in series A, $\lfloor bX_k \rfloor$ is the k-th digit of X_0 in the irrational Golden ratio base [Wiki], discussed later in this book. The underlying weakly autocorrelated dynamical system $\{X_k\}$ is in the same family as the dyatic map corresponding to $b = 2$, and related to the binary numeration system. Of course in series B, $\{X_k\}$ is not autocorrelated. Series C and D have long-range autocorrelations.

The operation that consists of transforming $\{X_k\}$ into $\{Z_k\}$ is known as integration. The resulting sequence $\{Z_k\}$ is smoother (less chaotic) than $\{Y_k\}$. The inverse operation is known as differentiation: it increases chaos. Note that $\{Z_k\}$ is a fully variance-normalized Brownian motion if you start with a white noise for $\{X_k\}$. So it is not really a Brownian motion, as oscillations stay within a standard range instead of growing in amplitude over

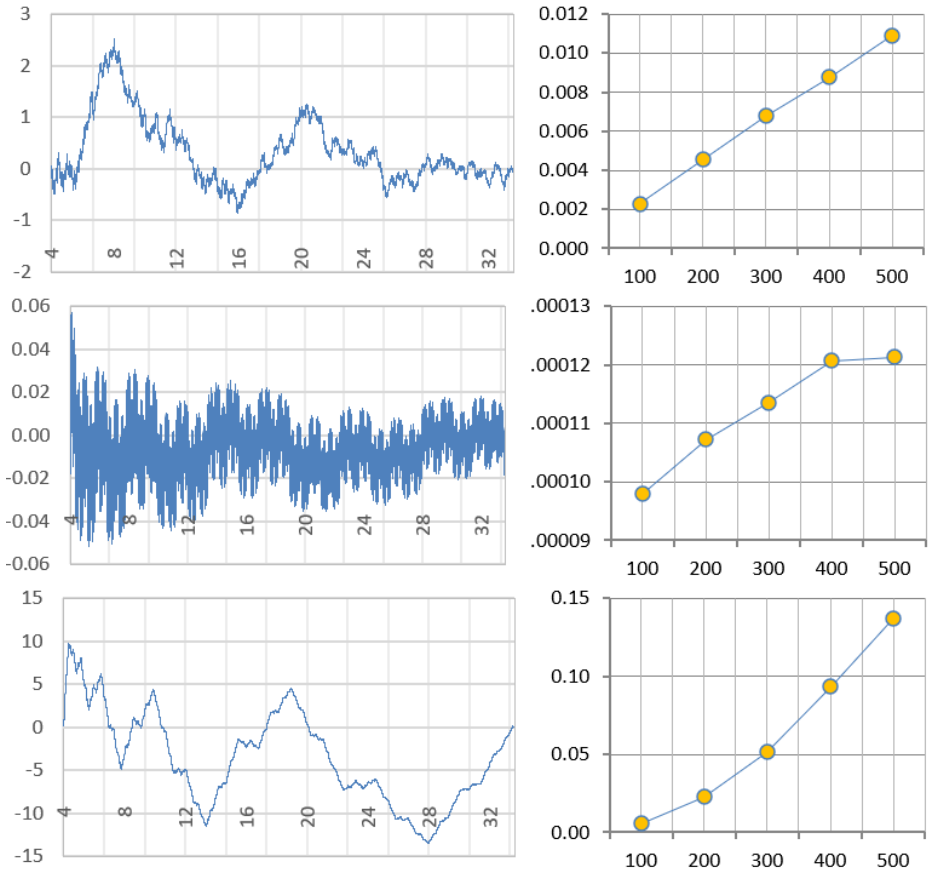


Figure 2.4: Series A, C, D (left) with corresponding chaos measurements (right)

time. It is related to fractional Brownian motions [Wiki] when the original sequence $\{X_k\}$ exhibits long-range autocorrelations.

Finally, Series A an B result in the same category of processes. Despite the fact that $\{X_k\}$ is autocorrelated in series A and not in series B, that autocorrelations are weak enough as to have no impact on $\{Z_k\}$ when turned into a time-continuous process. So I did not include series B in my illustrations. To the contrary, autocorrelations in series C and D are strong and long-range, and have a noticeable impact on the time-continuous version of $\{Z_k\}$.

2.3.1.1 Detrending moving averages and spreadsheet illustration

The traditional metric to measure the smoothness of $\{Z_k\}$ is the detrending moving average, and it is abbreviated as DMA. It is the mean squared error between your observations and its various moving averages of order $m = 1, 2, 3$, and so on. The exact definition can be found in "Statistical test for fractional Brownian motion based on detrending moving average algorithm" [13] and in many other articles. Other criteria are also used, such as FA and DFA. A comparison of these metrics can be found in "Comparing the performance of FA, DFA and DMA using different synthetic long-range correlated time series" [12]. In my spreadsheet, I use DMA, along with autocorrelations of various lags. The better notation DMA(m) emphasizes the fact that it depends on m. Now we have this well-known result:

$$\text{DMA}(m) \sim C(h) \cdot m^{2H},$$

where $C(\cdot)$ is a bounded function.

This is an asymptotic result, meaning that it becomes more accurate as m grows to infinity. The constant H is known as the Hurst exponent [Wiki]. It takes on values between 0 and 1, with $H = \frac{1}{2}$ corresponding to the Brownian motion. Higher values correspond to smoother time series, and lower ones to more chaotic data. In the spreadsheet, since I represent time-continuous processes where 750 increments of discrete time correspond to one increment or unit of continuous time (say a second), I use large values $m = 100, 200, \ldots, 500$.

Also, in order to work with stabilized variances and avoid the volatility present at the beginning of $\{Z_k\}$ (due to the fact that we start at 0), I study the time series and perform all the computations assuming we start at time $t = 4.00$, that is, after skipping the first 4×750 observations in discrete time. This is noticeable in Figure 2.4, where the X-axis on the left plots represents the time, and starts at $t = 4.00$.

The main takeaway from my spreadsheet and analysis, summarized in Figure 2.4, is the fact that the curvature of the function DMA(m) is a good indicator of the amount of chaos present in $\{Z_k\}$. If you look at the right plots in Figure 2.4, the curve with the yellow dots represent DMA(m), and the X-axis represents m.

Exercise 8 *A very smooth time series* – Create series E as follows: use $\{Z_k\}$ obtained in series D as your starting point $\{X_k\}$ to produce a new $\{Z_k\}$, smoother than that in series D.

Solution
Using my Excel spreadsheet, this can be done with one easy step: copy column $\{Z_k\}$ (the values only) onto column $\{X_k\}$ and all the summary statistics, the new $\{Z_k\}$, and the plot will automatically be updated. See solution in Figure 2.5. Note that as the smoothness increases (or in other words, chaos and entropy is reduced), the curvature of the curve with yellow dots on the right-hand side in Figure 2.4 and 2.5 is also increasing. Other examples are provided in section 1.3: see Figure 1.2.

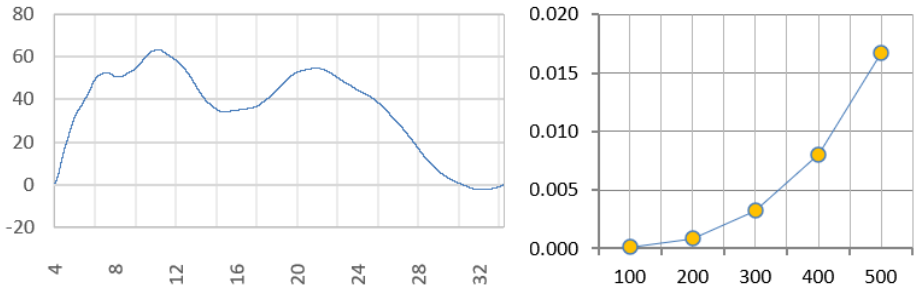


Figure 2.5: Series E (left) with corresponding chaos measurements (right)

2.3.2 Lyapunov exponent and related metrics

The Lyapunov exponent [Wiki] is used to detect a different feature of chaos: sensitivity to initial conditions, or the fact that two very close seeds X_0 and X_0' lead to totally different trajectories $\{X_k\}$ and $\{X_k'\}$ over time, typically after very few iterations. For instance, the ten-fold map $g(x) = 10x - \lfloor 10x \rfloor$ leads to the digits of X_0 in base 10: the k-th digit is $\lfloor 10X_k \rfloor$. If instead of (say) $X_0 = \log 2$, you start with $X_0' = X_0 + \pi \times 10^{-300}$, after $k = 300$ iterations, the digits become totally unrelated and the two sequences completely separate.

Another metric is the approximate entropy [Wiki]. It is used to quantify regularity and predictability in time series fluctuations. Applications include medical data, finance, physiology, human factors engineering, and climate sciences. It should not be confused with entropy [Wiki], which measures the amount of information attached to a specific probability distribution – with the uniform distribution on $[0, 1]$ achieving maximum entropy among all continuous distributions on $[0, 1]$, and the normal distribution achieving maximum entropy among all continuous distributions defined on the real line, given a specific variance.

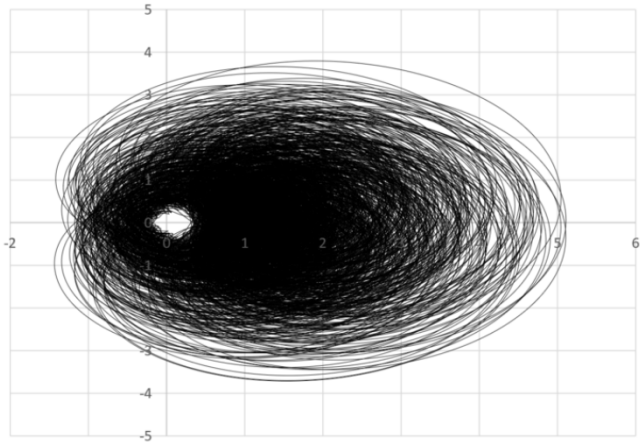


Figure 2.6: Non-periodic orbit with fractal dimension and basin of repulsion

You can also use the autocorrelation function of $\{X_k\}$ [Wiki] as an indicator of chaos. Autocorrelations that are very weak (close to zero) or decaying extremely fast are associated with increased chaos. To the contrary, long-range, strong autocorrelations are associated with higher predictability and less chaos, as illustrated in Figure 2.5 and the bottom part of Figure 2.4. For instance, in the ten-fold map just discussed, the lag-m

autocorrelation is 10^{-m} for $m = 1, 2$ and so on. It is decaying exponentially fast as m increases, making this system quite chaotic. Methods based on the autocorrelation structure fit in the field of spectral theory [10].

Finally, the fractal dimension [Wiki] is also used, especially to characterize chaos in the orbit of 2D dynamical systems. In a 2D system, the time-continuous path or orbit (X_t, Y_t) may densely fill an entire domain. This domain is sometimes called the basin of attraction, and zones that are never visited are referred to as repulsion basins. The cover of this book feature such basins. Figure 2.6 illustrates the concept, using some orbit of the Riemann zeta function $\zeta(\sigma, t)$. Depending on the parameter σ, some hole around the origin is never visited (when ζ has no zero), and the area being visited may extend to the entire complex plane. More about this in my book on stochastic processes [7]. The fractal dimension of the orbit, even though it is a one-dimensional curve, is a real number between 1 and 2. It generalizes to higher dimensions.

2.4 The pillow map: a fascinating bivariate system

This section is a brief introduction to 2D discrete chaotic dynamical systems. I focus exclusively on the sine map, which I renamed the pillow map for reasons that will soon become obvious. In one dimension, it is defined by $g(x) = -\rho x + \lambda \sin x$. In two dimensions, by $g(x, y) = (-\rho x + \lambda \sin y, -\rho y + \lambda \sin x)$, or equivalently:

$$\begin{cases} X_{k+1} = -\rho X_k + \lambda \sin Y_k, \\ Y_{k+1} = -\rho Y_k + \lambda \sin X_k. \end{cases} \tag{2.6}$$

It is governed by two parameters ρ, λ. Before exploring the 2D version, I discuss results related to the one-dimensional case.

2.4.1 The one-dimensional version: Arnold's tongues

Here, let $\rho = 1$. The one-dimensional case is related to Arnold's tongues [Wiki] and the circle map. You would expect the corresponding sequence $\{X_k\}$ to depend on X_0 and to exhibit a chaotic, Brownian-type behavior, and indeed it does that pretty much all the time. However, if $\lambda = 8$ (also true if λ is very close to 8), we have $X_k \sim \pm 2k\pi$ as $k \to \infty$. The sign depends on the initial value X_0, and the symbol $\sim$ means asymptotically equal. Assuming $X_0 = 2$ and $\lambda = 8$, as $k \to \infty$ we have

$$X_{2k} - X_{2k-1} \to \alpha \approx 7.939712,$$
$$X_{2k-1} - X_{2k-2} \to \beta \approx -1.65653,$$

with $\alpha + \beta = 2\pi$ and α solution of $2\pi = \alpha + \alpha \cos \alpha - \sqrt{\lambda^2 - \alpha^2} \sin \alpha$.

If X_0 is large (say $X_0 = 67$) and $1 < \lambda < 3$, then X_k converges very rapidly so the sequence looks flat (non-chaotic). If $X_0 = 67, \lambda = 7.99$, the sequence is chaotic. If $X_0 = 67, \lambda = 8$, we have the behavior described earlier. And with $\lambda > 8.02$ we are back to chaotic behavior. Now if $X_0 = 67, \lambda = 4$, then X_k stays in a flat, narrow band, constantly oscillating.

2.4.2 Two-dimensional case

For the 2D version, references include "Chaotic Synchronization and Antisynchronization in Coupled Sine Maps" [14] and "Basins and Critical Curves Generated by A Family of Two-Dimensional Sine Maps" [9]. Here I discuss the non-chaotic case when $|\rho| < 1$ and $|\lambda|$ is not too large, so that the iterates in (2.6) converge to a limit (x, y) depending on the seed, as $k \to \infty$. If the system (2.6) convergences, the limit vector must satisfy

$$\begin{cases} x = -\rho x + \lambda \sin y, \\ y = -\rho y + \lambda \sin x. \end{cases} \tag{2.7}$$

Solutions to system (2.7) must satisfy

$$x = \eta \sin(\eta \sin x), \quad y = \eta \sin(\eta \sin y), \quad \text{with } \eta = \frac{\lambda}{1 + \rho}. \tag{2.8}$$

Solutions also satisfy $y \sin y = x \sin x$. There are 7 solutions to (2.8) for x, and 7 for y, thus a combined total of $7 \times 7 = 49$ solutions for (x, y). Among these 49 solutions, four of them are the limit vector of the fixed-point iteration (2.6), for some seed satisfying $-4 \leq X_0, Y_0 \leq 4$. Their approximate values are

$$(x, y) \approx \pm(1.3588, 2.6069) \text{ or } \pm(2.6069, 1.3588).$$

These four solutions, also called fixed points or attractors [Wiki], lead to four bifurcations (determined by the seed) in the orbit of the system. In Figure 2.7, all seeds (X_0, Y_0) in $[-4, 4] \times [-4, 4]$ leading to a same fixed-point are painted in the same color. More details are provided in section 2.4.3.

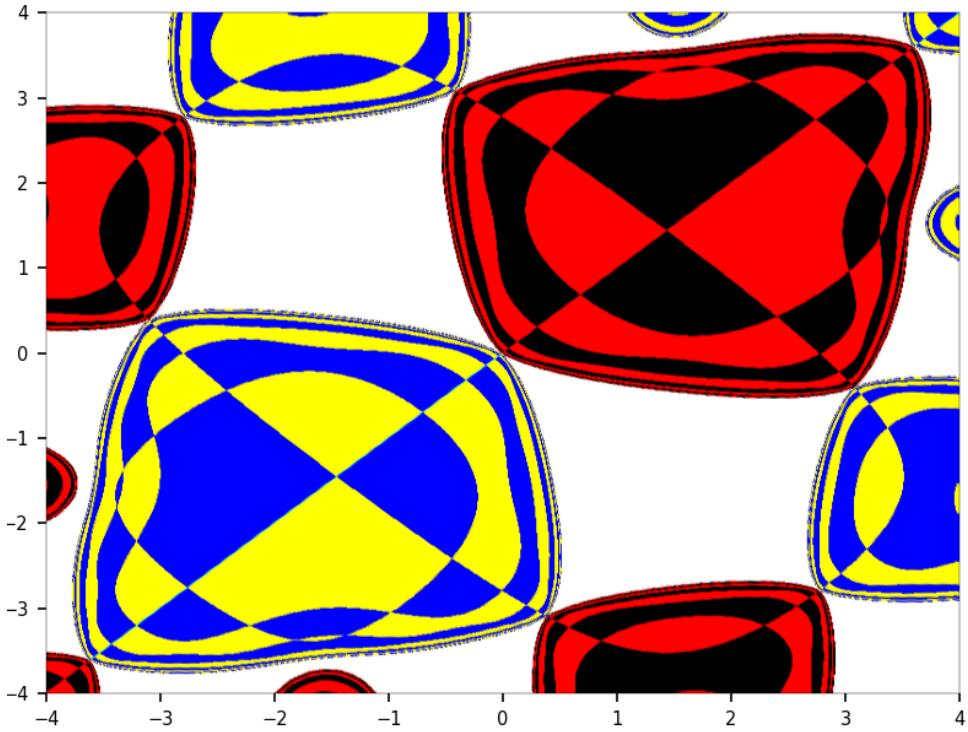


Figure 2.7: Basins of attraction of the 2D sine map

2.4.3 Python code to generate the basins of attraction

The Python code is also on GitHub, here. The implementation is based on $\lambda = 2$ and $\rho = -0.25$. Different parameters may result in more basins: in that case, you need to add enough colors to the `color` table, so that a color is assigned to each basin. When the sequence does not converge, the corresponding seed (X_0, Y_0) is attached to a virtual basin with `basin_ID=-1`, and painted in white: the color entry `(1,1,1)`.

There are 4 large attraction basins (black, red, yellow, blue) in addition to the white area, each corresponding to a root of (2.7). Each seed vector belongs to one of these basins, or to the white area. The components x, y of the root attached to the basin `basin_ID` are stored in `basin_x[basin_ID]` and `basin_y[basin_ID]`. A few seeds do not belong to any of these basins, nor to the white area: they constitute tiny separate "artificial basins", so small that they are not visible in Figure 2.7. Typically, they are located at the border between two basins, and caused by convergence issues. All basins (real or artificial) are listed, along with their size and color, in the output produced by the Python code. The 4 real basins are easy to identify due to their large size; each one contains 25% of the seeds not in the white area.

```
# basins.py | find and plot basins of attraction of 2D sine map

import numpy as np
import matplotlib.pyplot as plt

llambda = 2
theta  = 1
rho    = -0.25
eps    = 0.00001

n_basins = 0
basin_count = {}
basin_color = {}
basin_x  = {}
basin_y  = {}
list_x = []
list_y = []
list_color = []

color = [(0, 0, 0),
       (1, 0, 0),
```

```
        (0, 0, 0),
        (1, 0, 0),
        (1, 1, 1), # (0.9, 0.8, 1)
        (1, 1, 0),
        (0, 0, 1),
        (0, 1, 1)]

OUT=open("basins.txt","w")

for X_0 in np.arange(-4, 4, 0.01):
   print("X_0 = %5.3f" % (X_0))
   for Y_0 in np.arange(-4, 4, 0.01):
      x = X_0
      y = Y_0
      k = 0
      delta = 999999.9
      for k in range(100):
         old_x = x
         old_y = y
         x = -rho*old_x + llambda*np.sin(theta*old_y)
         y = -rho*old_y + llambda*np.sin(theta*old_x)
         delta = max(abs(x-old_x), abs(y-old_y))
      if delta>0.2:
         basin_ID = -1 # non-convergence zone (oscillating)
      else:
         basin_ID = int(100 + 10*x + y + 0.5)
      if basin_ID in basin_count:
         basin_count[basin_ID] += 1
      else:
         basin_count[basin_ID] = 1
         n_basins = n_basins + 1
         basin_color[basin_ID] = color[n_basins]
         basin_x[basin_ID] = x
         basin_y[basin_ID] = x
      list_x.append(X_0)
      list_y.append(Y_0)
      list_color.append(basin_color[basin_ID])

OUT.close()

for basin_ID,count in basin_count.items():
   col = str(basin_color[basin_ID])
   x  = basin_x[basin_ID]
   y  = basin_x[basin_ID]
   print("basinID: %4d count: %6d color: %8s x: %7.4f y: %7.4f" \
       % (basin_ID,count,col,x,y))

axes = plt.axes()
[axx.set_linewidth(0.2) for axx in axes.spines.values()]
# axes.set_facecolor("white") # background color
axes.margins(x=0)
axes.margins(y=0)
axes.tick_params(axis='both', which='major', labelsize=7)
axes.tick_params(axis='both', which='minor', labelsize=7)
plt.scatter(list_x, list_y, c=list_color, s=0.1)
plt.xlim(-4, 4)
plt.ylim(-4, 4)
plt.show()
```

Bibliography

[1] Rabi Bhattacharya and Edward Waymire. *Random Walk, Brownian Motion, and Martingales*. Springer, 2021. 10

[2] D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Springer, second edition, 2002. Volume 1 – Elementary Theory and Methods. 16

[3] D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Springer, second edition, 2014. Volume 2 – General Theory and Structure. 16

[4] P. A. Van Der Geest. The binomial distribution with dependent Bernoulli trials. *Journal of Statistical Computation and Simulation*, pages 141–154, 2004. [Link]. 11

[5] B.V. Gnedenko and A. N. Kolmogorov. *Limit Distributions for Sums of Independent Random Variables*. Addison-Wesley, 1954. 17

[6] Manuel González-Navarrete and Rodrigo Lambert. Non-markovian random walks with memory lapses. *Preprint*, pages 1–14, 2018. arXiv [Link]. 10

[7] Vincent Granville. *Stochastic Processes and Simulations: A Machine Learning Perspective*. MLTechniques.com, 2022. [Link]. 17, 32

[8] Vincent Granville. *Synthetic Data and Generative AI*. MLTechniques.com, 2022. [Link]. 4, 6, 11, 12, 24

[9] Nasr-Eddine Hamri and Yamina Soula. Basins and critical curves generated by a family of two-dimensional sine maps. *Electronic J. of Theoretical Physics*, 24:139–150, 2010. [Link]. 32

[10] D. Lenz. Spectral theory of dynamical systems as diffraction theory of sampling functions. *Monatshefte für Mathematik*, 192:625–649, 2020. [Link]. 32

[11] Peter Mörters and Yuval Peres. *Brownian Motion*. Cambridge University Press, 2010. Cambridge Series in Statistical and Probabilistic Mathematics, Volume 30 [Link]. 10, 16

[12] Ying-Hui Shao et al. Comparing the performance of FA, DFA and DMA using different synthetic long-range correlated time series. *Preprint*, pages 1–9, 2018. arXiv:1208.4158 [Link]. 30

[13] Grzegorz Sikora. Statistical test for fractional Brownian motion based on detrending moving average algorithm. *Chaos, Solitons & Fractals*, 116:54–62, 2018. [Link]. 30

[14] Maistrenko V et al. Chaotic synchronization and antisynchronization in coupled sine maps. *International Journal of Bifurcation and Chaos*, 15:2161–2177, 2005. [Link]. 32

[15] Lan Wua, Yongcheng Qi, and Jingping Yang. Asymptotics for dependent Bernoulli random variables. *Statistics and Probability Letters*, pages 455–463, 2012. [Link]. 10

Index