Applied Stochastic Processes, Chaos Modeling, and Probabilistic Properties of Numeration Systems

By Vincent Granville, Ph.D.

www.DataScienceCentral.com

June 2018.

This book is intended for professionals in data science, computer science, operations research, statistics, machine learning, big data, and mathematics. In 100 pages, it covers many new topics, offering a fresh perspective on the subject. It is accessible to practitioners with a two-year college-level exposure to statistics and probability. The compact and tutorial style, featuring many applications (Blockchain, quantum algorithms, HPC, random number generation, cryptography, Fintech, web crawling, statistical testing) with numerous illustrations, is aimed at practitioners, researchers and executives in various quantitative fields.

New ideas, advanced topics, and state-of-the-art research are discussed in simple English, without using jargon or arcane theory. It unifies topics that are usually part of different fields (data science, operations research, dynamical systems, computer science, number theory, probability) broadening the knowledge and interest of the reader in ways that are not found in any other book. This short book contains a large amount of condensed material that would typically be covered in 500 pages in traditional publications. Thanks to cross-references and redundancy, the chapters can be read independently, in random order.

This book is available for Data Science Central members exclusively. The text in blue consists of clickable links to provide the reader with additional references. Source code and Excel spreadsheets summarizing computations, are also accessible as hyperlinks for easy copy-and-paste or replication purposes. The most recent version of this book is available from this link, accessible to DSC members only.

About the author

Vincent Granville is a start-up entrepreneur, patent owner, author, investor, pioneering data scientist with 30 years of corporate experience in companies small and large (eBay, Microsoft, NBC, Wells Fargo, Visa, CNET) and a former VC-funded executive, with a strong academic and research background including Cambridge University.

Content

The book covers the following topics:

- From Random Walks to Brownian Motion
- Stationarity, Ergodicity, Fractal Behavior
- Memory-less or Markov Property
- Non-Brownian Process

2. Integration, Differentiation, Moving Averages

We introduce more advanced concepts about stochastic processes. Yet we make these concepts easy to understand even to the non-expert. This is a follow-up to Chapter 1.

- Integrated, Moving Average and Differential Process
- Proper Re-scaling and Variance Computation
- Application to Number Theory Problem

3. Self-Correcting Random Walks

We investigate here a breed of stochastic processes that are different from the Brownian motion, yet are better models in many contexts, including Fintech.

- Controlled or Constrained Random Walks
- Link to Mixture Distributions and Clustering
- First Glimpse of Stochastic Integral Equations
- Link to Wiener Processes, Application to Fintech
- Potential Areas for Research
- Non-stochastic Case

4. Stochastic Processes and Tests of Randomness

In this transition chapter, we introduce a different type of stochastic process, with number theory and cryptography applications, analyzing statistical properties of numeration systems along the way -- a recurrent theme in the next chapters, offering many research opportunities and applications. While we are dealing with deterministic sequences here, they behave very much like stochastic processes, and are treated as such. Statistical testing is central to this chapter, introducing tests that will be also used in the last chapters.

- Gap Distribution in Pseudo-Random Digits
- Statistical Testing and Geometric Distribution
- Algorithm to Compute Gaps
- Another Application to Number Theory Problem
- Counter-Example: Failing the Gap Test

5. Hierarchical Processes

We start discussing random number generation, and numerical and computational issues in simulations, applied to an original type of stochastic process. This will become a recurring theme in the next chapters, as it applies to many other processes.

- Graph Theory and Network Processes
- The Six Degrees of Separation Problem
- Programming Languages Failing to Produce Randomness in Simulations
- How to Identify and Fix the Previous Issue
- Application to Web Crawling

6. Introduction to Chaotic Systems

While typically studied in the context of dynamical systems, the logistic map can be viewed as a stochastic process, with an equilibrium distribution and probabilistic properties, just like numeration systems (next chapters) and processes introduced in the first four chapters.

- Logistic Map and Fractals
- Simulation: Flaws in Popular Random Number Generators
- Quantum Algorithms

7. Chaos, Logistic Map and Related Processes

We study processes related to the logistic map, including a special logistic map discussed here for the first time, with a simple equilibrium distribution. This chapter offers a transition between chapter 6, and the next chapters on numeration system (the logistic map being one of them.)

- General Framework
- Equilibrium Distribution and Stochastic Integral Equation
- Examples of Chaotic Sequences
- Discrete, Continuous Sequences and Generalizations
- Special Logistic Map
- Auto-regressive Time Series
- Literature
- Source Code with Big Number Library
- Solving the Stochastic Integral Equation: Example

8. Numerical and Computational Issues

These issues have been mentioned in chapter 7, and also appear in chapters 9, 10 and 11. Here we take a deeper dive and offer solutions, using high precision computing with BigNumber libraries.

- Precision Issues when Simulating, Modeling, and Analyzing Chaotic Processes
- When Precision Matters, and when it does not
- High Precision Computing (HPC)
- Benchmarking HPC Solutions
- How to Assess the Accuracy of your Simulation Tool

9. Digits of Pi, Randomness, and Stochastic Processes

Deep mathematical and data science research (including a result about the randomness of π, which is just a particular case) are presented here, without using arcane terminology or complicated equations. Numeration systems discussed here are a particular case of deterministic sequences behaving just like the stochastic process investigated earlier, in particular the logistic map, which is a particular case.

- Application: Random Number Generation
- Chaotic Sequences Representing Numbers
- Data Science and Mathematical Engineering
- Numbers in Base 2, 10, 3/2 or π
- Nested Square Roots and Logistic Map
- About the Randomness of the Digits of π
- The Digits of π are Randomly Distributed in the Logistic Map System
- Paths to Proving Randomness in the Decimal System
- Connection with Brownian Motions
- Randomness and the Bad Seeds Paradox
- Application to Cryptography, Financial Markets, Blockchain, and HPC
- Digits of π in Base π

10. Numeration Systems in One Picture

Here you will find a summary of much of the material previously covered on chaotic systems, in the context of numeration systems (in particular, chapters 7 and 9.)

- Summary Table: Equilibrium Distribution, Properties
- Reverse-engineering Number Representation Systems
- Application to Cryptography

11. Numeration Systems: More Statistical Tests and Applications

In addition to featuring new research results and building on the previous chapters, the topics discussed here offer a great sandbox for data scientists and mathematicians.

- Components of Number Representation Systems
- General Properties of these Systems
- Examples of Number Representation Systems
- Examples of Patterns in Digits Distribution
- Defects found in the Logistic Map System
- Test of Uniformity
- New Numeration System with no Bad Seed
- Holes, Autocorrelations, and Entropy (Information Theory)
- Towards a more General, Better, Hybrid System
- Faulty Digits, Ergodicity, and High Precision Computing
- Finding the Equilibrium Distribution with the Percentile Test

- Central Limit Theorem, Random Walks, Brownian Motions, Stock Market Modeling
- Data Set and Excel Computations

12. The Central Limit Theorem Revisited

The central limit theorem explains the convergence of discrete stochastic processes to Brownian motions, and has been cited a few times in this book. Here we also explore a version that applies to deterministic sequences. Such sequences and treated as stochastic processes in this book.

- A Special Case of the Central Limit Theorem
- Simulations, Testing, and Conclusions
- Generalizations
- Source Code

13. How to Detect if Numbers are Random or Not

We explore here some deterministic sequences of numbers, behaving like stochastic processes or chaotic systems, together with another interesting application of the central limit theorem.

- Central Limit Theorem for Non-Random Variables
- Testing Randomness: Max Gap, Auto-Correlations and More
- Potential Research Areas
- Generalization to Higher Dimensions

14. Arrival Time of Extreme Events in Time Series

Time series, as discussed in the first chapters, are also stochastic processes. Here we discuss a topic rarely investigated in the literature: the arrival times, as opposed to the extreme values (a classic topic), associated with extreme events in time series.

- Simulations
- Theoretical Distribution of Records over Time

15. Miscellaneous Topics

We investigate topics related to time series as well as other popular stochastic processes such as spatial processes.

- How and Why: Decorrelate Time Series
- A Weird Stochastic-Like, Chaotic Sequence
- Stochastic Geometry, Spatial Processes, Random Circles: Coverage Problem
- Additional Reading (Including Twin Points in Point Processes)

16. Exercises

1. Introduction to Stochastic Processes

We introduce these processes, used routinely by Wall Street quants, with a simple approach consisting of re-scaling random walks to make them time-continuous, with a finite variance, based on the central limit theorem.

Stochastic processes have many applications, including in finance and physics. It is an interesting model to represent many phenomena. Unfortunately the theory behind it is very difficult, making it accessible to a few 'elite' data scientists, and not popular in business contexts.

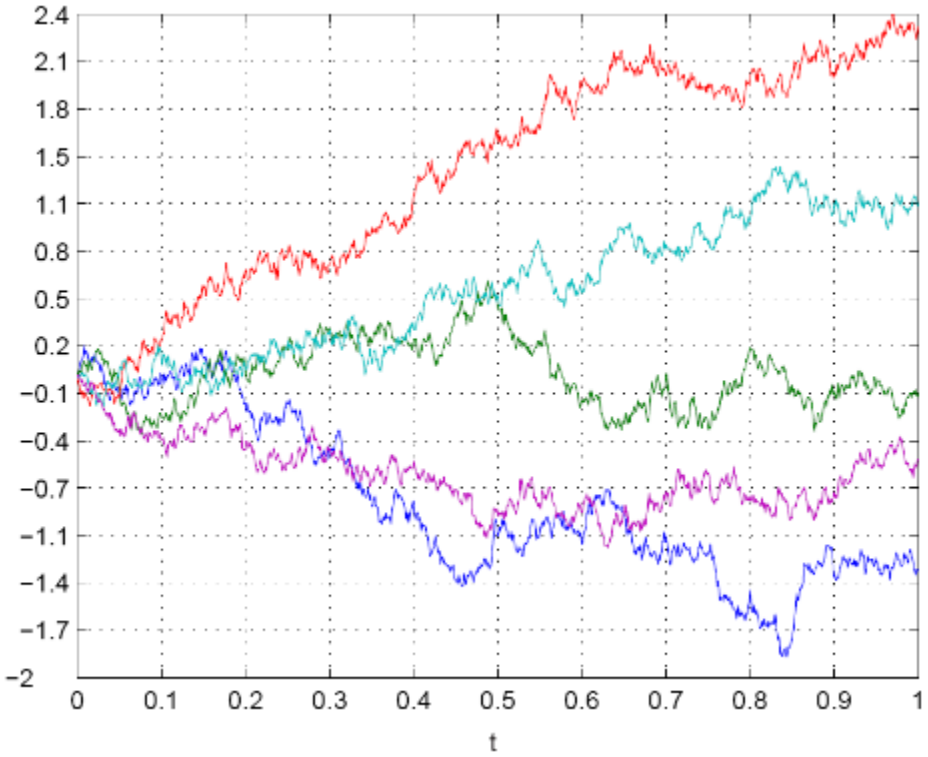
One of the most simple examples is a random walk, and indeed easy to understand with no mathematical background. However, time-continuous stochastic processes are always defined and studied using advanced and abstract mathematical tools such as measure theory, martingales, and filtration. If you wanted to learn about this topic, get a deep understanding on how they work, but were deterred after reading the first few pages of any textbook on the subject due to jargon and arcane theories, here is your chance to really understand how it works.

Rather than making it a topic of interest to post-graduate scientists only, here I make it accessible to everyone, barely using any math in my explanations besides the central limit theorem (see chapter 12.) In short, if you are a biologist, a journalist, a business executive, a student or an economist with no statistical knowledge beyond Stats 101, you will be able to get a deep understanding of the mechanics of complex stochastic processes, after reading this article. The focus is on using applied concepts that everyone is familiar with, rather than mathematical abstraction.

My general philosophy is that powerful statistical modeling and machine learning can be done with simple techniques, understood by the layman, as illustrated in my article on machine learning without mathematics or advanced machine learning with basic excel.

1. Construction of Time-Continuous Stochastic Processes: Brownian Motion

Probably the most basic stochastic process is a random walk (see chapter 3) where the time is discrete. The process is defined by $X(t+1)$ equal to $X(t) + 1$ with probability 0.5, and to $X(t) - 1$ with probability 0.5. It constitutes an infinite sequence of auto-correlated random variables indexed by time. For instance, it can represent the daily logarithm of stock prices, varying under market-neutral conditions. If we start at $t = 0$ with $X(0) = 0$, and if we define $U(t)$ as a random variable taking the value +1 with probability 0.5, and -1 with probability 0.5, then $X(n) = U(1) + ... + U(n)$. Here we assume that the variables $U(t)$ are independent and with the same distribution. Note that $X(n)$ is a random variable taking integer values between $-n$ and $+n$.



Five simulations of a Brownian motion (X-axis is the time t, Y-axis is Z(t))

What happens if we change the time scale (*X*-axis) from daily to hourly, or to every millisecond? We then also need to re-scale the values (*Y*-axis) appropriately; otherwise the process exhibits massive oscillations (from *-n* to *+n*) in very short time periods. At the limit, if we consider infinitesimal time increments, the process becomes a continuous one. Much of the complex mathematics needed to define these continuous processes do no more than finding the correct re-scaling of the *Y*-axis, to make the limiting process meaningful.

You can define these time-continuous processes as the limit of their time-discrete version: using the correct re-scaling is straightforward. Let us define $Y(t, n)$ as the same process as $X(t)$, but with small time increments of T/n instead of T, where T is the time unit (say, a day). In other words, $Y(t/n, n) = X(t)$: we just re-scaled the time axis; both t and n are still integers. Now $Y(t, n)$ can take on very high values (between $-n$ and $+n$) even when $t = 1$. Thus we also need to re-scale the *y*-axis.

Note that

$$\mathrm{Var}[Y(1,n)] = \sum_{t=1}^{n} \mathrm{Var}[U(t)] = n \cdot \mathrm{Var}[U(1)]$$

The only way to make the right-hand side of the equation not depending on n is to re-scale Y as follows. Define

$$Z(t,n) = \frac{Y(t,n)}{\sqrt{n}}$$
$$Z(t) = \lim_{n\to\infty} Z(t,n)$$

Then

$$\text{Var}[Z(1,n)] = \text{Var}[U(1)]$$
$$\text{Var}[Z(t,n)] = t \cdot \text{Var}[U(1)]$$
$$\text{Var}[Z(t)] = t \cdot \text{Var}[U(1)]$$

Also, because of the central limit theorem, by construction $Z(t)$ has a Gaussian distribution, regardless of the distribution of $U(1)$. The final process $Z(t)$ is both time-continuous and continuous on the Y-axis, though nowhere differentiable. It looks like a fractal and it is known as a Brownian motion -- the standard time-continuous stochastic process -- from which many other processes have been derived.

Note that if instead of using a binary random variable for $U(1)$, you use a Gaussian one, then the limiting process $Z(t)$ is identical, but we are dealing with Gaussian variables throughout the construction, making it easier to study the covariance structure and other properties. It then becomes a simple exercise for any college student, to derive the covariance between $Z(t)$ and $Z(s)$. The covariance can also be estimated using simulations.

Finally, note that $Z(0) = 0$ and $E[U(1)] = 0$.

2. General Properties

The Brownian motion can also be viewed as a Gaussian stationary time series, characterized by its covariance or auto-correlation structure. It is also related to deterministic dynamical systems (see chapters 6 and 7) that exhibit a fractal behavior. Under appropriate transformations, all these processes can be made equivalent.

One question is whether the above construction (the limit of a time-discrete random walk) covers all types of Brownian motions, or only a few particular cases. One way to investigate this is to check whether this construction can generate any kind of covariance structure that characterizes these processes. The answer is positive, making advanced mathematical theory unnecessary to build and study Brownian motions, as well as the numerous complex stochastic processes derived from this base process.

Indeed, if you allow the random variables $U(t)$ used in our construction to NOT be independent, then you can build more sophisticated time-continuous stochastic processes, that are not Brownian motions.

Definitions
All the stochastic processes introduced so far, whether time-discrete or time-continuous, share the following properties. In most cases, it is easy to turn a stochastic process into one that satisfies these properties, using simple transformations, as illustrated later in this section.

- **Stationarity**: This is sometimes called the *homogeneous* property, and represents the fact that there is no trend, drift, or more precisely, the fact that the properties of the process in question do not explicitly depend on the time parameter t. Such processes are usually characterized by their auto-correlation structure alone.

- **Ergodicity**: This means that one instance of the process is enough to derive all its properties. You don't need to make hundreds of simulations to study the process' properties or compute estimates: One simulation over a very long time period will do.

- **Fractal behavior**: If you zoom in or out on any single realization of these processes, you will get a new process with the exact same properties and behavior, indistinguishable from the parent process. Two different time windows provide two versions of the process that are identical with respect to their statistical properties.

- **Memory-less**: The future observations are depending on the present value only, not on past observations. This is sometimes referred to as the *Markov property*.

It is sometimes possible to transform a process so that it satisfies some of the above properties. For instance, if $X(t)$ is a time series with a linear trend and discrete time increments, the differences $X(t) - X(t-1)$ may represent a stationary time series. Likewise, if $X(t)$ depends on $X(t-1)$ and $X(t-2)$, then the vector $(X(t), Y(t))$ with $Y(t) = X(t-1)$ represents a bivariate memory-less time series.

Exercise
Simulate 1,000 realizations of a Brownian motion on [0, 1], using the random walk construction described in this article. Study the distribution of the following quantities, using estimates based on your simulations. In particular, what is the mean and variance, for t in [0, 1], for the following quantities:

- min $Z(t)$, max $Z(t)$ over [0, 1]
- proportion of time with $Z(t) > 0$ (note that $Z(0) = 0$)
- number of times the sign of $Z(t)$ oscillates

Keep in mind that the $Z(t)$'s are auto-correlated. Given a particular realization of a stochastic process, these statistics can be used to check if it is a Brownian motion, or not. Another interesting exercise is to study the process in question if the variable $U(1)$ does not have a variance, for instance if $U(1)$ has a Cauchy distribution centered at 0.

2. Integration, Differentiation, Moving Averages

We introduce more advanced concepts about stochastic processes. Yet we make these concepts easy to understand even to the non-expert. This is a follow-up to Chapter 1.

In chapter 1, I introduced some of the complex stochastic processes used by Wall Street data scientists, using a simple approach that can be understood by people with no statistics background other than a first course such as stats 101. I defined and illustrated the continuous Brownian motion (the mother of all these stochastic processes) using approximations by discrete random walks, simply re-scaling the X-axis and the Y-axis appropriately, and making time increments (the X-axis) smaller and smaller, so that the limiting process is a time-continuous one. This was done without using any complicated mathematics such as measure theory or filtrations.

Here I am going one step further, introducing the integral and derivative of such processes, using rudimentary mathematics. All the articles that I've found on this subject are full of complicated equations and formulas. It is not the case here. Not only do I explain this material in simple English, but I also provide pictures to show how an Integrated Brownian motion looks like (I could not find such illustrations in the literature), how to compute its variance, and focus on applications, especially to number theory, Fintech and cryptography problems. Along the way, I discuss moving averages in a theoretical but basic framework (again with pictures), discussing what the optimal window should be for these (time-continuous or discrete) time series.

1. General Framework

As in my previous article, we define a time-continuous process as the limit of a time-discrete process. The time-discrete process is referred to as the *base process*. The fundamental example is as follows:

Start with discrete random variables $U(k)$, $k = 1, 2$, and so on (the base process) that are independently and identically distributed, with mean equal to 0. Typically, the time series $\{ U(k) \}$ is a white noise.

1. Define $X(n) = U(1) + ... + U(n)$
2. Standardize $X(n)$ so that its variance does not depend on n, that is, introduce $Y(n) = X(n) / \text{SQRT}(n)$. This step consists of re-scaling the Y-axis.
3. Re-scale the time-axis (the X-axis) so that time increments are now equal to $1/n$ rather than 1, and let n tends to infinity. The limiting variable for $Y(n)$, as n tends to infinity, is denoted as $Z(1)$. The value at time t (t being continuous this time) is the limiting value of $Y(\text{INT}(nt))$ as n tends to infinity, where INT is the integer part function, and it is denoted as $Z(t)$.

The collection of random variables { $Z(t)$ } defines the resulting, time-continuous, properly re-scaled stochastic process. In this case, Var[$Z(t)$] = t Var[$U(1)$]. Also $Z(t)$ has a Gaussian distribution, by construction and by virtue of the central limit theorem (see chapter 12.) This process is known as a Brownian motion. The initial random variables $U(k)$ could be Gaussian, or uniform on {-1, +1}, or uniform on [-1, +1]. It does not matter. See illustration in chapter 1.

This process is continuous everywhere but nowhere differentiable. Thus the idea to build processes that are derived from { $Z(t)$ }, but smoother (differentiable everywhere.)

We introduce two such types of processes that meet this goal:

- The cumulative or *integrated process* { $S(t)$ } derived from { $Z(t)$ }
- The theoretical *moving average process* { $M(t)$ } derived from { $Z(t)$ }

Finally, we also define the inverse operation of integration as differentiation.
The *differentiated process* of $S(t)$ is $Z(t)$. In practice, the smoother (integrated or moving average) process is easier to study and sometimes displays patterns that can't be identified in the original process. More on this in the last section.

2. Integrated, Moving Average and Differential Process

Here we define three operations to transform a stochastic process into another one, hopefully more useful for interpretation and decision making, than the original process: Integration, differentiation, and moving average. In all three cases, the construction follows the same principle: In the construction of the Brownian motion described in the previous section, replace $X(n) = U(1) + ... + U(n)$ by $X(n) = V(1) + ... + V(n)$, where $V(k)$ is described below for each transformation. We also discuss some of the challenges to make this methodology mathematically robust.

- **Integrated Process**, Construction: $V(k) = U(1) + ... + U(k)$.

- **Differential Process**, Construction: $V(k) = U(k+1) - U(k)$. If { $Z(t)$ } is the Brownian motion described in the introduction, then the resulting process is a white noise: continuous and differentiable nowhere.

- **Moving Average Process**, Construction: $V(k) = U(k) + U(k + 1) + ... + U(k + h(k))$ where $h(k)$ is as small as possible to make the resulting process continuous and differentiable everywhere. For Brownian motions, $h(k) = SQRT(k)$ works.
  Does $h(k) = \log(k)$ work? This would make the resulting process far more similar to the original one, but maybe barely (if at all) continuous -- in other words, more chaotic than with $h(k) = SQRT(k)$.

Challenges

The general construction procedure described above needs to be further studied from a theoretical point of view, for the following reasons:

- Are the derived processes depending on *U*(1), *U*(2) and so on? This should not be the case. This issue is especially critical for the differentiated process.
- Is the differentiation operation truly the reverse of integration?
- Are my definitions of differential and integrated processes compatible with or equivalent to the highly technical definitions found in the literature?

Proper Re-scaling and Variance Computation

The second step in the general framework (see first section) needs to be adjusted to get things right, when constructing differential, integrated, or moving average processes. In short, you must keep the variance of *X*(*n*) not depending on *n* as *n* tends to infinity. Let us show you how it works for the integrated Brownian motion. In this case, for the integrated process, we have:

$$X(n) = \sum_{k=1}^{n} V(k) = \sum_{k=1}^{n} k \cdot U(n-k+1).$$

Thus,

$$\mathbf{Var}[X(n)] = \sum_{k=1}^{n} k^2 \cdot \mathbf{Var}[U(1)] = \frac{n(n+1)(2n+1)}{6} \cdot \mathbf{Var}[U(1)] \sim \frac{n^3}{3}\mathbf{Var}[U(1)].$$

So, the proper re-scaling factor for the Y-axis, as *n* tends to infinity, is in this case *Y*(*n*) = *X*(*n*) / SQRT(n^3/3). Using similar arguments, one can easily prove that

$$\mathbf{Var}[S(t)] = t^3 \cdot \mathbf{Var}[U(1)].$$

Also, *S*(*t*) has a Gaussian distribution for the same reasons as described in the first section. The same logic applies to compute Var[*M*(*t*)]. The details are left as an exercise. A more complicated exercise consists of computing the covariance between *S*(*t*) and *S*(*t* - s) for *s* > 0, and proving that {*S*(*t*) } is NOT a Brownian motion itself (being differentiable everywhere unlike Brownian motions.)

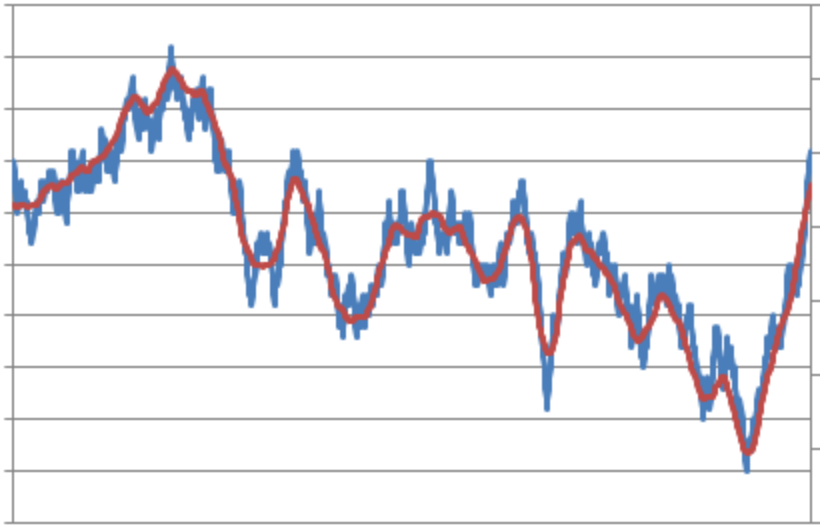


Figure 1: Brownian motion realization (blue) and its moving average (red)

In Figures 1 and 2, the X-axis represents the time axis, between 0 and 1. The Brownian motion is continuous everywhere while differentiable nowhere. To the contrary, the moving average and integrated processes are both continuous and differentiable everywhere.

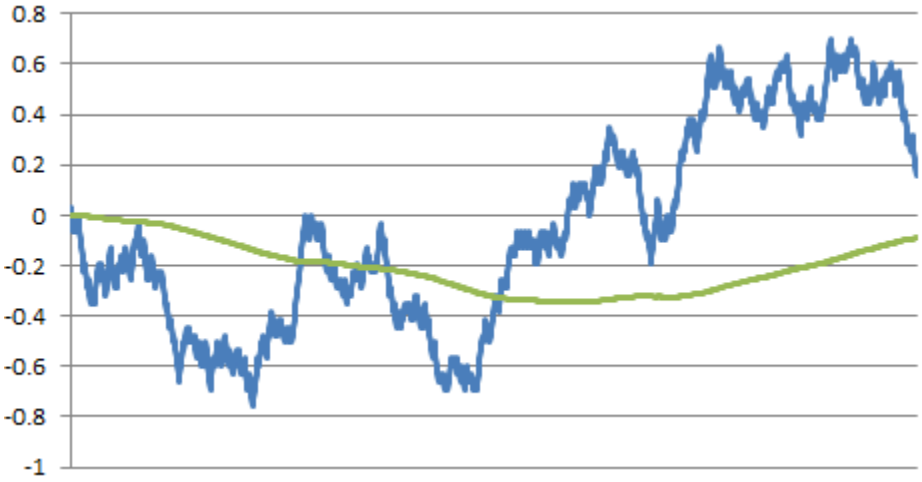


Figure 2: Brownian motion realization (blue) and its integrated process (green)

3. Application to Number Theory

The purpose here is to study pairs of large prime numbers p and q with $p < q$, used to design cryptographic keys $m = pq$. Some of these primes exhibit strong patterns that make them unsuitable for use in highly secure cryptographic systems. It makes it less difficult to factor m. Factoring a product of two large primes, each with hundreds of digits, is usually an intractable problem if p and q are carefully chosen. In other words, we are looking for primes p and q that are not as "random" (or to put it differently, less strongly prime) than your typical large prime numbers. Factoring m allows you to break the key in cryptographic systems, something you want to avoid precisely by carefully choosing p and q.

We discuss here one example of such bad pair, namely p = 1,879 and q = 3,803. We use the same construction technique as outlined in the first section, resulting in a process that looks exactly like a Brownian motion up to some value of t, then suddenly exhibits two big jolts very early on the time scale, in this particular example.

The base process { U(k) } (see first section) is defined by

$$c(k) = \frac{m}{k} - \left\lfloor \frac{m}{k} \right\rfloor,$$

$$\text{If } c(k) > 0.5 \text{ then } U(k) = -1 + c(k), \text{ else } U(k) = c(k)$$

The brackets in the formula represent the integer part function. The resulting process { $Z(t)$ } created using the technique described in the first section, is a Brownian motion up to the first jolt occurring at k = SQRT(m); the second jolt occurs at $k = q$. Note that for small k's, { $U(k)$ } is reminiscent of number representations in various systems, as described in chapter 9 and 10. In other examples, the interesting jolt occurs at $k = p$ (the smaller prime.) In most cases, no obvious patterns are found.

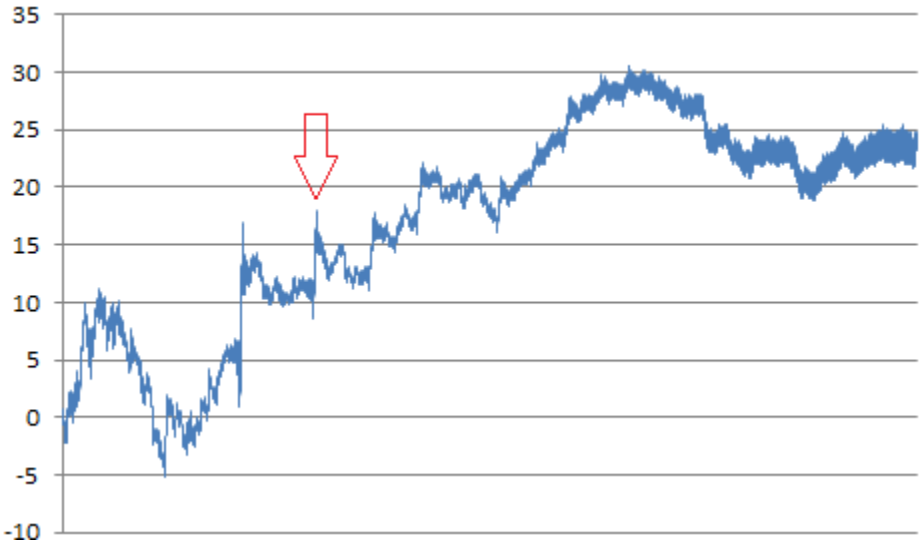


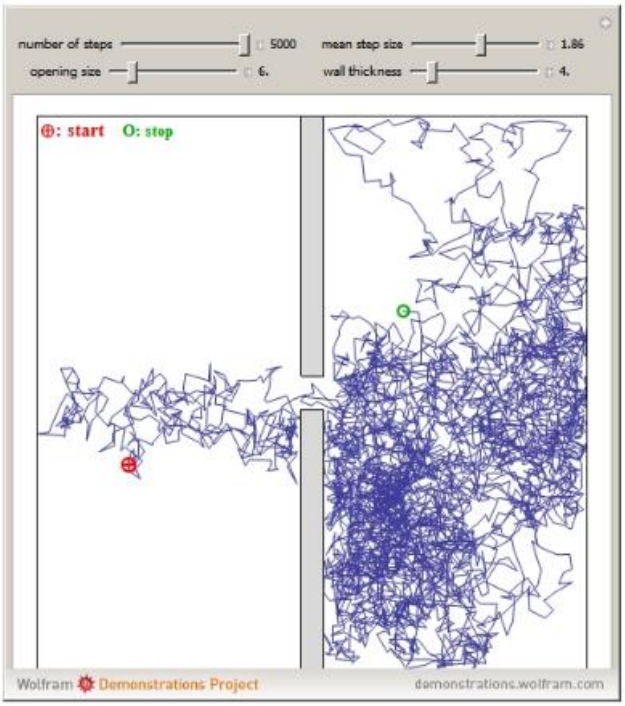
Figure 3: Brownian motion up to first jolt; second jolt used to break cryptographic key

Figure 3 is based on the first 25,000 observations. The base process { $U(k)$ } does not display these jolts. They are only visible in the process { $Z(t)$ } pictured in Figure 3. Detecting these jolts (the second one) is of particular interest since it allows you to factor m to retrieve the two large prime factors p and q. To do this, one solution consists of using sampled values of $U(k)$ together with interpolation techniques. Note that as k becomes large (higher than 12,000) the $U(k)$'s become so heavily auto-correlated that the process { $Z(t)$ } is no longer a Brownian motion. In some ways, it could be used as a better model to simulate stock market prices, than standard Brownian motions. For testing purposes, tables of large prime numbers are easy to find on the Internet, for instance here. Details about the computations are found in in this spreadsheet.

3. Self-Correcting Random Walks

We investigate here a breed of stochastic processes that are different from the Brownian motion, yet are better models in many contexts, including Fintech.

This is another off-the-beaten-path problem, one that you won't find in textbooks. You can solve it using data science methods (my approach) but the mathematician with some spare time could find an elegant solution. My Excel spreadsheet with all computations is accessible from this article. You don't need a deep statistical background to quickly discover some fun and interesting results playing with this stuff. Computer scientists, software engineers, quants, BI and analytic professionals from beginners to veterans, will also be able to enjoy it!



Source for picture: 2-D constrained random walk (snapshot - video available here)

1. The problem

We are dealing with a stochastic process barely more complicated than a random walk. Random walks are also called *drunken walks*, as they represent the path of a drunken guy moving left and right seemingly randomly, and getting lost over time. Here the process is called *self-correcting random walk* or also *reflective random walk*, and is related to *controlled random walks*, and *constrained random walks* (see also here) in the sense that the walker, less drunk than in a random walk, is able to correct any

departure from a straight path, more and more over time, by either slightly over- or under-correcting at each step. One of the two model parameters (the positive parameter *a*) represents how drunk the walker is, with *a* = 0 being the worst. Unless *a* = 0, the amplitude of the corrections decreases over time to the point that eventually (after many steps) the walker walks almost straight and arrives at his destination. This model represents many physical processes, for instance the behavior of a stock market somewhat controlled by a government to avoid bubbles and implosions, or when it hits a symbolic threshold and has a hard time breaking through. It is defined as follows:

Let's start with $X(1) = 0$, and define $X(k)$ recursively as follows, for $k > 1$:

$$X(k) = X(k-1) + \frac{U(k)}{k^a} \text{ if } X(k-1) < 0$$

$$X(k) = X(k-1) - \frac{U(k)}{k^a} \text{ if } X(k-1) \geq 0$$

and let's define $U(k)$, $Z(k)$, and Z as follows:

$$Z(k) = k^a X(k)$$

$$Z = \lim_{k \to \infty} Z(k)$$

$$U(k) = V(k)^b$$

where the $V(k)$'s are deviates from *independent* uniform variables on [0, 1], obtained for instance using the function RAND in Excel. So there are two *positive* parameters in this problem, *a* and *b*, and $U(k)$ is always between 0 and 1. When $b = 1$, the $U(k)$'s are just standard uniform deviates, and if $b = 0$, then $U(k) = 1$. The case $a = b = 0$ is degenerate and should be ignored. The case $a > 0$ and $b = 0$ is of special interest, and it is a number theory problem in itself, related to this problem when $a = 1$. Also, just like in random walks or Markov chains, the $X(k)$'s are not independent; they are indeed highly auto-correlated.

Prove that if $a < 1$, then $X(k)$ converges to 0 as *k* increases. Under the same condition, prove that the limiting distribution *Z*

- always exists, (Note: if $a > 1$, $X(k)$ may not converge to zero, causing a drift and asymmetry)
- always takes values between -1 and +1, with $\min(Z) = -1$ and $\max(Z) = +1$,
- is symmetric, with mean and median equal to 0,
- and does not depend on *a,* but only on *b.*

For instance, for $b = 1$, even $a = 0$ yields the same triangular distribution for *Z*, as any $a > 0$.

If $a < 1$ and $b = 0$, (the non-stochastic case) prove that

- *Z* can only take 3 values: -1 with probability 0.25, +1 with probability 0.25, and 0 with probability 0.50

- If $U(k)$ and $U(k+1)$ have the same sign, then $U(k+2)$ is of opposite sign

And here is a more challenging question: In general, what is the limiting distribution of Z? Also, what happens if you replace the $U(k)$'s with (say) Gaussian deviates? Or with $U(k) = |\sin(k*k)|$ which has a somewhat random behavior?

2. Hints to solve this problem

It is necessary to use a decent random number generator to perform simulations. Even with Excel, plotting the empirical distribution of $Z(k)$ for large values of k, and matching the kurtosis, variance and empirical percentiles with those of known statistical distributions, one quickly notices that when $b = 1$ (and even if $a = 0$) the limiting distribution Z is well approximated by a symmetric triangular distribution on [-1, 1], and thus centered on 0, with a kurtosis of -3/5 and a variance of 1/6. In short, this is the distribution of the difference of two uniform random variables on [0, 1]. In other words, it is the distribution of $U(3) - U(2)$. Of course, this needs to be proved rigorously. Note that the limiting distribution Z can be estimated by computing the values $Z(n+1)$, ..., $Z(n+m)$ for large values of n and m, using just one instance of this simulated stochastic process. See chapter 13 for details.

Does it generalize to other values of b? That is, does Z always have the distribution of $U(3) - U(2)$? Obviously not for the case $b = 0$. But it could be a function, combination and/or mixture of $U(3)$, $-U(2)$, and $U(3) - U(2)$. This works both for $b = 0$ and $b = 1$.

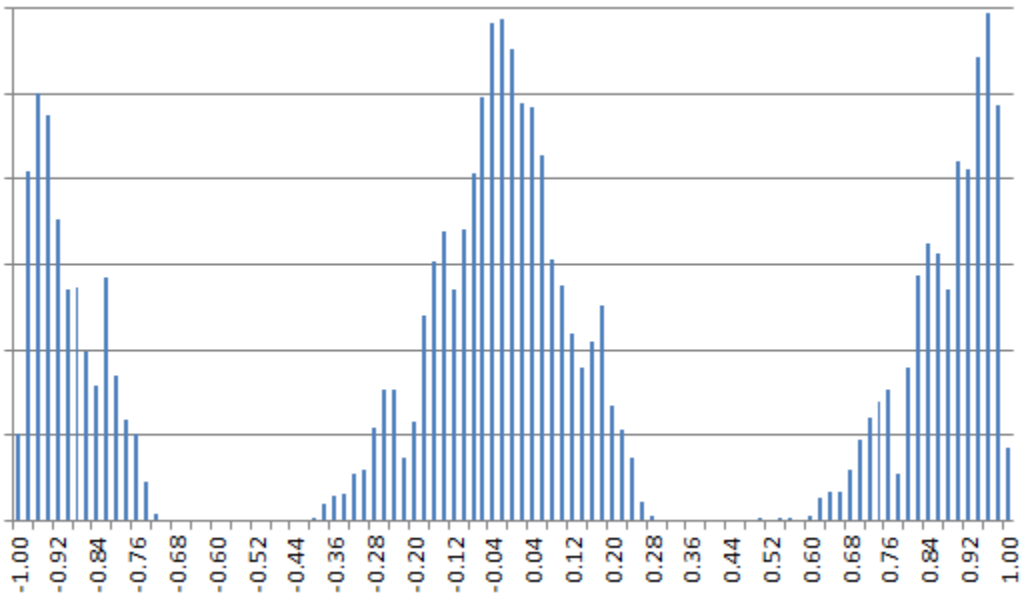


Figure 1: Mixture-like distribution of Z (estimated) when b = 0.01 and a = 0.8

Interestingly, for small values of b, the limiting distribution Z looks like a mixture of (barely overlapping) simple distributions. So it could be used as a statistical model in clustering problems, each component of the mixture representing a cluster. See Figure 1.

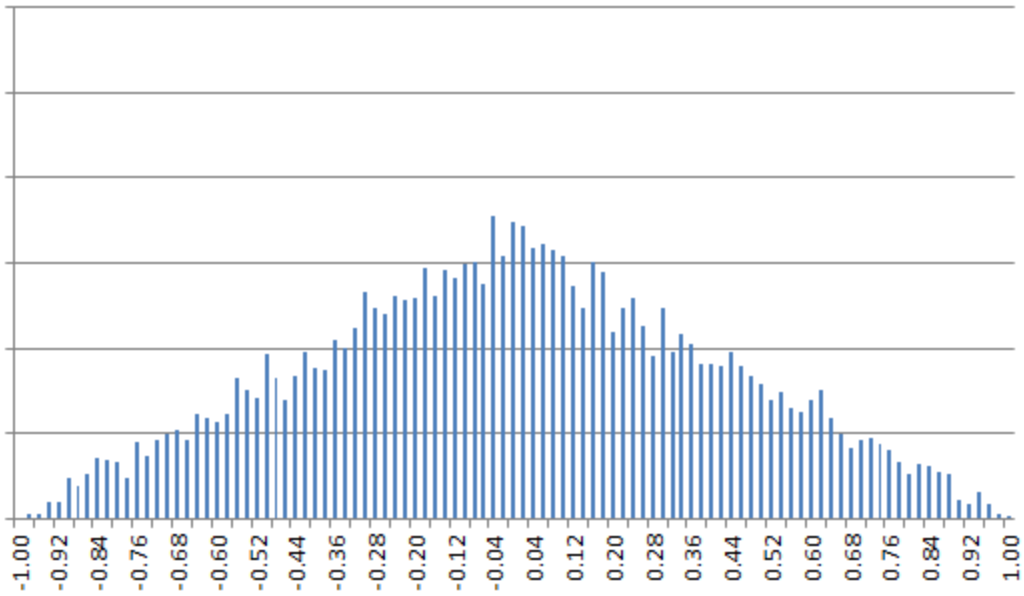


Figure 2: Triangular distribution of Z (estimated) when b = 1 and a = 0.8

The spreadsheet with all computations and model fitting can be downloaded here.

3. Deeper dive

So far, my approach has been data science oriented: it looks more like guesswork. Here I switch to mathematics, to try to derive the distribution of *Z*. Since it does not depend on the parameter *a*, let us assume here that *a* = 0. Note that when *a* = 0, *X*(*k*) does not converge to zero; instead *X*(*k*) = Z(*k*) and both converge in distribution to *Z*. It is obvious that *X*(*k*) is a mixture of distributions, namely *X*(*k*-1) + *U*(*k*) and *X*(*k*-1) - *U*(*k*). Since *X*(*k*-1) is in turn a mixture, *X*(*k*) is actually a mixture of mixtures, and so on, In short, it has the distribution of some nested mixtures.

As a starting point, it would be interesting to study the variance of *Z* (the expectation of Z is equal to 0.) The following formula is incredibly accurate for any value of *b* between 0 and 1, and even beyond. It is probably an exact formula, not an approximation. It was derived using the tentative density function obtained at the bottom of this section, for Z:

$$\mathrm{Var}(Z) = \frac{b+1}{3(3b+1)}$$

It is possible to obtain a functional equation for the distribution *P*(*Z* < *z*), using the equations that define *X*(*k*) in section 1, with *a* = 0, and letting *k* tends to infinity. It starts with

$$P(X(k) < z) = \int_{-1}^{1} P(X(k) < z | X(k-1) = x) f_{X(k-1)}(x) dx$$

Let's introduce *U* as a random variable with the same distribution as *U*(*k*) or *U*(2).
As *k* tends to infinity, and separating the two cases *x* negative and *x* positive, we get

$$P(Z<z)=\int_0^1 P(U>x-z)f_Z(x)dx+\int_{-1}^0 P(U<z-x)f_Z(x)dx$$

Taking advantages of symmetries, this can be further simplified to

$$F_Z(z)=\frac{1}{2}+\int_0^1\{F_U(x+z)-F_U(x-z)\}f_Z(x)dx$$

where *F* represents the distribution function, *f* represents the density function, and *U* has the same distribution as *U*(2), that is

$$F_U(y)=y^{1/b},\ \text{for } y \text{ in } [0,1].$$

Taking the derivative with respect to *z*, the functional equation becomes the following Fredholm integral equation, the unknown being *Z*'s density function:

$$f_Z(z)=\int_0^1\{f_U(x+z)+f_U(x-z)\}f_Z(x)dx$$

We have the following particular cases:

- When *b* tends to zero, the distribution of *Z* converges to a uniform law on [-1, 1] thus with a variance equal to 1/3.
- When *b* = 1/2, *Z* has a parabolic distribution on [-1, +1], defined by P(*Z* < *z*) = (2 + 3*z* - z^3) / 4. This needs to be proved, for instance by plugging this parabolic distribution in the functional equation, and checking that the functional equation is verified if *b* = 1/2. However, a constructive proof would be far more interesting.
- When *b* = 1, *Z* has the triangular distribution discussed earlier. The density function for *Z*, defined as the derivative of P(*Z* < *z*) with respect to *z*, is equal to 1 - |*z*| when *b* = 1, and 3 (1 - z^2) / 4 when *b*= 1/2.

So for *b* = 1, *b* = 1/2, or the limiting case *b* = 0, we have the following density for *Z*, defined on [-1, 1]:

$$f_Z(z)=\frac{b+1}{2}\cdot(1-|z|^{1/b})$$

Is this formula valid for any *b* between 0 and 1? This is still an open question. The functional equation applies regardless of *U*'s distribution though, even if exponential or Gaussian. The complexity in the cases discussed here, arises from the fact that *U*'s density is not smooth enough, due to its bounded support domain [0, 1] (outside the support domain, the density is equal to 0.) A potential more generic version of the previous formula would be:

$$f_Z(z)=\frac{1}{2}\cdot\frac{1-F_U(|z|)}{1-\int_0^1 F_U(x)dx}=\frac{1-F_U(|z|)}{2E(U)}$$

where *E* denotes the expectation. However, I haven't checked whether and under which conditions this formula is correct or not, except for the particular cases of *U* discussed here. One of the requirements is that the support domain for *U* is [0, 1]. If this formula is not exact in general, it might still be a good approximation in some cases.

Connection with topics covered later in this book
Solving this type of equation (actually, a stochastic integral equation) is discussed in details in chapter 7 and 11, respectively in the contexts of the logistic map and numeration systems. The distribution, solution of the equation, is called the *equilibrium distribution*.

4. Potential Areas of Research

Here are a few interesting topics for research:

- Develop a 2-D or 3-D version of this process, investigate potential applications in thermodynamics or statistical mechanics, for instance modeling movements of gas molecules in a cube as the temperature goes down (a >0) or is constant (a = 0), and comparison with other stochastic processes used in similar contexts..
- Continuous version of the discrete reflective random walk investigated here, with a = 0, and increments $X(k)$ - $X(k$-1) being infinitesimally small, following a Gaussian rather than uniform distribution. The limiting un-constrained case is known as a *Wiener process* or *Brownian motion.* What happens if this process is also constrained to lie between -1 and +1 on the Y-axis? This would define a reflected Wiener process, see also here for a similar process, and also here.
- Another direction is to consider the one-dimensional process as time series (which economists do) and to study the multivariate case, with multiple cross-correlated time series.
- For the data scientist, it would be worth checking whether and when, based on cross-validation, my process provides better model fitting, leading to more accurate predictions and thus better stock trading ROI (than say a random walk, after removing any trend or drift) when applied to real stock market data publicly available.

This is the kind of mathematics used by Wall Street quants and in operations research. Hopefully my presentation here is much less arcane than the traditional literature on the subject, and accessible to a much broader audience, even though it features the complex equations characterizing such a process (and even hinting to a mathematical proof that is not as difficult as it might seem at first glance, and supported by simulations). Note that my reflective random walk is not a true random walk in the classical sense of the term: A better term might be more appropriate.

5. Solution for the (non-stochastic) case b = 0

We have the following result:

- If a < = 1, then the sequence $\{X(k)\}$ converges to zero.
- If a = 3, $\{X(\text{k})\}$ converges to Zeta(3) - 5/4 =~ -0.048.
- If a = 4, $\{X(k)\}$ converges to $(\pi^4/ 90)$ - 9/8 =~ -0.043.

You can read the proof here. Much more can also be explored regarding the case b = 0. For instance, when a = 1 and b = 0, the problem is similar to this one, where we try to

approximate the number 2 by converging sums of elementary positive fractions without ever crossing the boundary Y= 2, staying below at all times. Here, by contrast, we try to approximate 0, also by converging sums of the same elementary fractions, but allowing each term to be either positive or negative, thus crossing the boundary Y = 0 very regularly. The case with alternating signs for $X(k)$, is a problem of interest: It shows strong patterns.

4. Stochastic Processes and New Tests of Randomness

In this transition chapter, we introduce a different type of stochastic process, with number theory and cryptography applications, analyzing statistical properties of numeration systems along the way -- a recurrent theme in the next chapters, offering many research opportunities and applications. While we are dealing with deterministic sequences here, they behave very much like stochastic processes, and are treated as such. Statistical testing is central to this chapter, introducing tests that will be also used in the last chapters.

This article is intended for practitioners who might not necessarily be statisticians or statistically-savvy. The mathematical level is kept as simple as possible, yet I present an original, simple approach to test for randomness, with an interesting application to illustrate the methodology. This material is not something usually discussed in textbooks or classrooms (even for statistical students), offering a fresh perspective, and out-of-the-box tools that are useful in many contexts, as an addition or alternative to traditional tests that are widely used. This chapter is written as a tutorial, but it also features an interesting research result in the last section.

1. Context

Let us assume that you are dealing with a time series with discrete time increments (for instance, daily observations) as opposed to a time-continuous process. The approach here is to apply and adapt techniques used for time-continuous processes, to time-discrete processes. More specifically (for those familiar with stochastic processes) we are dealing here with discrete Poisson processes. The main question that we want to answer is: Are some events occurring randomly, or is there a mechanism making the events not occurring randomly? What is the gap distribution between two successive events of the same type?

In a time-continuous setting (Poisson process) the distribution in question is modeled by the exponential distribution. In the discrete case investigated here, the discrete Poisson process turns out to be a Markov chain, and we are dealing with geometric, rather than exponential distributions. Let us illustrate this with an example.

Example
The digits of the square root of two (SQRT(2)), are believed to be distributed as if they were occurring randomly. Each of the 10 digits 0, 1, ... , 9 appears with a frequency of 10% based on observations, and at any position in the decimal expansion of SQRT(2), on average the next digit does not seem to depend on the value of the previous digit (in short, its value is unpredictable.) An event in this context is defined, for example, as a digit being equal to (say) 3. The next event is the first time when we find a subsequent

digit also equal to 3. The *gap* (or time elapsed) between two occurrences of the same digit is the main metric that we are interested in, and it is denoted as *G*. If the digits were distributed just like random numbers, the distribution of the gap *G* between two occurrences of the same digit, would be geometric, that is,

$$P(G = k) = (1 - p)^{k-1}p, k = 1, 2, \ldots$$

with *p* = 1/10 in this case, as each of the 10 digits (0, 1, ..., 9) seems -- based on observations -- to have a frequency of 10%. We will show that this is indeed the case: In other words, in our example, the gap *G* is very well approximated by a geometric distribution of parameter *p* = 1/10, based on an analysis of the first 10 million digits of SQRT(2).

What else should I look for, and how to proceed?
Studying the distribution of gaps can reveal patterns that standard tests might fail to catch. Another statistic worth studying is the maximum gap, see chapter 14. This is sometimes referred to as extreme events / outlier analysis. Also, in our above example, studying gaps between groups of digits (not just single digits, but for instance how frequently the "word" 234567 repeats itself in the sequence of digits, and what is the distribution of the gap for that word. For any word consisting of 6 digits, *p* = 1 / 1,000,000. In our case, our data set only has 10 million digits, so you may find 234567 maybe only 2 times, maybe not even once, and looking at the gap between successive occurrences of 234567, is pointless. Shorter words make more sense. This and other issues are discussed in the next section.

2. Methodology

The first step is to estimate the probabilities *p* associated with the model, that is, the probability for a specific event, to occur at any time. It can easily be estimated from your data set, and generally, you get a different *p* for each type of event. Then you need to use an algorithm to compute the empirical (observed) distribution of gaps between two successive occurrences of the same event. In our example, we have 10 types of events, each associated with the occurrence of one of the 10 digits 0, 1,..., 9 in the decimal representation of SQRT(2). The gap computation can be efficiently performed as follows:

Algorithm to compute the observed gap distribution
Do a loop over all your observations (in our case, the 10 first million digits of SQRT(2), stored in a file; each of these 10 million digits is one observation). Within the loop, at each iteration *t*, do:

- Let *E* be the event showing up in the data set, at iteration *t*. For instance, the occurrence of (say) digit 3 in our case. Retrieve its last occurrence stored in an array, say LastOccurrences[*E*]
- Compute the gap *G* as *G* = *t* - LastOccurrences[*E*]
- Update the LastOccurrences table as follows: LastOccurrences[*E*] = *t*

- Update the gap distribution table, denoted as GapTable (a two-dimensional array or better, an hash table) as follows: GapTable[*E*, *G*]++

Once you have completed the loop, all the information that you need is stored in the GapTable summary table.

Statistical testing

If some events occur randomly, the theoretical distribution of the gap, for these events, is known to be geometric, see above formula in first section. So you must test whether the empirical gap distribution (computed with the above algorithm) is statistically different from the theoretical geometric distribution of parameter *p* (remember that each type of event may have a different *p*.) If not statistically different, then the assumption of randomness should be discarded: you've found some patterns. This work is typically done using a Kolmogorov- Smirnov test. If you are not a statistician but instead a BI analyst or engineer, other techniques can be used instead, and are illustrated in the last section:

- You can simulate events that are perfectly randomly distributed, and compare the gap distribution obtained in your simulations, with that computed on your observations. See here how to do it, especially the last comment featuring an efficient way to do it. This Monte-Carlo simulation approach will appeal to operations research analysts.

- In Excel, plot the gap distribution computed on your observations (one for each type of event), add a trendline, and optionally, display the trendline equation and its R-Squared. When choosing a trendline (model fitting) in Excel, you must select the Exponential one. This is what we did (see next section) and the good news is that, despite the very limited selection of models that Excel offers, Exponential is one of them. You can actually test other trendlines in Excel (polynomial, linear, power, or logarithmic) and you will see that by far, Exponential offers the best fit -- if your events are really randomly distributed.

Further advice

If you have collected a large number of observations (say 10 million) you can do the testing on samples of increasing sizes (1,000, 10,000, 100,000 consecutive observations and so on) to see how fast the empirical distribution converges (or not) to the theoretical geometric distribution. You can also compare the behavior across samples (cross-validation), or across types of events (variance analysis). If your data set is too small (100 data points) or your events too rare (*p* less than 1%), consider increasing the size of your data set if possible.

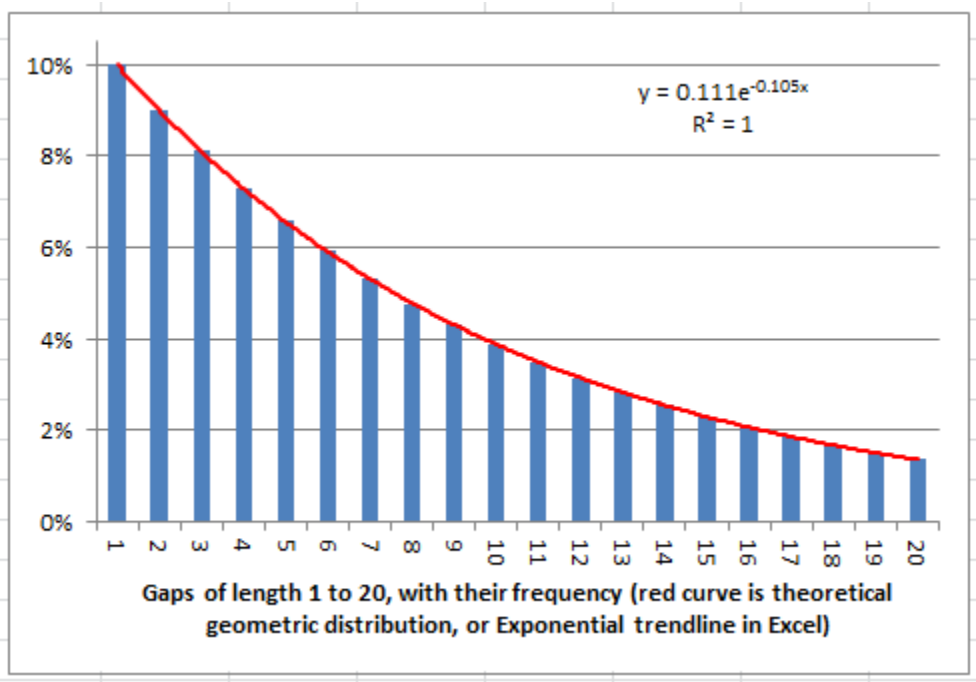
Even with big data, if you are testing a large number of rare events (in our case, tons of large "words" such as occurrences 234567 rather than single digits in the decimal representation of SQRT(2)) expect many tests to result in false negatives (failure to detect true randomness.) You can even compute the probability for this to happen,

assuming all your events are perfectly randomly distributed. This is known as the curse of big data.

3. Application to Number Theory Problem

Here, we further discuss the example used throughout this article to illustrate the concepts. Mathematical constants (and indeed the immense majority of all numbers) are thought to have their digits distributed as if they were randomly generated, see chapter 10 for details.

Many tests have been performed on many well-known constants (see here), and none of them was able to identify any departure from randomness. The gap test illustrated here is less well known, and when applied to SQRT(2), it was also unable to find departure from randomness. In fact, the fit with a random distribution, as shown in the figure below, is almost perfect.



There is a simple formula to compute any digit of SQRT(2) separately, see here, however it is not practical. Instead, we used a table of 10 million digits published here by NASA. The source claims that digits beyond the first five million have not been double-checked, so we only used the first 5 million digits. The summary gap table, methodological details, and the above picture, can be found in my spreadsheet. You can download it here.

The above chart shows a perfect fit between the observed distribution of gap lengths (averaged across the 10 digits 0, 1, ..., 9) between successive occurrences of a same

digit in the first 5 million decimals of SQRT(2), and the geometric distribution model, using the Exponential trendline in Excel.

I also explored the last 2 million decimals available in the NASA table, and despite the fact that they have not been double-checked, they also display the exact same random behavior. Maybe these decimals are all wrong but the mechanism that generates them preserves randomness, or maybe all or most of them are correct.

A counter-example
The number 0.12345678910111213141516171819202 1... known as the Champernowne constant, and obtained by concatenating the decimal representations of the natural numbers in order, has been proved to be "random", in the sense that no digit or group of digits, occurs more frequently than any other. Such a number is known as a normal number. However, it fails miserably the gap test, with the limit distribution for the gaps (if it even exists) being totally different from a geometric distribution. I tested it on the first 8, 30, 50, 100 and 400 million decimals, and you can try too, as an exercise. All tests failed dramatically.

Ironically, no one known if SQRT(2) is a normal number, yet it passed the gap test incredibly well. Maybe a better definition of a "random" number, rather than being normal, would be a number with a geometric distribution as the limit distribution for the gaps. Can you create an artificial number that passes this test, yet exhibits strong patterns of non-randomness? Is it possible to construct a non-normal number that passes the gap test?

Potential use in cryptography
A potential application is to use digits that appear to be randomly generated (like white noise, and the digits of SQRT(2) seem to fit the bill) in documents, at random positions that only the recipient could reconstruct, perhaps three or four random digits on average for each real character in the original document, before encrypting it, to increase security -- a bit like steganography. Encoding the same document a second time would result in a different kind of white noise added to the original document, and peppered randomly, each time differently -- with a different intensity, and at different locations each time. This would make the task of hackers more complicated.

4. Conclusion

Finally, this is an example where intuition can be wrong, and why you need data science. In the digits of SQRT(2), while looking at the first few thousand digits (see picture below), it looked to me like it was anything but random. There were too many 99, two few 37 (among other things), according to my intuition and visual inspection (you may call it *gut feelings*.) It turns out that I was wrong. Look at the first few thousand digits below, chances are that your intuition will also mislead you into thinking that there are some patterns. This can be explained by the fact that patterns such as 99 are easily detected by the human brain and do stand out visually, yet in this case, they do occur with the right frequency if you use analytic tools to analyze the digits.

1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753432764157273501384623091229702492483605585073721264412149709993583141322266592750559275579995050115278206057147010955997160597027453459686201472851741864088919860955232923048430871432145083976260362799525140798968725339654633180882964062061525835239505474575028775996172983557522033753185701135437460340849884716038689997069900481503054402779031645424782306849293691862158057846311159666871301301561856898723723528850926486124949771542183342042856860601468247207714358548741556570696776537202264854470158588016207584749226572260020855844665214583988939443709265918003113882464681570826301005948587040031864803421948972782906410450726368813137398552561173220402450912277002269411275736272804957381089675040183698683684507257993647290607629969413804756548237289971803268024744206292691248590521810044598421505911202494413417285314781058036033710773091828693147101711116839165817268894197587165821521282295184884720896946338628915628827659526351405422676532396946175112916024087155101351504553812875600526314680171274026539694702403005174953188629256313851881634780015693691768818523786840522878376293892143006558695686859645951555016447245098368960368873231143894155766510408839142923381132060524336294853170499157717562285497414389991880217624309652065642118273167262575395947172559346372386322614827426222086711558395999265211762526989175409881593486400834570851814722318142040704265090565323333984364578657967965192672923998753666172159825788602633636178274959942194037777536814262177387991945513972312740668983299898953867288228563786977496625199665835257761989393228453447356947949629521688914854925389047558288345260965240965428893945386466257449275563819644103169798330618520193793849400571563337205480685405758679996701213722394758214263065851322174088323829472876173936474678374319600015921888073478576172522118674904249773669292073110963697216089337086611567345853348332952546758516447107578486024636008344491148185876555542864551233142199263113325179706084365597043528564100879185007603610091594656706768836055717400767569050961367194013249356052401859991050621081635977264313806054670102935699710424251057817495310572559349844511269227803449135066375687477602831628296055324224269575345290288387684464291732827708883180870253398523381227499908123718925407264753678503048215918018866167108972869229201197599880703818543332536460211082299279293072871780799888099176741774108983060800326311816427988231171543638696617029999341616148786860180455055539869131151860103863753250045581860448040750241195184305674533683613674597374423988553285179308960373898915173195874134428817842125021916951875593444387396189314549999906107587049090260883517636224749757858858368037457931157339802099986622186949922595913276423619410592100328026149874566599688874067956167391859572888642473463585886864496822386006983352642799056283165613913942557649062065186021647263033362975075697870606606856498160092718709292153132368281356988937097416504474590960537472796524477094099241238710614470543986743647338477454819100872886222149589529591187892149179833981083788278153065562315810360648675873036014502273208829351341387227684176678436905294286984908384557445794095986260742499549168028530773989382960362133539875320509199893607513906444495768456993471276364507163279154701597733548638939423257277540038260274785674172580951416307159597849818009443560379390985590

First few hundred digits or SQRT(2). Do you see any pattern?

5. Hierarchical Processes and Web Crawling

We start discussing random number generation, and numerical and computational issues in simulations, applied to an original type of stochastic process. This will become a recurring theme in the next chapters, as it applies to many other processes.

This famous statement -- the six degrees of separation -- claims that there is at most 6 degrees of separation between you and anyone else on Earth. Here we feature a simple algorithm that simulates how we are connected, and indeed confirms the claim. We also explain how it applies to web crawlers: Any web page is connected to any other web page by a path of 6 links at most.

The algorithm below is rudimentary and can be used for simulation purposes by any programmer: It does not even use tree or graph structures. Applied to a population of 2,000,000 people, each having 20 friends, we show that there is a path involving 6 levels or intermediaries between you and anyone else. Note that the shortest path typically involves fewer levels, as some people have far more than 20 connections.

Starting with you, at level one, you have twenty friends or connections. These connections in turn have 20 friends, so at level two, you are connected to 400 people. At level three, you are connected to 7,985 people, which is a little less than 20 x 400, since some level-3 connections were already level-2 or level-1. And so on.

Level	New connections added at each level	Total reach
1	20	20
2	400	420
3	7,985	8,405
4	152,880	161,285
5	1,439,308	1,600,593
6	399,407	2,000,000

Note that in practice, people are connected through clusters of people, not randomly as in this model. Yet the simulator gives a pretty good idea of the process at play. In the above example, at level 5, you are already connected to 80% of all people.

Connections Simulator

The algorithm is described by this simple piece of Perl code:

```
$n=2000000;
$rand=0;
```

```
$npeople=1;
$new[0]=1;
for ($level=1; $level<7; $level++) {
  $new[$level]=0;
  for ($k=0; $k<$new[$level-1]; $k++) {
    $nfriends=20;
    for ($m=0; $m<$nfriends; $m++) {
      $rand=(10232193*$rand + 3701101) % 54198451371;
      $friend= $rand % $n;
      if ($a[$friend] == 0) {
        $a[$friend]++;
        $new[$level]++;
      }
    }
  }
  $npeople+=$new[$level];
  print "$level: $new[$level] - $npeople \n";
}
```

The array `a[]` stores the connections acquired at each level. The number of new connections at each level is equal to `$new[$level]`, while the total reach (at each level) is `$npeople`. Note that depending on the parameters, you might not always be able to connect everyone to everyone. This is especially true if you allow people to only have two friends, rather than 20. Just like in the real world, if some clusters of people are totally disconnected from the rest of the world, you won't find a path between you and them.

The Need for a Good Random Number Generator

I have suspected for a long time that the function rand() in Perl provides a poor random generator. In this case, it fails miserably, being able to produce only 32,768 (that is, 2^{15}) different values, while we need 2,000,000. So I replaced it by a basic formula that fits the bill: This is what the computation of the variable `$rand` is for. Note that the operator `%` in the code, represents the modulo function.

Application to Web Crawlers

When crawling the web, the best strategy is to proceed by successive layers as we did in our simulation: You start with (say) 1,000,000 seed URLs (easy to find using DMOZ or Quantcast) or even just 500 seed URLs, and at each iteration or level, you follow new links found in webpages discovered at the previous level, making sure that you rarely if ever revisit web pages that were previously crawled. It can be done on a laptop with multiple web crawlers running in parallel, if you are interested in only capturing 99.9% of the visible Internet. We did it, and here is the size of the data gathered at each level (compressed, in bytes):

Level	Size of data gathered at each level
1	2,877
2	456,084
3	8,722,723
4	26,942,508
5	39,443,366
6	42,429,041
7	13,175,749

The distribution is somewhat similar to the one shown in the first table, with a drop in the last level, and a rapid build-up during the first few levels. If you play with the simulator for a while (changing the parameters), you will occasionally stumble upon very similar distributions.

It took several months to gather all these links, using a 2-second timeout for each webpage crawl, downloading only the first 16 KB of each page, using a highly distributed architecture, and accessing only a few webpages per domain. Also, our web crawler was able to get re-started and resume where it stopped, in case our Internet connection crashed. Finally, all links that crashed were given the chance to be re-visited one more time, weeks later.

Some Mathematics

Watts and Strogatz showed that the average path length between two nodes in a random network is equal to log N / log K, where N = total nodes and K = acquaintances per node. Thus if N = 300,000,000 (90% of the US population) and K = 30 then Degrees of Separation = 5.7 and if N = 6,000,000,000 and K = 30 then Degrees of Separation = 6.6.

6. Chaotic Systems, Fractals and Quantum Algorithms

While typically studied in the context of dynamical systems, the logistic map can be viewed as a stochastic process, with an equilibrium distribution and probabilistic properties, just like numeration systems (next chapters) and processes introduced in the first four chapters.

The logistic map is the most basic recurrence formula exhibiting various levels of chaos depending on its parameter. It has been used in population demographics to model chaotic behavior. Here we explore this model in the context of randomness simulation, and revisit a bizarre non-periodic random number generator discovered 70 years ago, based on the logistic map equation. We then discuss flaws and strengths in widely used random number generators, as well as how to reverse-engineer such algorithms. Finally, we discuss quantum algorithms, as they are appropriate in our context.

1. Logistic map

The logistic map is defined by the following recursion

$$X(k) = r\,X(k\text{-}1)\,(1 - X(k\text{-}1))$$

with one positive parameter r less or equal to 4. The starting value $X(0)$ is called the seed, and must be in [0, 1]. The higher the parameter r, the more chaotic the behavior. At $r = 3.56995...$ is the onset of chaos. At this point, from almost all seeds, we no longer see oscillations of finite period. In other words, slight variations in the initial population yield dramatically different results over time, a prime characteristic of chaos.

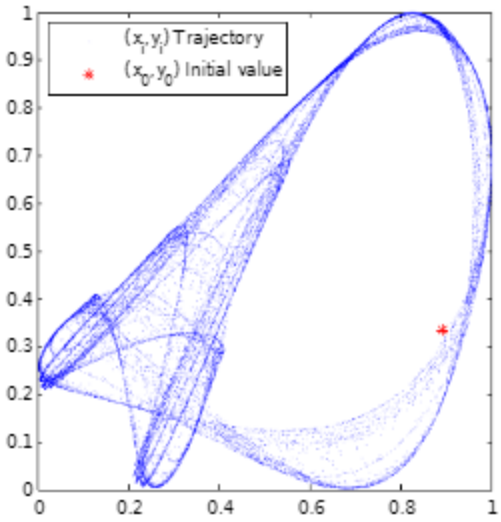
When $r = 4$, an exact solution is known, see here. In that case, the explicit formula is

$$X(k) = \sin^2\left(2^k \cdot \arcsin(X^{1/2}(0))\right)$$

The case $r = 4$ was used to build a random number generator decades ago. The k^{th} number $Z(k)$ produced by the random number generator in question, is equal to

$$Z(k) = \frac{2}{\pi} \cdot \arcsin(X^{1/2}(k))$$

The numbers $Z(k)$'s are uniformly distributed on [0, 1]. I checked whether they were correlated or not, and could not find any statistically significant auto-correlations. Interestingly, I initially found all this information in a military document published in 1992, still hosted here on the .mil domain, and of course, not classified (not sure if it was ever classified.) The original work is by S.M. Ulam and J. von Neumann, and was published in 1947. Very few seeds will result in periodicity -- an infinite number of them actually, but they are extremely rare, just like rational numbers among all real numbers.

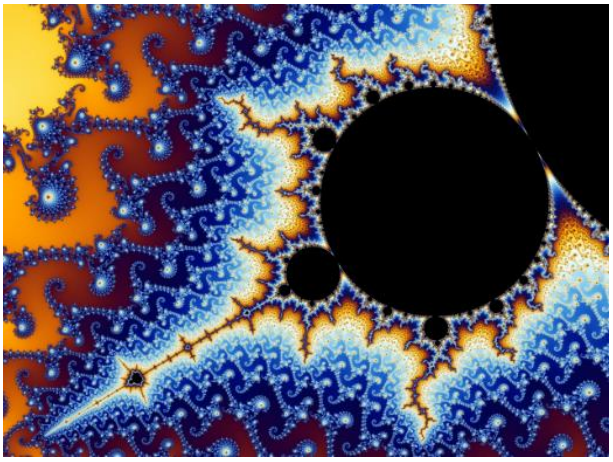


Source: 2-D logistic map, see here

The logistic map has been generalized, for instance in two dimensions. see here for an application to population dynamics. The 2-dimensional case has also been used in image encryption, see here. Also, in 2012, Liu Yan and Tong Xiao-Yung published a paper entitled *A new pseudorandom number generator based on complex number chaotic equation.* However, the simplest non-periodic good random number generator might just be defined in the complex plane (on the unit circle) by

$$X(k) = \sin(ab^k) + i\cos(ab^k) = \{X(k-1)\}^b, \text{ with } b > 1.$$

A recurrence formula can easily be obtained for the real and imaginary parts separately (both are real-valued random number generators) using complex exponentiation. Just like with the logistic map generator, a transformation must be applied to map the non-uniform deviate $X(k)$, to a uniform deviate $Z(k)$. The case $b = 2$ is similar to the logistic map generator with $r = 4$. Such home-made generators are free from NSA backdoor, in contrast for instance with the Dual Elliptic Curve Deterministic Random Bit Generator.



The Mandelbrot set is produced by a recursion similar to the logistic map, in the complex plane

2. Flaws in Popular Random Number Generators

Many applications require very good random number generators. For instance, to produce secure cryptographic systems, or when simulating or testing a large number of stochastic processes to check whether they fit with particular statistical distributions, as in chapter 3. In chapter 5, we also needed a random number generator capable of producing millions of distinct integer values, and that is how I discovered the flaws with the Perl rand() function still in use today.

This generator can only produce 32,768 distinct values, and when you multiply any generated value by 32,768, you obtain an integer value. These values are supposed to simulate uniform deviates on [0, 1]. Applications that are designed using this generator might not be secure. Indeed, the Perl documentation states that

"Rand() is not cryptographically secure. You should not rely on it in security-sensitive situations. As of this writing, a number of third-party CPAN modules offer random number generators intended by their authors to be cryptographically secure, including: Data::Entropy, Crypt::Random, Math::Random::Secure, and Math::TrulyRandom."

Chances are that similar issues can be found in many random number generators still in use today. I also tested the Excel rand() function, but could not replicate these issues. It looks like Microsoft fixed glitches found in its previous versions of Excel, as documented in this report. The Microsoft document in question provides details about the Excel random number generator: Essentially, it is equivalent to the sum of three linear congruential generators of periods respectively equal to a = 30,269, b = 30,307, and c = 30,323, so its overall period is equal to the product $N = abc$, at best. Note that a, b, and c are prime numbers. Also, in Excel, N * rand() should always be an integer, if you have this version of the algorithm built in your spreadsheet. This is hard to check, because we are beyond Excel's precision, limited to about 15 digits. To put it differently, Excel cannot produce a uniform deviate smaller than 1/N, other than zero, but again, this is difficult to check for the same reason.

To test these flaws and indeed reverse-engineer a random number generator producing deviates on [0, 1], one can proceed as follows.

- Generate a few deviates. Multiple the deviates by N, testing all N's between 1 and 1,000,000,000, to see if some N always result in an integer. Note that if you try with too large values of N, say $N = 10^{14}$, you will always get an integer, but in this case it is due to machine precision, not because of the random number generator.

- Generate millions of deviates. When you find one that is identical to the first deviate, check whether you've reached periodicity, with all subsequent values repeating the numbers found at the beginning. If that is the case, you've discovered the period of your random number generator. Note that the time series

of deviates might not be periodic immediately, and may enter into a periodic cycle only after a certain time, or it could display quasi-periodicity.

- Generate millions of deviates. Check how many distinct values you get. For instance, if you generate 10,000,000 deviates and find only 32,768 distinct values, your random number generator has a severe issue.

- Generate millions of deviates. At each iteration, update the minimum value found so far. Is this minimum approaching zero over time, or is it stuck somewhere, very close yet not close enough to zero?

Interestingly, unlike the random number generator produced by the logistic map and discovered in 1947 (described above in this article) or generators based on hardware or on the time function of your computer, many random number generators used today (for instance the one used in Java) are based on linear congruence and are thus periodic, and poor. But a few are excellent, and have such a large period that they are un-breakable for all practical purposes, such as the Mersenne twister, with a period of $2^{19937} - 1$. Note that $2^{19937} - 1$ is a Mersenne prime.

An advantage of the logistic map generator, over hardware-based generators, is that you can reproduce the same sequence of random numbers over and over in any programming language and on any computer -- a critical issue in scientific research, with many tests published in scientific journals not being re-producible. Other non-periodic, high quality random numbers offering the same reproducibility feature, include those based on irrational numbers. For instance, a basic recursion no more complicated than the logistic map formula, produces all the binary digits of SQRT(2)/2, one per iteration, and these digits are known to have a random behavior, according to standard tests. See also here, for an application about building a lottery that would not be illegal. For a fast random number generator based on the decimals of π, click here.

Note that the Perl random number generator has a period larger than 2^{23} according to my computations, despite producing only 32,768 distinct values. It could even be non-periodic, after all the binary digits of SQRT(2) produce only two distinct values - 0 and 1 - yet they are not periodic, otherwise it would represent a rational number.

Finally, the standard Diehard tests to assess the randomness of these generators should be updated. It was published in 1995 when big data was not widespread, and I am not sure whether these tests would be able to detect departure from randomness in generators used in sensitive applications requiring extremely large quantities of pseudo-random numbers.

3. Quantum algorithms

Quantum algorithms work well for applications that require performing a large number of repetitive, simple computations in which no shortcut seems available. The most famous one today is probably Shor's algorithm, to factor a product of integers. Traditional

encryption keys use a product of two very large prime numbers, large enough that no computer is able to factor them. Factoring a number is still considered an intractable problem, requiring to test a bunch of numbers as potential factors, without any obvious pattern that could accelerate the search. It is relevant to our context, as much of our concern here is about cryptographic applications and security. Also, the issue with these encryption keys is that they also should appear as random as possible - and products of large primes mimic randomness well enough. Maybe one day a classical (non-quantum) but efficient factoring algorithm will be found, but for now, quantum computing with Shor's algorithm seems promising, and can theoretically solve this problem very efficiently, thus jeopardizing the security of systems ranging from credit card payments to national security. This is why there is so much hype about this topic. Yet in 2014, the largest integer factored with quantum computers was only 56,153, a big increase over the record 143 achieved in 2012. But because of security concerns, new post-quantum cryptographic systems are being designed.

Another problem that benefits from quantum computers is counting the number of distinct values in a large set of integers. This is the element distinctness problem (click on the link to see the gain obtained with a quantum version.) In particular, in my 6 degrees of separation problem, I had to make sure that my random number generator was able to produce a very large number of distinct integers, each one representing a human being. Distinctness had to be tested. Again, this is an example of algorithm where you need to test many large blocks of numbers - independent from each other - without obvious shortcut to increase efficiency. Quantum algorithms can solve it far more efficiently than classical algorithms. As an alternative, the logistic map is known to produce infinite sequences of pseudo-random numbers that are all distinct - thus no test is needed. The related problem of detecting a period in long sequences that may or may not be periodic, and if periodic, have a very large period, can probably benefit as well from quantum architectures.

For further, reading, I suggest searching for Simulation of a Quantum Algorithm on a Classical Computer. Since quantum algorithms are more complex than classical ones, for quantum computing to become popular, one will have to develop a high-level programming or algorithmic language that can translate classic tasks into quantum code.

7. Chaos, Logistic Map, and Related Processes

We study processes related to the logistic map, including a special logistic map discussed here for the first time, with a simple equilibrium distribution. This chapter offers a transition between chapter 6, and the next chapters on numeration system (the logistic map being one of them.)

Here we describe well-known chaotic sequences, including new generalizations, with application to random number generation, highly non-linear auto-regressive models for times series, simulation, random permutations, and the use of big numbers (libraries available in programming languages to work with numbers with hundreds of decimals) as standard computer precision almost always produces completely erroneous results after a few iterations -- a fact rarely if ever mentioned in the scientific literature, but illustrated here, together with a solution. It is possible that all scientists who published on chaotic processes, used faulty numbers because of this issue.

This chapter is accessible to non-experts, even though we solve a special stochastic equation for the first time, providing an unexpected exact solution, for a new chaotic process that generalizes the logistic map. We also describe a general framework for continuous random number generators, and investigate the interesting auto-correlation structure associated with some of these sequences. References are provided, as well as fast source code to process big numbers accurately, and even an elegant mathematical proof in the last section.

While all the sequences investigated here are deterministic, we treat them as stochastic processes (after all, they are supposed to simulate chaos and randomness!) and we use probabilistic and statistical tools to handle them. To the best of my knowledge, this is the first time that continuous random number generators are described under a general unified framework, at least in the statistical literature. Similar methodology has been applied in several contexts pertaining to physics and mathematics. I hope that this chapter will jump-start some further research on this topic, and can be understood by a broad audience.

We solve here a stochastic integral equation problem, also with exact solution, similar to the one associated with self-correcting random walks (another kind of memory-less process) in chapter 3.

The approach used here starts with traditional data science and simulations for exploratory analysis, with empirical results confirmed later by mathematical arguments in the last section. For simplicity, some advanced topics required to make this framework completely rigorous, are not discussed here, for instance the stochastic fixed-point theorem for non-linear integral operators (the fixed point being a statistical

distribution, not a static value) including conditions of convergence and uniqueness of solutions.

1. General Framework

We are interested in the sequence of real numbers defined iteratively by

$$X(k) = g(X(k - 1)) \text{ for } k > 0,$$

with $s = X(0)$ in [0, 1] being called the *seed*, and g being a mapping (real-valued function, usually non invertible) from [0, 1] onto [0,1]. The successive values $X(k)$ can be seen as deviates from a distribution X, used for instance to simulate random numbers. While this concept of using a statistical distribution to model deterministic values is intuitive to non-statisticians, it is not for some statisticians. So let's discuss X in more details, as it is central to this article. For the statistician, the distribution of X is the limit of the empirical distribution computed on the $X(k)$'s, just as if the $X(k)$'s were observations of some experiment involving sampling. The distribution X may or may not exist (or can be singular) depending on the function g and on the seed.

Desirable Properties
Desirable properties include:

- Each $X(k)$ has the same distribution as X on [0, 1] regardless of the seed (except possibly for rare “bad seeds”.)
- $X(k)$ is always in [0, 1] and the successive values eventually fill the whole interval in some random order.

The $X(k)$'s may or may not be correlated. General conditions for convergence and uniqueness are beyond the scope of this article. Suffice it to say, for the initiated, that X is the fixed-point of a mathematical operator defined by g, and should not depend on the seed (but only on g) except possibly for a set S of “bad seeds” that yield periodicity or other non-random behavior, with the Lebesgue measure of this set S equal to 0. This is the case in all the examples discussed here.

The most simple case exhibiting the desired properties is the logistic map, defined by $g(X) = 4 X(1 - X)$. It has been investigated thoroughly, used in many applications, generalized, and exact formulas for $X(k)$ and the distribution of X are known: see chapter 6.

General Solution
In all cases, the distribution of X is solution of the following equation:

$$P(X < x) = P(g(X) < x).$$

While this sounds like a simple equation, it is actually a stochastic integral equation in disguise, and exact solutions are known only in very few cases. We will solve this

equation later in this article, for a special case that at first glance seems more complicated than the logistic map. What makes this equation challenging is the fact that the function *g*, in all practical examples, is non-invertible on [0, 1].

Usage

Besides solving this functional equation, we are interested here in transforming the *X*(*k*)'s to obtain uniform deviates on [0, 1], for instance to produce random numbers on [0, 1]. While the *X*(*k*)'s and *X* also have [0, 1] as support domain, generally the distribution is not uniform, and thus a mapping is necessary to "uniformize" them. When no exact solution is known, an approximate solution obtained via interpolation on the *X*(*k*)'s empirical percentiles, can be used. Indeed that's what we did in the case that we solved (see later in this chapter), and that's how we discovered that it had indeed an exact solution.

Another example of self-replicating process (fractals)

2. Examples of Chaotic Sequences, and Big Number Programming

Because of the nature of the recursion involved to compute *X*(*k*), rounding errors propagate very fast, to the point that after 40 or so iterations, all computed values of *X*(*k*) are completely wrong. It is possible that ***all scientists who published on chaotic processes, used faulty numbers*** because of this issue. In some cases, it matters, and in some, it does not. See the examples below for details. Usually it is not a real issue, because it is like re-starting the sequence with a new seed every 40 iterations or so. So the fact that the conclusions made by prior authors are sound despite using totally faulty values, is an artifact of the process itself, which yields the same distribution *X* regardless of the seed (a result of the ergodic nature of this process, see chapter 1.)

Interestingly, I computed the values for a generalized logistic map (referred to as LM 0.5 below) both with big numbers -- that is, with extreme precision -- and with traditional arithmetic including Excel, and noticed the issue, with both sequences being totally different after 50 iterations. I then computed and used the empirical percentiles from both sequences (big numbers and Excel versions) theoretically representing the same exact process, to fit them with known distributions. I discovered that $P(X < px - qx^2) = x$ with p close to 2, and q close 1 in both cases. However, the faulty sequence produced p and q even closer to 1 and 2, so close and with such a great fit, that I decided to check whether indeed, it was the exact solution. Had I been working only with the big numbers, I might not have found that this was indeed the exact solution, as proved in the last section.

It is possible that re-seeding the process periodically, which is what the faulty sequence does, improves the overall estimation for a number of statistics, to the point that it outperforms using correct values computed on a single seed. The source code using big numbers is provided later in this article.

Examples of Sequences

Here are a few examples. I tested them all, mostly with the seed $s = \pi/11$.

- **FRACT**: Uses $g(X) = mX - \text{INT}(mX)$ where m is the multiplier, and INT represents the integer part function. This is one of the most random, and thus best sequences, with all auto-correlations equal to 0, and distributed uniformly on [0, 1] without the need for any mapping. This is a consequence of the equidistribution theorem. Computations run very fast, even with big numbers. However, with traditional arithmetic, it gets stuck to 0 forever, after 10 iterations, if $m = 32$ This is not the case if $m = 31$: It then produces a faulty sequence, but one that appears random enough for practical purposes. If you avoid using a power of 2 for the multiplier m, and use a seed that is not too close to a simple rational number, you should be fine.

- **ABSIN**: Uses $g(X) = |\sin(mX)|$ where m is a multiplier as in FRACT. My main comment is that with big numbers, it runs very slowly. Also, it is not free from auto-correlations.

- **LM 1**: Uses $g(X) = 4X(1 - X)$. This is the classic logistic map, with all auto-correlations equal to 0. An exact solution is known, Also there are infinitely many "bad seeds" to avoid, but these seeds are known and extremely rare. It runs fast even with big numbers. More about LM 1 can be found in chapter 6.

- **LM 0.5**: Uses $g(X) = \text{SQRT}(4X(1 - X))$. It has a curious auto-correlation structure, with exponential decay and alternate signs. The first auto-correlation (lag-1) is equal to -1/2. So auto-correlations are quickly vanishing, having no impact on convergence. The sequence can be modified to remove the auto-correlations, using $Z(k) = \sin(256\, X(k))$ instead of $X(k)$. The implementation runs fast with big numbers, if you use the SQRT function rather than the power function to compute

square roots. Despite the appearance, LM 0.5 is totally different from LM 1, and no smooth mapping exists to map one sequence onto the other. An exact solution for the distribution of X is available, see next section.

Discrete, Continuous Sequences and Generalizations

Finally, it would be interesting to see if a similar framework could be applied to discrete random number generators. Pretty much all random number generators in use today are discrete ones, in the sense that each $X(k)$ generated is an integer. Most of them are periodic, though some have a very long period, see chapter 6. Continuous random number generators like those presented here are non-periodic, have nice properties, and might be easier to study. However, due to the finite amount of memory and data storage, for all practical purposes, because of machine precision, they become periodic when implemented, though the period can be immense.

All these sequences, whether made of integers or real numbers, are also stationary, in the sense that $X(k)$ depends only on $X(k\text{-}1)$ but not explicitly on k. A possible generalization consists of sequences of the form

$$X(k) = g(X(k\text{-}1), X(k\text{-}2))$$

with two seeds $s = X(0)$ and $t = X(1)$.

Also of interest is the sequence **LM *p*** defined by $g(X) = (4X\ (1 - X))^p$ where p is a positive parameter. The factor 4 that appears in the formula for g, guarantees that it maps [0, 1] onto [0, 1], as $X(1\text{-}X)$ has a maximum value equal to 1/4. The parameter p must be in a certain range for the sequence to exhibit the desired randomness properties; p =1 and p = 0.5 work, but p = 0.2 or p = 1.1 do not. Low values of p result in significant auto-correlations, which eventually, when p is low enough, start impacting convergence. High values of p result in the sequence quickly converging to 0.

3. The Case LM 0.5, Auto-Regressive Time Series, Source Code

We investigate the case LM 0.5 defined by $g(X) = \text{SQRT}(4X\ (1 - X))$, that is, $X(k) = \text{SQRT}(\ 4\ X(k\text{-}1)\ (1 - X(k\text{-}1))\ )$. As discussed earlier, we have empirically obtained a solution, and proved that it was an exact solution, see next section for the proof. This solution is $P(X < x) = 1 - \text{SQRT}(1 - x)$. As a result, $Z(k) = 1 - \text{SQRT}(1 - X(k))$ has a uniform distribution on [0, 1]. Whether there exists a simple exact formula for $X(k)$, remains an open question. Note that for the case LM 1, an exact formula exists for $X(k)$, see chapter 6.

Auto-correlation Structure

We have $E[X] = 2/3$, $\text{Var}[X] = 4/45$, $E[g(X)] = E[X]$, $\text{Var}[g(X)] = \text{Var}[X]$, and

$$\mathrm{E}[X \cdot g(X)] = \int_0^1 \frac{x \cdot g(x)}{2\sqrt{1-x}} dx = \int_0^1 x^{3/2} dx = \frac{2}{5}$$

Thus Covar[X, g(X)] = E[X g(X)] - 4/9 = -2/45, and the lag-1 autocorrelation is equal to Covar[X, $g(X)$] / Var[X] = -1/2.

A good approximation for the lag-n autocorrelation is $(-1/2)^n$. This might actually be the exact value, and it is indeed the case for $n = 1$ and $n = 2$. One might be able to find the exact value by deriving a recurrence formula to compute $R(n) = \mathrm{E}[\, X\, g(g(g(...g(X)...)))\,]$ where the symbol g appears n times inside the expectation operator E, and using the change of variable $y = g(x)$ in the integral. The lag-n autocorrelation is equal to $(R(n) - \mathrm{E}^2[X]) / \mathrm{Var}[X]$.

There are several ways to decorrelate time series (see chapter 15), but here we simply used $Z(k) = \sin(256\, X(k))$, as it fits our purpose of generating sequences with increased randomness (the opposite of what statisticians do in their daily life), and it works pretty well.

Auto-regressive, Non-linear Time Series

The LM 0.5 sequence can be used to simulate non-linear auto-regressive time series with an auto-correlation structure that is decaying exponentially fast. Indeed, it can be used as an alternate base model for auto-regressive processes. Each seed provides a different sequence, yet all sequences have the same distribution. Also, all sub-sequences of a given sequence have the same distribution. In short, re-seeding a sequence now and then, does not alter the distribution, as the sequence emulates a memory-less stochastic process or time series.

Related Sequences and Literature

I tried to find other articles about the LM 0.5 sequence, but could not find any. There is a lot of literature on generalized logistic maps, and chapter 6 includes references to 2-D logistic map and applications to image encryption and random number generators. See also the Wofram Encyclopedia entry on this subject. This is topic of active research.

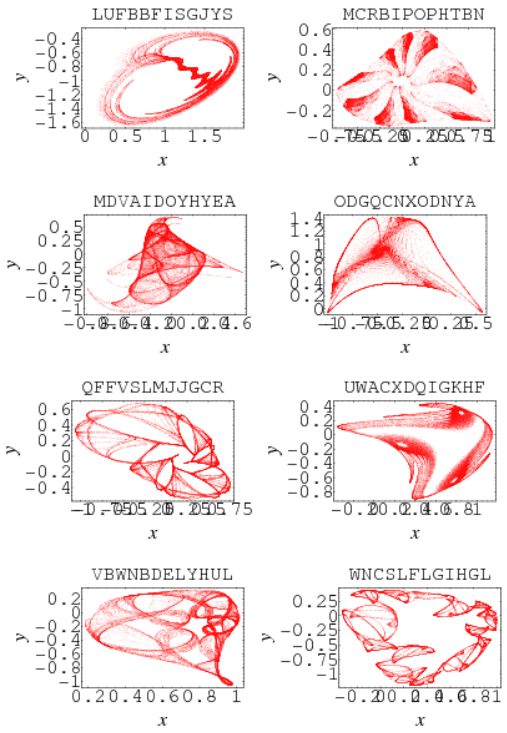
Other articles worth reading:

- Route to chaos in generalized logistic map (2017) using $g(X) = r\, X^p\, (1 - X^q)$. Authors: Rafał Rak, Ewa Rak. Accepted for publication in *Acta Physica Polonica A.*
- On some generalized discrete logistic maps (2012) using the same function g as in the previous reference. Published in *Journal of Advanced Research*. Author: Ahmed G. Radwan.
- A generalization of the logistic map equation (1999), mater's thesis, 57 pages. See also here. The focus is on identifying cases with an exact solution. Authors: Valerie Poulin (Carleton), Hugo Touchette (McGill.)
- A Generalization of the Logistic Map, and Its Applications In Generating Pseudo-random Numbers (2006, PDF document.) Authors: Mahesh Shastry et al. See

also similar paper by Samar M. Ismail et al, published in the proceedings of the IEEE 2015 International Conference on Science and Technology, here.

- Importance of Generalized Logistic Distribution in Extreme Value Modeling (2013, available here.) Published in *Scientific Research*. Authors: K. Nidhin, C. Chandran.
- Design of Positive, Negative, and Alternating Sign Generalized Logistic Maps (2015, PDF document), using g(X) = $r\,X\,(a + b\,X)$. Published in *Discrete Dynamics in Nature and Society*. Authors: Wafaa S. Sayed et al.
- Modified Logistic Maps for Cryptographic Application (2015.) Published in *Applied Mathematics*. Authors: Shahram Etemadi Borujeni, Mohammad Saeed Ehsani.
- Implementation of the Logistic Map with FPGA using 32 bits fixed point standard (2017, accessible here.) Authors: Diego A. Silva et al.
- Generalized Smooth Transition Map Between Tent and Logistic Maps (2017, click here -- it will cost you $35) published in *International Journal of Bifurcation and Chaos*. Authors: Wafaa S. Sayed et al.

Didier Gonze (2015) wrote an interesting tutorial on the subject, see here or here. It describes the same generalizations of the logistic map, the Lyapunov exponent which measures how chaotic the system is, the universal Feigenbaum chaos constant associated with these processes, and the case when dealing with time-continuous time series, that is, when k is a real number rather than an integer, resulting in a simple differential equation. The context is to model population dynamics. A more comprehensive reference on the subject is the textbook Chaotic Dynamical Systems (Robert L. Devaney, 2003) where the logistic map is referred to as the quadratic map. For a more recent textbook for beginners, read "Non Linear Dynamics and Chaos" (Steven H. Strogatz, 2015, 2nd Edition.)



Picture: Attractors for various quadratic maps. (Source: Wolfram)

None of these authors mention LM 0.5 nor its exact solution. The fact that all computations, if based on standard arithmetic, are erroneous, is not mentioned either. It is not clear to me which conclusions are wrong if any, in these studies, because of this issue.

Source Code with Big Numbers, and Model Fitting with Excel

The source code below (in Perl) uses big numbers for LM 0.5. It is interesting to compare the results with the faulty sequence obtained with traditional arithmetic. Results are in this spreadsheet, which also features the data science technique used for model fitting, based on the empirical distribution and percentiles analysis.

Big number arithmetic also produces faulty sequences, but after many more iterations. This is not an issue, as long as X does not depend on the seed. The loss of accuracy with standard computer arithmetic becomes dramatic after 50 iterations or so. Think about this: For the case LM 1 for instance, $X(k)$ is a polynomial (function of the seed s), of degree 2 at power 2^{k-1}. How do you expect to get a value correct with just one decimal, using standard arithmetic, for k as small as 50?

We encourage the reader to post a Python version (with big number arithmetic) in the comment section below. The Python libraries that I've found handle big integers, but not decimal numbers with hundreds of decimals. It might be possible to just use big integers, afterall the computations in the code below rely implicitly on big integers, via `$z=int(0.5+$bigint*$z)/$bigint`.

```
use strict;
use bignum;
my $pi=4*atan2(1,1);
my $bigint= 10**50;
my $seed=$pi/11;
my $k;
my $z=$seed;
my $zp;
open(OUT,">randgen.txt");
for ($k=1; $k<10000; $k++) {
  $z=4*$z*(1-$z);
  $z=sqrt($z);
# Note: $z=$z**0.5 is much slower than $z=sqrt($z)
  $z=int(0.5+$bigint*$z)/$bigint;
  $zp=sprintf("%.10f",$z); # get first 10 decimals
  print OUT "$k\t$zp\n";
  print "$k -- $zp\n";
  select()->flush();       # you can delete this instruction
}
close(OUT);
```

The variable `$bigint` in the code is used to set the precision, here to 50 decimals. Without it, the code is incredibly slow and becomes much slower at each iteration, producing values with more and more decimals.

4. Proof of Main Results

We prove the following results, for the LM 0.5 sequence, with x in [0, 1]:

- $P(X < x)$ = 1 - SQRT(1 - x)
- Let Y = 1 - SQRT(1 - X). Then Y has a uniform distribution on [0, 1].
- $P(X < 2x - x^2) = x$.

To prove the first result, we plug X's distribution in the stochastic functional equation $P(X< x) = P(g(X)< x)$, using the fact that g is a two-to-one convex function on [0, 1], and we check that the stochastic equation is satisfied. The second result follows immediately from the probability integral transform. The third result is left as an exercise. The proof of the first result proceeds as follows:

$$g(X) = \sqrt{4X(1-X)} < x \text{ if } X < g_1(x) \text{ or } X > g_2(x) \text{ with}$$

$$g_1(x) = \frac{1-\sqrt{1-x^2}}{2},$$

$$g_2(x) = \frac{1+\sqrt{1-x^2}}{2}.$$

Thus we have:

$$\begin{aligned} P(g(X) < x) &= P(X < g_1(x)) + P(X > g_2(x)) \\ &= P(X < g_1(x)) + 1 - P(X < g_2(x)) \\ &= 1 - \sqrt{1 - g_1(x)} + \sqrt{1 - g_2(x)} \\ &= 1 - \sqrt{\frac{1 + \sqrt{1 - x^2}}{2}} + \sqrt{\frac{1 - \sqrt{1 - x^2}}{2}} \end{aligned}$$

and the stochastic equation to verify becomes, after successive squaring and re-arrangements:

$$P(X < x) = P(g(X) < x) \Leftrightarrow 1 - \sqrt{1 - x} = 1 - \sqrt{\frac{1 + \sqrt{1 - x^2}}{2}} + \sqrt{\frac{1 - \sqrt{1 - x^2}}{2}}$$

$$\Leftrightarrow 1 - x = \frac{1 + \sqrt{1 - x^2}}{2} + \frac{1 - \sqrt{1 - x^2}}{2} - 2 \cdot \sqrt{\frac{1 + \sqrt{1 - x^2}}{2}} \cdot \sqrt{\frac{1 - \sqrt{1 - x^2}}{2}}$$

As most terms cancel out in the right-hand side of the last equality, it becomes 1 - *x* = 1 - *x*, which shows that the stochastic equation is satisfied, thus completing the proof. Note that there are automated tools for this type of tedious computations; they are usually referred to as symbolic math software. Some are available online.

8. Numerical and Computational Issues

These issues have been mentioned in chapter 7, and also appear in chapters 9, 10 and 11. Here we take a deeper dive and offer solutions, using high precision computing with BigNumber libraries.

In some applications, using the standard precision in your programming language of choice, may not be enough, and can lead to disastrous errors. In some cases, you work with a library that is supposed to provide very high precision, when in fact the library in question does not work as advertised. In some cases, lack of precision results in obvious problems that are easy to spot, and in some cases, everything seems to be working fine and you are not aware that your simulations are completely wrong after as little as 30 iterations. We explore this case in this article, using a simple example that can be used to test the precision of your tool and of your results.

Such problems arise frequently with algorithms that do not converge to a fixed solution, but instead generate numbers that oscillate continuously in some interval, converging in distribution rather than in value, unlike traditional algorithms that aim to optimize some function. The examples abound in chaotic theory, and the simplest case is the recursion $X(k + 1) = 4\, X(k)\, (1- X(k))$, starting with a seed $s = X(0)$ in [0, 1]. We will use this example - known as the logistic map - to benchmark various computing systems. Examples of algorithms that can be severely impacted by aggregated loss of precision, besides ill-conditioned problems, include:

- Markov Chain Monte Carlo (MCMC) simulations, a modern statistical method of estimation for complex problems and nested or hierarchical models, including Bayesian networks.
- Reflective stochastic processes, see chapter 3. This includes some types or Brownian or Wiener processes.
- Chaotic processes, see chapter 7 (especially section 2.) These include fractals.
- Continuous random number generators, see chapter 7.

The conclusions based on the faulty sequences generated are not necessarily invalid, as long as the focus is on the distribution being studied, rather than on the exact values from specific sequences. For random number generators, it may result in a much smaller period than expected, still acceptable for most applications except strong cryptography. An example of problem where high precision values are needed, is when comparing two sequences generated by two different chaotic processes, to check whether you can map one onto the other, with a smooth transformation. Another example is computing long-range autocorrelations associated with these sequences.

Besides, it is interesting to know how many decimals are correct in your computations, especially when using high precision libraries that fail to deliver on the promises.

1. Benchmarking

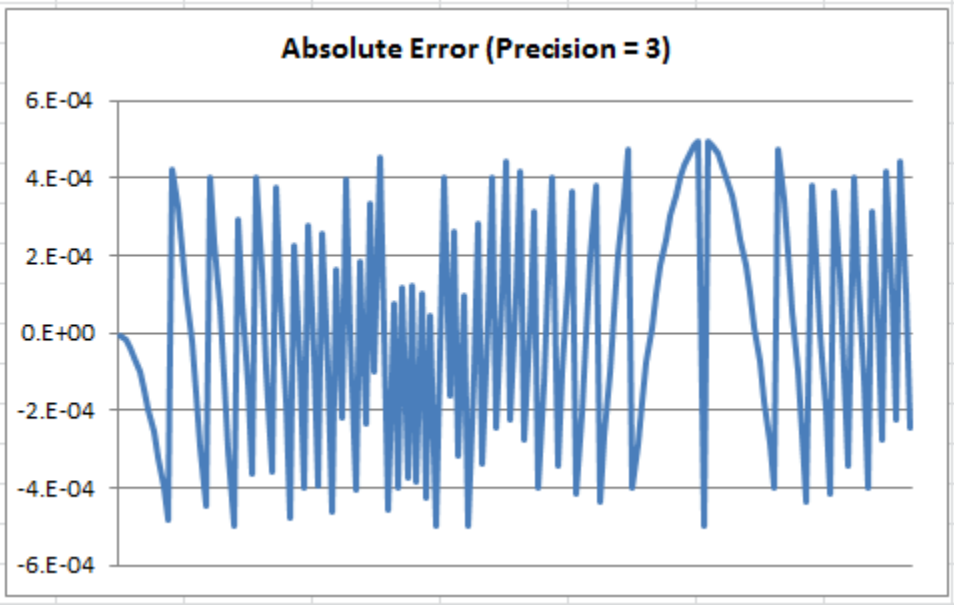
We use the sequence $X(k)$ defined in the introduction, with the seed $s = 3 / 10$, All the values generated lie in [0, 1]. We define *precision* as the metric used to determine how many decimals are correct. If n decimals are correct, we say that precision = n. We denote as $X(k, n)$ the value of $X(k)$ rounded to provide a precision equal to n. We simulate low precision using rounded high precision values, to show the impact or cumulative errors on the generated values, as k increases, as well as to detect when (for which value of n) the high precision libraries become untrustworthy. Note that in most systems, including Excel, the standard precision is approximately $n = 14$, which is not enough here.

The table below shows the first 138 iterations (some rows truncated to make it easier to read), computed with various precision levels. The full table can be downloaded here (Excel) and contains all the pictures included in this article, with detailed computation.

	Precision									
Iteration	5	10	15	20	25	30	35	40	45	50
0	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
1	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84
10	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
11	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17
29	0.31	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61	0.61
30	0.86	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
47	0.95	0.00	0.04	0.05	0.05	0.05	0.05	0.05	0.05	0.05
48	0.19	0.00	0.16	0.17	0.17	0.17	0.17	0.17	0.17	0.17
66	0.55	0.98	0.41	0.01	0.01	0.01	0.01	0.01	0.01	0.01
67	0.99	0.06	0.97	0.02	0.03	0.03	0.03	0.03	0.03	0.03
86	0.06	0.14	0.95	0.98	0.61	0.61	0.61	0.61	0.61	0.61
87	0.23	0.49	0.20	0.08	0.95	0.95	0.95	0.95	0.95	0.95
98	0.80	0.63	0.04	0.99	0.47	0.90	0.90	0.90	0.90	0.90
99	0.63	0.93	0.16	0.02	1.00	0.35	0.36	0.36	0.36	0.36
117	0.94	0.08	0.59	0.28	0.12	1.00	0.04	0.04	0.04	0.04
118	0.24	0.30	0.96	0.80	0.41	0.01	0.15	0.14	0.14	0.14
136	0.05	0.97	0.90	0.54	0.34	0.06	0.13	0.69	0.69	0.69
137	0.17	0.11	0.35	0.99	0.89	0.22	0.46	0.85	0.86	0.86
138	0.57	0.40	0.91	0.02	0.38	0.70	0.99	0.50	0.48	0.48

As you can see, the number of iterations providing a rather accurate value of $X(k)$ increases linearly as a function of the precision n. This linear relationship is easy to understand. Every now and then, almost periodically, we hit a value $x = X(k)$ such that the error computed on $g(x) = 4x(1 - x)$ with n decimals, reaches a peak, as shown in the

picture below. Such values of *x* are easy to identify, and result in an increased deterioration of accuracy each time with get too close to them. And because the sequence *X*(*k*) oscillates without ever converging, the peak errors, however small, keep showing up unabated every now and then, rather than decreasing to 0 as in standard algorithms. Note that the distribution of *X*(*k*) can still be computed accurately despite the massive cumulative error, but not the exact values after as little as 40 iterations.

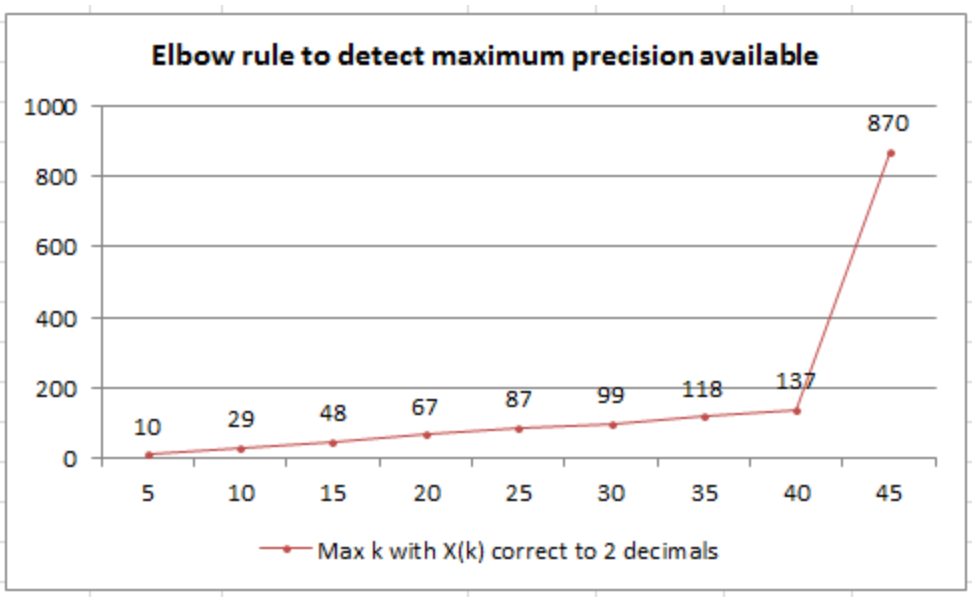


Absolute (non-cumulative) error for the first 170 iterations (n = 3)

The optimum precision to work with is function of how many correct successive values you need, but also based on the practical limitations of your high precision tool, and on how fast the algorithm needs to run. The higher the precision, the slower the computations. If you generate a large number of values, this becomes an issue.

2. Testing How Reliable your High Precision Tool is

This linear relationship between the number of iterations generating a rather accurate value (say correct to two decimals) and the precision *n,* can be used to test how good your high precision library is, and when it fails. In this sense, the sequence studied here is good for benchmarking purposes.



Interpretation: With n =20, only the first 67 iterations are correct up to two decimals; -- Somewhere between n = 40 and n = 45, the gain in precision becomes an illusion

Note that for Excel (n = 14) and for most programming languages using standard arithmetic, the Y-value in the above picture would have been close to 47.

In the above picture, somewhere between n = 40 and n = 45, the gain in precision becomes an illusion. It is impossible that suddenly, we get so many iterates of $X(k)$ correct up to 2 decimals. I actually computed 100,000 iterations with n as high as 100, without any subsequent improvement. Clearly, my BigNumbers library (in Perl) is not working as advertised beyond n = 45, or maybe some functions used in my code, possibly INT, have a lower limit on precision. I have read similar stories for the Decimals library in Python. Getting results slightly better than mine is probably easy, but computing a correct value with two decimals for k = 1,000,000 is likely a very challenging problem, even though a simple direct formula exists for $X(k)$, see chapter 6. Such values could be used to produce secure cryptographic systems, though they might be breakable using quantum algorithms or high performance computing (HPC.)

Application to Cryptography

The RSA encryption algorithm relies on factoring a product of two very large primes. If you only know the product, it is computationally impossible to factor it to retrieve the two primes to decrypt the message. If you know one of the two primes, factoring is easy. These primes are associated with the public and private keys.

Instead of using the product of two primes, one could use the sum of two numbers $X(k)$ and $X(k+m)$ computed with 100 decimals, for large k and m. If you know the sum, it is computationally impossible to retrieve the two terms. If you know $X(k)$, you can easily retrieve $X(k+m)$. Better, use transformed $X(k)$'s that have a uniform distribution on [0, 1].

A simple transformation exists to "uniformize" the sequence, making your system even stronger, see chapter 6. You can also use two different seeds for $X(k)$ and $X(k+m)$. Note that $X(k)$ and $X(k+m)$ are not correlated, even if using the same seed. High precision is critical here.

Source Code

Source code in Perl, Python and Java, is available here. The Perl version is available in the previous article, in the last part of section 3.

9. Digits of Pi, Randomness, and Stochastic Processes

Deep mathematical and data science research (including a result about the randomness of π, which is just a particular case) are presented here, without using arcane terminology or complicated equations. Numeration systems discussed here are a particular case of deterministic sequences behaving just like the stochastic process investigated earlier, in particular the logistic map, which is a particular case.

The topic discussed here, under a unified framework, is at the intersection of mathematics, probability theory, chaotic systems, stochastic processes, data and computer science. Many exotic objects are investigated, such as an unusual version of the logistic map, nested square roots, and representation of a number in a fractional or irrational base system.

This chapter is also useful to anyone interested in learning these topics, whether they have any interest in the randomness or π or not, because of the numerous potential applications. I hope the style is refreshing, and I believe that you will find plenty of material rarely if ever discussed in textbooks or in the classroom. The requirements to understand this material are minimal, as I went to great lengths (over a period of years) to make it accessible to a large audience. The content (including truly original exercises and applications) is also valuable to college professors looking for off-the-beaten-path material to teach, as well as to students wishing to get exposure and interest to advanced topics usually reserved for 'elite' data scientists, without being discouraged due to the simplicity of the presentation.

The randomness of the digits of π is one of the most fascinating, unsolved mathematical problems of all times, having been investigated by many millions of people over several hundred years. The scope of this article encompasses this particular problem as part of a far more general framework. More questions are asked than answered, making this document a stepping stone for future research.

For a quick summary of the material presented in this long chapter, see chapter 10.

1. General Framework

We are dealing with sequences $x(n)$ of positive real numbers defined by a recurrence relationship $x(n+1) = g(n)$ for some function g, and initiated with a **seed** $x(1) = x$. We are interested only in functions g that produce chaos and preserve the domain. For instance, if $x(n)$ is anywhere in [0, 1] we want $x(n+1)$ to also be anywhere in [0.1].

We also study another sequence derived from $x(n)$, and defined by $a(n) = h(x(n))$ for some function h. The function h is typically very simple and produces only small integer

values such as (say) 0, 1,..., 9 regardless of $x(n)$. The sequence $a(n)$ is called the **digits** of x represented in the chaotic system defined by the function g.

Using appropriate functions for g, the sequence $x(n)$ behaves as a realization of a stochastic (random) process, each term $x(n)$ acting as a random deviate of some random variable $X(n)$. To be of any value here, the function g must be associated with random variables that are identically distributed, though not necessarily independent. Instead of using the notation $X(1)$, $X(2)$ and so on, we simply write X as all these random variables have the same distribution. The distribution of X is called the **equilibrium distribution** (or fixed-point distribution) of the system, and it is the solution of a simple stochastic integral equation. Read chapter 7 for some illustrations, especially section 3. Likewise, the $a(n)$ sequence has a statistical distribution associated with it, called the **digits distribution**, and easily derived from the equilibrium distribution.

Examples of functions g producing these patterns are investigated in details in this chapter. Such systems are usually referred to as chaotic systems. It is a special case of (ergodic) stochastic process or time series. Note that in practice, for a given chaos-inducing function g, not all seeds -- the initial value $x(1)$ -- produce a sequence $x(n)$ with the desired properties. Seeds failing to do so are called **bad seeds** and result in equilibrium and digits distributions that are degenerate. Even though all the systems discussed here have infinitely many bad seeds, these seeds are so rare in practice (compared with the good seeds) that the probability to stumble upon one by chance for a given chaos-inducing g, is zero.

At this point, the initiated reader probably knows what I am going to talk about, even though much of it is original research published for the first time. In the grand scheme of things, the randomness of π (and I will discuss it later in this article) appears as a very peculiar case, certainly not the most noteworthy result to prove, though still very fascinating.

Questions, Properties and Notations about Chaotic Sequences Investigated Here

For the remaining of this article, we are going to focus on the following, applied to a few different types of chaos-inducing functions g:

- The digits $a(n)$ are denoted as $[a(1), a(2), a(3)...]$ or $\{a(n)\}$ and uniquely represent the number x. We want to build sequences $\{a(n)\}$ based on the functions h and g, such that if x and x' are two distinct numbers, then the respective representations $\{a(n)\}$ and $\{a'(n)\}$ are different.
- There is a function f, usually straightforward (but not always) such that for a given chaos-inducing g and a given h, we have $x = f(a(1), a(2), a(3)...)$.
- For a given g and h, can the algorithm producing $\{a(n)\}$ generate any arbitrary $\{a(n)\}$ or only some specific types of $\{a(n)\}$ with some constraints on the digits? Sometimes yes (example: decimal representations), sometimes no (example: continuous fractions).
- Are the sequences $\{x(n)\}$ and $\{a(n)\}$ auto-correlated? Sometimes yes, sometimes no. What are the implications? For the representation of a number in any integer

base b such as $b = 10$, $\{x(n)\}$ is auto-correlated while $\{a(n)\}$ is not. If the base b is not an integer, $\{x(n)\}$ is much less auto-correlated, but $\{a(n)\}$ becomes auto-correlated. Also if the base b is not an integer, the digits distribution is no longer uniform.

- Can we compute the equilibrium and digits distributions? Yes using an iterative algorithm, but in all the examples discussed in the next section, exact solutions in closed form are known and featured, and I will show you how they were obtained.

Potential Applications

The sequences $\{x(n)\}$ and $\{a(n)\}$ can be used for continuous / discrete number generation and in cryptographic applications. See also chapter 6 with applications to modeling population demographics, and physics. They can also be used for machine precision benchmarking, as numerical errors accumulate so fast that after 50 iterations, regardless of the sequence, all digits are wrong unless you use high precision computing (see chapter 8.)

2. Examples of Chaotic Sequences Representing Numbers

All the g functions presented here are chaos-inducing. This is necessary if one wants, with a given g, to represent all numbers, not just a small fraction of them. For each case, we present, whenever possible, the following features:

- The function f discussed earlier, to reconstruct the original number x (the seed)
- The (continuous) equilibrium distribution and its support domain
- The (discrete) digits distribution
- The functions g and h
- Other properties such as auto-correlations or exact formula for $x(n)$ when available

Three types of chaotic processes are discussed

- Tradition representation of a number in base b, with $b > 1$ (integer or not) and denoted as Base(b)
- Logistic map of exponent p, denoted as LM(p)
- Nested powers of exponent q, denoted as NP(q)

In all cases, the seed $x(1) = x$ is a positive real number. The equilibrium distribution (if it exists) is always solution of the stochastic integral equation

$$P(X < y) = P(g(X) < y).$$

A way to solve this apparently complex equation is described in details, and in very simple words, ib chapter 7, with an illustration for the logistic map LM(1/2) discussed below. It involves two steps:

- **Data Science Step**: Gather data to compute the empirical percentile distribution for $x(n)$, and perform various regressions (polynomial, logarithmic and so on) until you discover one with an incredibly fantastic fit with data -- like a perfect fit.

- **Mathematical Step**: Plug that tentative distribution into the stochastic integral equation, and see if it solves the equation.

Also, by definition, as discussed earlier, $x(n+1) = g(x(n))$ and $a(n) = h(x(n))$. This can be used to design a trivial algorithm to compute all the digits in any system powered by a chaos-inducing function g. The seed x can be reconstructed using the formula $x = f(\{a(n)\})$. Here the notation INT represents the integer part, sometimes called floor function. In the formula displayed as images, the floor function is represented by left and right brackets: this is the standard convention.

I now discuss three types of number representation system.

Numbers in Base 2, 10, 3/2 or π

This system (the traditional decimal system in base b) is characterized by:

$$g(x) = bx - \lfloor bx \rfloor$$

$$h(x) = \lfloor bx \rfloor$$

$$x = f(\{a(n)\}) = \frac{a(1)}{b} + \frac{a(2)}{b^2} + \frac{a(3)}{b^3} + \cdots$$

with $b > 1$. The seed x must be in [0,1] so that all $x(n)$'s are also in [0.1]. If $x = \pi$, use x-3 instead.

We distinguish two cases:

- **If *b* is an integer**: The equilibrium distribution is uniform on [0,1] and the digits distribution is uniform on {0, 1, ... , b} for all but very rare bad seeds x such as rational numbers. This fact has been “known” for millennia (at least for $b = 10$) and it is taken for granted; it can be proved rigorously by solving the stochastic integral equation $P(X < y) = P(g(X) < y)$ attached to this system. However, in general, such systems do not produce a uniform distribution for the digits: this is an exception to the examples discussed here. The auto-correlation between $x(n+1)$ and $x(n)$ is equal to $1/b$, while the auto-correlation between $a(n+1)$ and $a(n)$ is equal to 0. So the h function actually decorrelates the sequence $\{x(n)\}$, to use the vocabulary of time series theory.

- **If *b* is not an integer**: The equilibrium distribution is NOT uniform on [0,1]. The auto-correlation between $x(n+1)$ and $x(n)$ is not 0, but much closer to 0 than in the above case. This time, the auto-correlation between $a(n+1)$ and $a(n)$ is NOT equal to 0. The digits take values in {0, 1, ... , INT(b)}. I did not test if the digits distribution was uniform on that domain, but my gut feelings tells me that this is not the case.

Computation details for a customizable base *b*, are available in this spreadsheet. Rather than computing auto-correlations based on a single seed (that is, one instance of the process) using large values of *n*, I computed them using a large number of seeds (all random numbers) computing the auto-correlations for a small, fixed *n* but across many seeds. Due to the nature of the process (it is ergodic) both computations yield the same conclusions. The reason for using many seeds and a small *n* is because large *n* (above 20) lead to total loss of numerical accuracy.

Nested Square Roots

This is a particular case of a far more general model described in section 3 in the following article. It is characterized by

$$g(x) = x^2 - \lfloor x^2 \rfloor + 1$$

$$h(x) = \lfloor x^2 \rfloor - 1$$

$$x = f(\{a_n\}) = \sqrt{a(1) + \sqrt{a(2) + \sqrt{a(3) + \cdots}}}$$

This system is related to continued fractions, except that *a*(*n*) can only be equal to 0, 1, or 2, as opposed to taking any arbitrary positive integer value. The seed *x* must be in [1, 2] so that all *x*(*n*)'s are also in [1, 2]. If $x = \pi$, use *x*-2 instead.

The equilibrium distribution is given by the following formula:

$$P(X < y) = -2 + \sqrt{5y - 1}, \text{ with } 1 \le y \le 2.$$

The digits distribution is given by the following formula:

$$P(a(n) = 0) = \sqrt{5\sqrt{2} - 1} - \sqrt{5\sqrt{1} - 1} = 0.4639536951544...$$

$$P(a(n) = 1) = \sqrt{5\sqrt{3} - 1} - \sqrt{5\sqrt{2} - 1} = 0.3037626988867...$$

$$P(a(n) = 2) = \sqrt{5\sqrt{4} - 1} - \sqrt{5\sqrt{3} - 1} = 0.2322836059588...$$

Again, these results are easily derived from the stochastic integral equation. Note that the sequence {*x*(*n*)} exhibits strong auto-correlations in this case, though auto-correlations for {*a*(*n*)} are close to (but different from) 0.

This system has an infinite number of bad seeds, like the other systems discussed here. These are numbers *x* that do not have the prescribed digits distribution. Their set has measure 0, so they are very rare. An example is (1 + SQRT(5)) / 2, with all the digits being equal to 1. Another one is SQRT(2), with *a*(1) = 0 and all other digits being equal to 2. Any number like SQRT(2) that at some point in its digit representation, only has 2's (the highest possible value for a digit in this system), causes the same problem as numbers in the decimal system that at some point, only have 9's (the highest possible value for a digit in the decimal system of base 10).

Finally, for the nested square root system, the formula $x = f(\{a(n)\})$ to reconstruct the seed x using its digits, must be computed backwards. The opposite is true for the decimal system. For more on nested square roots, click here and read entry #118 (the one that is most relevant to our problem.)

Logistic Map LM(*p*)

The logistic map has many applications, for instance in social sciences. Here we are interested in the logistic map of exponent p, with $p = 1$ corresponding to the standard, well known version. We also discuss the case $p = 1/2$, as these are two cases (possibly the only two cases) that provide a *chaotic-inducing g* covering all real numbers (one that meets the criteria outlined in section 1) with known equilibrium distribution that can be expressed with a simple closed form formula. The function h associated with this system was chosen arbitrarily, but this is also the simplest function h that one could come up with. I don't know yet if there is a simple formula for the function f.
This system is characterized by

$$g(x) = \Big(4x(1-x)\Big)^p$$
$$h(x) = \lfloor 2x \rfloor$$
$$x = f(\{a_n\}) = ?$$

The seed x must be in [0,1] so that all $x(n)$'s are also in [0.1]. If $x = \pi$, use x-3 or x/4 instead. A remarkable fact, for the case $p = 1$, is that the auto-correlations for $\{x(n)\}$ are all equal to 0, making it, in some way, more random than the decimal system or other systems discussed here. For the case $p = 1/2$ the auto-correlation between $x(n+1)$ and $x(n)$ is equal to -1/2, it is equal to +1/4 between $x(n+2)$ and $x(n)$, and to -1/8 between $x(n+3)$ and $x(n)$, see proof in section 3 in chapter 7. By contrast, it is equal to 1/2 for the decimal system in base 2.

The equilibrium distributions, defined for y in [0, 1], are equal to:

$$P(X < y) = \frac{1}{\pi}\int_0^y \frac{1}{\sqrt{x(1-x)}}dx \text{ if } p = 1,$$
$$P(X < y) = 1 - \sqrt{1-y} \text{ if } p = 1/2.$$

As usual, this applies to good seeds only (the vast majority of seeds). A proof of the formula for $p = 1/2$ can be found in chapter 7. The integral for the case $p = 1$ can be explicitly and automatically computed using Wolfram Alpha, see here (I think the formula returned by this API is wrong though, as it involves an expression in one square root and its opposite in another square root: one of them has to be negative.)

For the case $p = 1$, the digits distribution is uniform on {0, 1}. The probability for a digit to be 0 is equal to $P(X < 1/2) = 0.5$, based on this computation. This is not true for the logistic map with $p = 1/2$.

3. About the Randomness of the Digits of π

The discussion here is not just about π, but about any number. The digits {a(*n*)} of a number *x*, in a specific system defined by *g* and *h*, are said to be *random*, if they have the proper digits distribution, as well as the proper auto-correlation structure associated with that system. In short, any finite set of digits of *x* must have the proper joint distribution that all but a tiny subset of numbers (of Lebesgue measure zero) has in that system. The target digits distribution can be computed from the equilibrium distribution, as discussed in the previous section. Numbers satisfying this criterion are called *good seeds.*

For instance, in the decimal system of base *b*, if *b* is an integer, the distribution in question is uniform, auto-correlation between successive digits are zero, and the digits are independently distributed with that same uniform distribution. If the base *b* is not an integer (say *b* = 3/2), the theoretical auto-correlation between *a*(*n*+1) and *a*(*n*) is clearly not zero, the digits distribution is clearly not uniform, and both can be computed exactly using the same mathematical arguments as in section 3 of chapter 7.

In the literature, the randomness of the digits is defined as normalcy. For the traditional decimal system, the definitions (random digits, good seed, normal number) are similar though good seed is stronger in the sense that it takes the absence of auto-correlation into account. Anyway, for π or SQRT(2), even proving that the digit 5 appears infinitely many times -- a very weak result indeed -- would be a phenomenal discovery.

The Digits of π are Random in the Logistic Map System

We show here why π is a good seed in the standard logistic map system, with its digits behaving exactly like those of pretty much all numbers in that system. The result is based on the fact that for the standard logistic map (corresponding to *p* = 1 in the previous section), an exact formula is available for {*x*(*n*)} and thus for {*a*(*n*)} as well, see chapter 6. It is given by

$$x(n) = \sin^2(2^{n-1}\pi w(x)), \text{ with } w(x) = \frac{1}{\pi}\arcsin\sqrt{x}$$

$$a(n) = \lfloor 2x(n) \rfloor$$

It implies that bad seeds (all bad seeds?) are numbers *x* that make *w*(*x*) a rational number, as these seeds don't follow the equilibrium distribution; π/4 is not one of them, so π/4 must be a good seed.

While this result about the randomness of the digits of π may seem spectacular at first glance, there are still big hurdles to overcome to formally prove it. In particular:

1. Are the digits *a*(*n*) properly defined in the logistic map system? No function *x* = *f*({*a*(*n*)}) has been found yet, unlike in the decimal or nested square root systems. Can two different numbers have the same digits?
2. Are the only bad seeds those that are very easy to spot in the exact formula for *x*(*n*)? If difficult to prove, we may be back to square one.

3. Probably the biggest hurdle is to prove that $w(\pi/4)$ is irrational.

Note that for the logistic map system, the digits distribution (for good seeds, that is, for pretty much all numbers) is uniform on {0, 1} like for the decimal system of base 2. This was proved at the bottom of section 2 in this chapter. Also, the $a(n)$'s are independent since the $x(n)$'s are. Thus the digits system produced by the logistic map seems sound. Interestingly, except for the first digit $a(1)$, the numbers x and $1 - x$ always share the same digits representation. Finally, since x must be in [0, 1], here we use $\pi/4$ (say), rather than π.

Paths to Proving Randomness in the Decimal System

The problem with the decimal system (in any base) is the opposite of the logistic map system. The function $x = f(\{a(n)\})$ is known, but there is no useful formula to exploit for for $a(n)$. This makes things more complicated. Note however that there is a way to directly compute the n^{th} digit of π, $a(n)$, in base 16, using the Bailey–Borwein–Plouffe formula.

To prove that π is a good seed in the decimal system, we could try one of the following approaches:

- Find an approximation for $x(n)$ or maybe an infinite series involving terms similar to $w(x)$, making it easier to spot the bad seeds -- all of them. Today, the only bad seeds known are rational numbers and artificially manufactured numbers such as $x = 0.101101110...$
- Study the problem in a base b that is not an integer. Try a sequence of bases that converge to (say) $b = 2$ or $b = 10$. Or try sequences of bad seeds (rational numbers) that converge to π, and look at the convergence (or lack of) of the distributions attached to these bad seeds, to a uniform distribution on [0, 1]. See exercise 9 in chapter 16. Or better, consider a combination of sequence of bases converging to base 10 (say) and a sequence of bad seeds converging to π, simultaneously.
- Use a transformation of $\{x(n)\}$ that more easily leads to a solution. In particular, find a status-preserving mapping v such that for any x, if $v(x)$ is a good seed, then x is also a good seed. Maybe it could be easier to prove that $v(\pi)$ -- rather than π -- is a good seed.
- Find a mapping between the logistic map and the decimal system, for $x(n)$. However, such mappings may be very complicated if they exist at all. For base 2, an interesting result (with mapping to the logistic map) is available: See exercise 7 in chapter 16.
- The $x(n)$'s are auto-correlated in the decimal system, while they are not in the logistic map system. Could this be the source of the difficulties? Maybe finding a one-to-one mapping that decorrelates the $x(n)$'s could help. Note that in both systems, the digits $a(n)$'s are not auto-correlated.
- The function g in the decimal system of base b is not continuous: $g(x)$ is the fractional part of bx. To the contrary, it is continuous in the logistic map system. Yet, the fractional part function has a simple Fourier series (with terms consisting

of sine and cosine functions) that could possibly help. Its inverse also has a simple Fourier series, see here. In short, for the decimal system of base *b*, we have:

$$g(x) = bx - \lfloor bx \rfloor = \frac{1}{2} - \frac{1}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2\pi k b x)}{k}$$

These are all very challenging problems.

Connection with Brownian Motions

The focus here is to try to further characterize bad seeds, to help answer questions about the status of π or other mathematical constants. We raise a few extra suggestions to solve this problem.

The process {*x*(*n*)} is chaotic. In number theory, working with the cumulative process, in this case *y*(*n*) = *x*(*n*) + *x*(*n*-1) + ... + *x*(1), which is much smoother, sometimes makes things easier. We expect that *z*(*n*) = (*y*(*n*) - *n* E(*X*)) / SQRT(*n*) has a normal (Gaussian) distribution if *x* is a good seed (see chapter 12), since the *x*(*n*)'s all have the same distribution, and are barely correlated if *x* is a good seed, depending on *g*. The process {*z*(*n*)}, when properly scaled, is then a Brownian motion (introduced in chapter 1.) Here E(*X*) represents the expectation of the equilibrium distribution; in particular, it is equal to 0.5 for the decimal system of base *b* (*b* integer) and for the logistic map. It is also easy to get its exact value for the other systems discussed in this article.

But what if *x* is a bad seed? For the decimal system (of any base, even non-integer ones) maybe the Fourier series for *g*(*x*), provided in the previous sub-section, could help with the computation of the distribution of *z*(*n*). Yet an irrational number such as *x* = 0.223388000099.. in base 10 (with duplicated decimals) might still provide a Gaussian distribution, despite being a bad seed. What happens if *x* is rational? Or if *x* only contains 0's and 1's? These are also bad seeds in the decimal system of base 10. What about applying the same methodology to the *a*(*n*)'s directly, which are known not to be correlated for the decimal system, if *x* is a good seed and the base is an integer?

4. Curious Facts

Here we discuss additional interesting topics related to the representation of numbers in various systems.

Randomness and The Bad Seeds Paradox

Bad seeds (or non-normal numbers) for the decimal system at least, are so rare (though there are infinitely many of them, including all rational numbers) that their set has a Lebesgue measure equal to zero. In other words, if you pick up a number at random, the chance that it is a bad seed is zero. Contrarily, the chance that all its digits are uniformly and independently distributed -- the feature of a good seed -- is 100 per cent! Yet there are as many bad seeds as good ones. Any number with duplicated digits, such as 0.339955666677..is a bad seed (its digits are highly auto-correlated). And there is a one-to-one mapping (bijection) between these numbers and all real numbers in [0, 1]. So maybe, the Lebesgue measure is not a useful metric in this context.

Application to Cryptography, Financial Markets, Blockchain, and HPC
If there is no known, fully reversible mapping between the decimal system in base 2 and the digits produced by the logistic map system -- and since there is no known $f(\{a(n)\})$ available to reconstruct a seed x in the logistic map system -- one can build an encryption system as follows:

The code to be encoded is the seed x. It must be a good seed for the logistic map. The immense majority of numbers are, including simple numbers such as 3/10.

- The secret code is the digits of x in base 2.
- The public code is its sequence of digits in the logistic map system, easy to compute with high precision computing. But it is impossible, one would think, to reconstruct the secret code if you only know its digits in the logistic map system, assuming it has a enough digits.

This can be used for authentication, by checking whether the logistic map code entered by some users to sign-up on some platform, is actually a valid one. It can also be used in Blockchain / bitcoin mining, to make sure that the bits of data sent by blockchain or by bitcoin miners, are legit. Currently, hash functions are used instead, but the logistic map can play the same role. In short, sending data and getting it validated by incorporating a signature into it, is made extremely computer-intensive for security reasons, whether -- as of today -- the signature is a very peculiar kind of hash with many zeroes at the beginning (and very hard to produce by chance), or whether it consists of digits that match one of numerous pre-specified valid seeds or some digit patterns in some of the number representation systems discussed here.

Another application is about transmitting secret messages. Your secret message is the seed x, encoded in some base b in the decimal system. What you transmit publicly, is the same message but encoded in a different base, say b'. As long as one of the two bases is an irrational number, you are safe. Reconstructing the original message is very easy if you know the bases used in the encryption scheme. So the bases b and b' could be the encryption keys -- possibly one of them being public, the other one private.
Yet another application is in finance. The process $y(n) = x(1) + ... + x(n)$ simulates a generalized random walk (see chapter 3), and can be used (when scaled properly) as a Brownian motion to model stock price fluctuations, when using good seeds, and also with some special types or bad seeds.

Finally, the algorithm used here to compute the digits of π or any other number, in any system, is not practical. On most machines, $x(n)$ becomes totally wrong in as little as 30 iterations due to round-off errors accumulating exponentially fast. This is discussed in chapter 8. Nevertheless, it would be interesting to see how far you can go, with high precision computing (HPC, see chapter 8), until the digits being computed are no longer correct. Indeed, the computation of $a(n)$ for large n and good seeds, can be used to benchmark various libraries (in Python, Perl and so on) to check how well and how fast they perform, to produce correct values. For π, more than a trillion digits are known, and we can use this number as a seed, in these benchmarking tests.

In this article, the focus was never on a single number like π, but on all numbers. The processes investigated here being all ergodic, it is easier to identify general statistical properties by focusing on a large collection of numbers, using small values of *n* and digits correctly computed, rather than on tons of digits computed for an handful of numbers, with pretty much all digits incorrectly computed due to round-off errors with standard computations.

Digits of π in Base π

Finally, just for fun, the first 31 digits of π in base π are (after skipping the integer part equal to 3):

π - 3 = [0, 1, 1, 0, 2, 1, 1, 1, 0, 0, 2, 0, 2, 2, 1, 1, 3, 0, 0, 0, 1, 0, 2, 0, 0, 0, 2, 0, 3, 0, 1, ...]

They provide the approximation (when re-written in base 10) π = 3.14159265358979. This is just a standard sequence satisfying the equilibrium distribution of its system, with no obvious patterns. In base π, regardless of the seed *x* (as long as it is a good seed, and pretty much all of them are), all digits are either equal to 0, 1, 2, or 3, the digits distribution is not uniform, and we have auto-correlations. So it looks like π is normal (that is, a good seed) in base π, as its digits seem to follow the (strange) prescribed distribution that applies to all good seeds in that base.

Interestingly, in base *b* (*b* being an integer or not), *x* = 1 / (*b* -1) is always a bad seed, as all its digits *a*(*n*) are all equal to 1. This is also true for base *b* = π. However if *b* is irrational, *x* = *b* (in this case *x*= π) seems to be a good seed, despite 1 / (*b* - 1) being a bad one. For comparison purposes, here are the digits for a few other seeds, in base *b* = π:

- 3/7 = [1, 1, 0, 0, 2, 2, 0, 3, 0, 0, 2, 1, 1, 1, 0, 0, 0, 3, 0, 0, 0, 0, 2, 0, 1, 2, 0, 1, 0, 2, 2, ...]
- 1 / (π -1) = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...]
- SQRT(2) - 1 = [1, 0, 2, 3, 0, 0, 1, 2, 1, 2, 1, 2, 0, 2, 2, 2, 2, 1, 1, 0, 1, 1, 2, 1, 0, 1, 2, 0, 2, 2, 0, ...]

For instance, in layman's terms, here is how the expansion of SQRT(2) looks like in base Pi:

$$\sqrt{2}-1=\frac{1}{\pi}+\frac{0}{\pi^2}+\frac{2}{\pi^3}+\frac{3}{\pi^4}+\frac{0}{\pi^5}+\frac{0}{\pi^6}+\frac{1}{\pi^7}+\frac{2}{\pi^8}+\frac{1}{\pi^9}+\frac{2}{\pi^{10}}+\frac{1}{\pi^{11}}+\frac{2}{\pi^{12}}+\cdots$$

It is a good exercise for statisticians (see exercise 3 in chapter 16) to check, with a statistical test of hypotheses, whether the digits distribution for π and SQRT(2), are identical in base π. Also, while 1 / (π -1) is a very bad seed in base π, it is (most likely) a good seed in base 2 or 10, or pretty much in any other base (even non-integer ones.)

10. Numeration Systems in One Picture

Here you will find a summary of much of the material previously covered on chaotic systems, in the context of numeration systems (in particular, chapters 7 and 9.)

Back to the basics. Here we are dealing with the oldest data set, created billions of years ago -- the set of integers -- and mostly the set consisting of two numbers: 0 and 1. All of us have learned how to write numbers even before attending primary school. Yet, it is attached to the most challenging unsolved mathematical problems of all times, such as the distribution of the digits of Pi in the decimal system. The table below reflects this contrast, being a blend of rudimentary and deep results. It is a reference for statisticians, number theorists, data scientists, and computer scientists, with a focus on probabilistic results. You will not find it in any textbook. Some of the systems described here (logistic map of exponent $p = 1/2$, nested square roots, auto-correlations in continued fractions) are research results published here for the first time.

This material is described in chapter 9, including how to derive all the results, and the equivalence between the base-2 system and the standard logistic map (with $p = 1$) mentioned in exercise 7 (last chapter.) It includes applications to cryptography, random number generation, high performance computing (HPC), population growth modeling, financial markets, BlockChain, and more.

1. Summary Tables

The following two tables summarize much of the material covered in the last few chapters.

Many other systems are not described in these tables, including:

- The iterated exponential map, possibly one of the most mysterious chaotic systems. See exercise 4 (last chapter) for details.
- The nested cubic root, a generalization of the nested square root.
- The generalized continued fraction of power p, especially $p = 2$, defined by

$$g(x) = 1/x^2 - \lfloor 1/x^2 \rfloor$$

The equilibrium distribution is the one satisfying $P(X < y) = P(g(X) < y)$. The statistical properties listed in the table, are valid for almost all seeds x, except for a set of seeds of measure 0 (depending on the system), known as bad seeds.

	Logistic Map ($p = 1$)	Logistic Map ($p = 0.5$)	Nested Square Root
Support domain for $x = x(1)$, $x(n)$ and $g(x)$	[0, 1]	[0, 1]	[1, 2]
Support domain for digits $a(n)$	{0, 1}	{0, 1}	{0, 1, 2}
$g(x)$ [$x(n+1) = g(x(n))$]	$4x(1-x)$	$\sqrt{4x(1-x)}$	$x^2 - \lfloor x^2 \rfloor + 1$
$h(x)$ [$a(n) = h(x(n))$]	$\lfloor 2x \rfloor$	$\lfloor 2x \rfloor$	$\lfloor x^2 \rfloor - 1$
$x = f(\{a(n)\})$	Unknown	Unknown	$\sqrt{a(1) + \sqrt{a(2) + \cdots}}$
Digits distribution P($a(n) = k$)	Uniform on {0, 1}	$\sqrt{2}/2$ if $k = 1$	$r(k+2) - r(k+1)$ $r(k) = \sqrt{5\sqrt{k} - 1}$
Equilibrium distribution P($x(n) < y$)	$\frac{1}{\pi} \int_0^y \frac{1}{\sqrt{x(1-x)}} dx$	$1 - \sqrt{1-y}$	$-2 + \sqrt{5y - 1}$
Correlation between $x(n+1)$ and $x(n)$	0	-1/2	Non zero
Correlation between $a(n+1)$ and $a(n)$	0	-1/4	Non zero (but close)

	Base b ($b > 1$, integer)	Base b ($b > 1$, not an integer)	Continued Fraction
Support domain for $x = x(1)$, $x(n)$ and $g(x)$	[0, 1]	[0, 1]	[0, 1]
Support domain for digits $a(n)$	$\{0, 1, \cdots, b-1\}$	$\{0, 1, \cdots, \lfloor b \rfloor\}$	1, 2, 3, ...
$g(x)$ [$x(n+1) = g(x(n))$]	$bx - \lfloor bx \rfloor$	$bx - \lfloor bx \rfloor$	$1/x - \lfloor 1/x \rfloor$
$h(x)$ [$a(n) = h(x(n))$]	$\lfloor bx \rfloor$	$\lfloor bx \rfloor$	$\lfloor 1/x \rfloor$
$x = f(\{a(n)\})$	$\sum_{k=1}^{\infty} \frac{a(k)}{b^k}$	$\sum_{k=1}^{\infty} \frac{a(k)}{b^k}$	$\cfrac{1}{a(1) + \cfrac{1}{a(2)+\cdots}}$
Digits distribution $P(a(n) = k)$	Uniform on {0, 1, ... , b - 1)	Non uniform	$\log_2 \left(\frac{(k+1)^2}{k(k+2)} \right)$
Equilibrium distribution $P(x(n) < y)$	Uniform on [0, 1]	Non uniform	$\frac{\log(1+y)}{\log 2}$
Correlation between $x(n+1)$ and $x(n)$	1/b	Non zero	Non zero
Correlation between $a(n+1)$ and $a(n)$	0	Non zero	E[$a(n)$] is infinite

2. Other interesting facts

For the standard logistic map (p = 1) a closed-form formula is available for $x(n)$:

$$x(n) = \sin^2(2^{n-1}\pi w(x)), \text{ with } w(x) = \frac{1}{\pi}\arcsin\sqrt{x}$$

Also, if $x(n)$ is the base-2 sequence for x, then $\sin^2(\pi x(n))$ is the logistic map sequence for $\sin^2(\pi x)$. See exercise 7 in chapter 16 for details.

In base b, if b is an integer, the exact formulas are as follows:

$$x(n) = b^{n-1}x - \lfloor b^{n-1}x \rfloor,$$
$$a(n) = \lfloor b^n x \rfloor - b\lfloor b^{n-1}x \rfloor.$$

Note that $x(1) = x$, since x is between 0 and 1.

Thus, proving that x is a good seed in base b (b integer) amounts to proving that the sequence { $b^n x$ } is equidistributed modulo 1 (see also here for a curious fact about $x = \pi$.) The case x = SQRT(2) in base-2 is studied here. It was proved that for this particular

case, the frequencies of 0's and 1's in the base-2 representation, are both equal to 50%. See here. I am working on a more simple proof, using the simple recursion described here. A special formula for the digits of Pi in base-16 is also available, see here. See also this link.

Not all number representation systems are as well-behaved as those describe here. For examples of more complex systems, see here. Also, even in well-behaved systems, some numbers (a subset of bad seeds) do not have any kind of statistical distribution for their digits. An example in base-2 is as follows: the first digit is 1, the next 2 digits are 0, the next 4 digits are 1, the next 8 digits are 0, the next 16 digits are 1, the next 32 digits are 0, and so on.

3. Reverse-engineering number representation systems

Some systems, like the Logistic Map 1, or base-*b* where *b* is an integer, have an exact formula to compute *x*(*n*) for any *n* without having to compute *x*(*n*-1), *x*(*n*-2) and so on. Is it possible, in these systems, given a particular *n* and *x*(*n*), to retrieve the seed *x* (and thus all its digits) by solving the equation in question?

We tried in base-2 for *x* = SQRT(2) / 2 and *n* = 10, and the answer is positive (at least for a good seed *x*, which is the case here.) In this case, for a given *n*, one has to solve the equation

$$x(n) = b^{n-1}x - \lfloor b^{n-1}x \rfloor,$$

where *x* is the unknown, *b* = 2, and both *n* and *x*(*n*) are known. A dichotomic search algorithm provides a solution very efficiently, since we know that *x* is in [0, 1]. However, the solution is not unique. This could be an issue for NSA and other organizations developing strong encryption systems. Or, to the contrary, it could help them in their reverse-engineering activities. Or better, it could help them explore systems with a natural, built-in backdoor, without being accused of manufacturing man-made, mandatory, and artificial backdoor, in existing encryption schemes.

Imagine an encryption system where *x* is the message, and *x*(*n*) is the encrypted message, using (say) the Logistic map 1 with *n* = 500. No one is ever going to figure out that you can actually retrieve *x* from *x*(*n*). Though in the case of the logistic map, 2^n seeds *x* lead to the same *x*(*n*), as g(*x*) = *g*(1-*x*). Even computing *x*(*n*) itself is not easy, requiring industrial-strength high precision computing (see chapter 8.)
Retrieving *x* from *x*(*n*) is even more difficult, despite the simple equation providing the solution: It is all about industrial strength HPC. But the fear of reversibility (successful attacks on cryptographic keys) has led to the development of quantum algorithms (see chapter 6) and quantum computers.

11. Chaotic Systems: More Statistical Tests and Applications

In addition to featuring new research results and building on the previous chapters, the topics discussed here offer a great sandbox for data scientists and mathematicians.

We illustrate pattern recognition techniques applied to an interesting mathematical problem: The representation of a number in non-conventional systems, generalizing the familiar base-2 or base-10 systems. The emphasis is on data science rather than mathematical theory, and the style is that of a tutorial, requiring minimum knowledge in mathematics or statistics. However, some off-the-beaten-path, state-of-the-art number theory research is discussed here, in a way that is accessible to college students after a first course in statistics. This chapter is also peppered with mathematical and statistical oddities, for instance the fact that there are units of information smaller than the bit.

You will also learn how the discovery process works, as I have included research that I thought would lead me to interesting results, but did not. In all scientific research, only final, successful results are presented, while actually most of the research leads to dead-ends, and is not made available to the reader. Here is your chance to discover these hidden steps, and my thought process!

The topic discussed is one of active research with applications to Blockchain or strong encryption. It is of interest to agencies such as the NSA or private research labs working on security issues. This is which it is not easy to find many references; some are probably classified documents. As far as I know, it is not part of any university curriculum either. Indeed, the fear of reversibility (successful attacks on cryptographic keys using modern computer networks, new reverse-engineering algorithms, and distributed architecture) has led to the development of quantum algorithms (see chapter 6) and quantum computers, as well as Blockchain (see chapter 9.)

All the results in this chapter were obtained without writing a single line of code, and replicable as I share my Excel spreadsheets. See also applications to Fintech in the last section.

1. General Framework

All standard number representation systems have a few universal components and properties. Here, for simplicity, we assume that the number x to be represented (in base b, b integer or not, or in other systems such as the logistic map) is in [0, 1]. The number x is referred to as a *seed*. The details are found in the previous chapter.

Components of number representation systems
The three basic components are:

- An iterative algorithm $x(n+1) = g(x(n))$ defined by a function g that maps [0, 1] onto [0, 1]. The starting point is $x(1) = x$ (the number to be represented, or seed, assumed to be in [0, 1] in most systems.)
- Digits of x, denoted as $a(n)$ for the n^{th} digit (starting with n =1) defined by a simple function h, as follows: $a(n) = h(x(n))$.
- An equilibrium distribution, solution of $P(X < y) = P(g(X) < y)$. This is a stochastic integral equation in disguise, and the exact solutions are known for all standard systems. It is used to find the typical statistical distribution of digits in a particular system, applicable to most x's in that system. The observed data for the underlying random variable X consists of the points $x(1)$, $x(2)$, and so on, essentially regardless of the seed. The equilibrium distribution of a system applies to almost all x's, except for very few x's known as *bad seeds* for the system in question.

General properties of these systems

Some of the general properties include:

- All systems have bad seeds, but the set of bad seeds, even though infinite, is extremely small: Its Lebesgue measure is equal to 0. Even most bad seeds have a statistical distribution for the digits (most of the time different from the equilibrium distribution, see digits of the bad seed x = 1/3 in base 10), though some bad seeds do not have a statistical distribution at all, for the digits (see chapter 10.)
- In some cases, an exact formula is always available for $x(n)$ and $a(n)$, regardless of n, without having to compute previous values, including for the base-2 or base-10 systems, and for the logistic map system.
- In some cases, an exact formula is available to compute x, based on its digits only. This is the case for the base-2 and base-10 systems, but not for the logistic map system.
- Two different seeds have two different representations (sets of digits.)
- A small increase or decrease in x results in a small increase or decrease in $x(n)$, regardless of n.
- You can retrieve x if you only know n and $x(n)$ for a specific n (any n.) It may not always be easy from a computational point of view, but theoretically feasible.
- In most systems, the equilibrium distribution is not a uniform distribution, and the successive digits are auto-correlated. Base-2, base-10, or any integer base, are among the few exceptions. Non uniform systems can be rescaled (by transforming g) to make them uniform.
- Identifying the bad seeds is extremely hard. If it was easy, we would know by now whether the digits of π in base 10, have a uniform distribution or not.
- For a given system, if all the digits, across all seeds, cover all potential sequences $\{ a(n) \}$ without any gap -- which is the case for all standard systems -- then obviously the system in question always has an infinite number of bad seeds. In any system, the number x with digits equal to 0 if n is odd, and equal to 1 if n is even, is always a bad seed. Same with any number that has only a finite number of digits equal to 0. In short, good seeds (the immense majority of seeds) are numbers with digits that look like they are randomly distributed, though the distribution is not necessarily uniform. In some binary systems, even for good

seeds, there are more 0's than 1's in the digit sequence, and autocorrelation (patterns) are obvious even to the naked eye. Yet the distribution is that of a true (and usually, simple, well known) statistical distribution -- just like correlated time series can still be modeled with statistical distributions, yet are random nevertheless. See above table for examples.

Examples of number representation systems

We describe the specifics of two systems that are further investigated in the next section, and were featured in chapter 10. As usual, *n* starts with *n* = 1, and *x*(1) = *x*. We assume that *x* is in [0, 1]. In particular, *a*(*n*) represents the *n*-th digit.

The base-*b* system (with *b* an integer here) is characterized by:

$$g(x) = bx - \lfloor bx \rfloor,\ x(n) = b^{n-1}x - \lfloor b^{n-1}x \rfloor$$

$$a(n) = \lfloor b^n x \rfloor - b\lfloor b^{n-1}x \rfloor,\ x = \sum_{k=1}^{\infty} \frac{a(k)}{b^k}$$

The Logistic Map system is characterized by:

$$g(x) = 4x(1-x),\ x(n) = \sin^2(2^{n-1}\pi w(x)), \text{with } w(x) = \frac{1}{\pi}\arcsin\sqrt{x}$$

$$a(n) = \lfloor 2x(n) \rfloor.$$

The brackets in these formulas represent the integer part function.

Examples of patterns in digits distribution

Below are the digits of *x* = SQRT(2) - 1 in base *b* = π. This number *x* is supposed to be a good seed in that system, yet its digits exhibit strong patterns. However, all good seeds exhibit the same exact patterns in that system. It means that the equilibrium distribution is not uniform, and that in general, the digits are auto-correlated (with the same auto-correlation structure regardless of *x*, if *x* is a good seed.) Indeed, lack of auto-correlation, or uniform distribution of digits, for a given seed *x*, would mean that *x* is a bad seed, in that system.

$$\sqrt{2}-1 = \frac{1}{\pi} + \frac{0}{\pi^2} + \frac{2}{\pi^3} + \frac{3}{\pi^4} + \frac{0}{\pi^5} + \frac{0}{\pi^6} + \frac{1}{\pi^7} + \frac{2}{\pi^8} + \frac{1}{\pi^9} + \frac{2}{\pi^{10}} + \frac{1}{\pi^{11}} + \frac{2}{\pi^{12}} + \cdots$$

To the contrary, in base 2, the artificial number whose n^{th} digit is equal to 1 if $n\pi$ - INT($n\pi$) is less than 1/3, and equal to 0 otherwise, has a known digit distribution; the digits are not auto-correlated in that system; there is no periodicity in the digits, yet it is a bad seed. There is no pattern in the digits except that the digit 0, on average, occurs twice as frequently as the digit 1. Actually, this is true if you replace π by any good seed in base 2. So there are as many bad seeds as good seeds, but the set of bad seeds has measure 0, while the set of good seeds has measure 1. More about this paradox is discussed here.

Purpose of this research

The purpose is to create a new number representation system, one where the digits of π or 0 or any number, would all have a uniform distribution, and appear random. One limitation of such a system is that some configuration of digits, those with a non-uniform

distribution, cannot exist by definition. Thus such a system must have infinitely many “holes”, unlike the other systems discussed so far, that have no hole. We will explore this in section 3 and 4.

But first, let's start with revisiting an old system, and introducing some applied pattern analysis techniques. This is the topic of the next section.

2. Defects found in the Logistic Map system

For a long time, I thought that the logistic map system was the best one, with uniform digit distribution, and no auto-correlation between successive digits. It has some advantages in cryptographic applications, over base-2 or base-10: It is almost impossible to reconstruct the seed x if you only know its digits, or one value of $x(n)$ for some large n; see chapter 10 however to reconstruct x if you know $x(n)$ for a rather small n. While all this is true, after further investigations, I've found some subtle departures from pure randomness in the logistic map. I mention it here for two reasons:

- If you design a strong encryption system based on the logistic map, you need to be aware of this problem, as it will make your system a little bit weaker.
- This is our first opportunity in this article to discuss pattern recognition techniques based on actual data (I share my spreadsheet at the bottom of this section.)

It is also the first time that this research result is published.

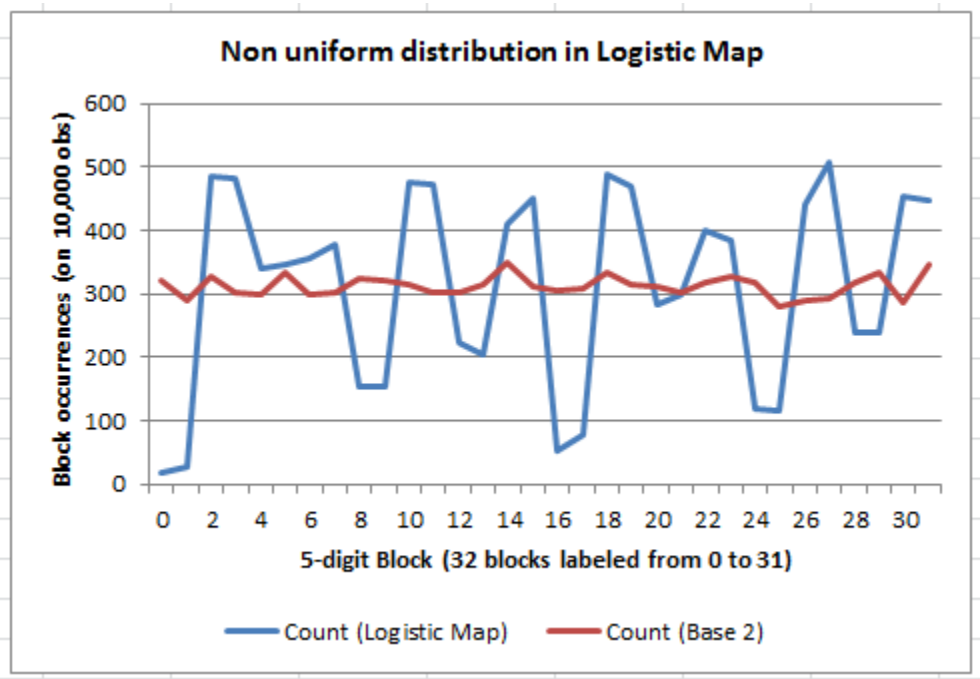
The Logistic Map system is defined in the previous section. I analyzed the distribution of blocks of digits (5 consecutive digits) and found that, contrarily to my expectations, the distribution was not uniform: See picture below. I tested 10,000 numbers (seeds) and compared the results with the base-2 system. In the picture below,

- The leftmost block on the X-axis corresponds to all 5 digits equal to 0, and it is clearly substantially under-represented for the logistic map (but not for base-2)
- The rightmost block corresponds to all 5 digits equal to 1.

Since there are 32 possible blocks of 5 binary digits, and 10,000 seeds in the test, on average each block should occur about 312.5 times. The block distribution, whether uniform (base-2) or not (Logistic Map), does not depend on the seed, as long as you pick up good seeds (pretty much all numbers are good seeds.) By increasing the number of seeds from 10,000 to (say) 20,000, 50,000 or 100,000, you could build confidence intervals confirming that the uniform distribution hypothesis does not hold for the logistic map.

It is surprising that despite the uniform distribution for single digits, and the lack of auto-correlation, we end up with a non-uniform distribution for blocks of digits. Another possible test consists of considering the sequence of digits { $a(n)$ } as a Markov chain, and computing the observed (or theoretical!) transition probabilities of moving from one state -- say $a(n) = 0$ -- to another one -- say $a(n+1) = 1$. Yet another test consists of analyzing the gaps between successive identical digits, or successive identical blocks of digits. If the distribution is uniform, these gaps have a geometric distribution: see

chapter 4 for how this test works, on a practical example, in a very similar context: SQRT(2) in base 10.



You can download the spreadsheet with detailed computations, here

3. First step in designing a new system

This is a step that usually no scientist would publish anywhere (it would be rejected in scientific journals), as it led me to a dead-end, with no practical value in the business context that I am interested in. However, I publish it here for the following reasons:

- It leads to more pattern recognition techniques that I will discuss in this section: It is useful for educational purposes.
- It features one of the wildest mathematical creatures that I have seen (in this case, discovered) in a long time.
- It was my intermediate step to get to a better system (discussed in the next section), and thus it illustrates the thought process (rarely discussed in the literature or in classrooms) involved in scientific research.

The purpose, as mentioned in the introduction, is to create a number representation system that has no bad seed. All standard systems have plenty of bad seeds. No one knows if π is one of those bad seeds, in any number representation system, be it base-2 or base-10. It is widely believed that it is not. In my new system described here, all numbers, including π (or zero for that matter) have digits that appear as if they were randomly generated, following some known statistical distribution.

This is part of a general research interest: Trying to prove that numbers such as π or SQRT(2) have digits that look random in standard systems, and trying to design better

systems for Blockchain or cryptographic applications. By creating this new system, I can say that I have found one in which the digits of Pi have a random distribution compatible with the equilibrium distribution of the system in question, thus debunking the myth that it is impossible to prove or disprove, regardless of the system, that digits of popular mathematical constants are randomly distributed. However this new system is not suitable for business applications, as we will soon see. The real value is in systems studied in section 4.

First version of new number representation system

The new system is defined as follows, and it involves a parameter *c* (just like the base-*b* system involves a parameter *b*) which in this case must be a positive irrational number:

$$g(x) = x + c - \lfloor x + c \rfloor,\ a(n) = \lfloor 2x(n) \rfloor,\ x \in [0,1],$$
$$x(n) = (n-1)c + x - \lfloor (n-1)c + x \rfloor,\ n = 1,2,3,\cdots.$$

Thus this system also has an exact formula for $x(n)$, just like the base-2 system. Also all digits of any seed x, are binary, equal to either 0 or 1, and $x(1) = x$. The equilibrium distribution is uniform on [0, 1] regardless of x or c (c irrational). You don't need to solve the stochastic integral equation attached to that system to figure this out, it is a consequence of the equidistribution theorem if c is irrational. An immediate consequence is that all seeds are good seeds. The behavior of the system depends on c (for the auto-correlation structure) but not on x. So you might as well study it using only $x = 0$. For instance, if $c = \log(2)$, the first few digits of $x = 0$ are as follows:

0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0

All numbers (regardless of c) have 50% of digits equal to 0, and 50% equal to 1. As discussed earlier, this system must have holes. For instance, no number x, not even zero, regardless of c (assuming c is irrational) has periodicity in its digit distribution, unlike all other systems. It is a sound system? Can two different numbers have the same digits? It seems that it is a sound system, though I still have some doubts.

However the most spectacular feature of this system is incredibly strong autocorrelations both in the { $x(n)$ } and { $a(n)$ } sequences, no matter which values of c or x you test. For a specific c, autocorrelations do not depend on x (so you might as well look at x =0 alone.) This is a consequence of having an equilibrium distribution. Also, there are incredibly strong conditions on { $a(n)$ } to make it a valid digit sequence, and thus, you could say that holes are everywhere, very obvious, and indeed occupy most of the space just like empty space in the universe.

If you want to check that { $x(n)$ } has a uniform distribution, using data science techniques, you can compute empirical percentiles for $x(n)$ for the first 10,000 values, maybe using $x = 0$, and find that they perfectly match those of a uniform distribution. This is illustrated in section 4 as well as in chapter 3, 7 and 11, and we will do it just using Excel!

Holes, autocorrelations, and entropy (information theory)
The new number representation systems introduced is this section, is full of holes, and actually mostly consists of holes, with respect to the distribution of the digits, as previously stated. Here we explore this feature in more details. One striking way to look at it is as follows.

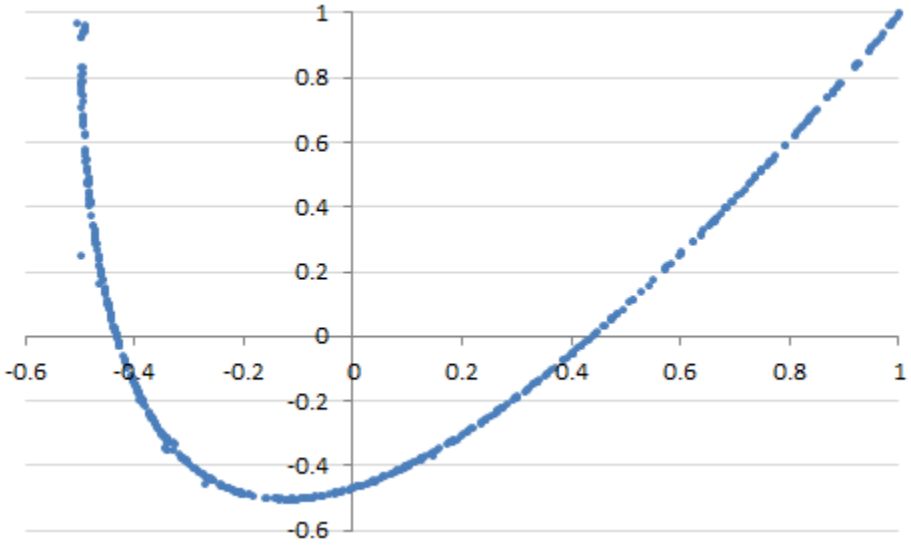
I computed the first 13 digits of more than 10,000 random numbers, for c = log(6) / log(10). I only found 26 = 2 x 13 possible (valid) configurations, listed below:

```
0,0,0,1,1,0,0,1,1,0,0,0,1
0,0,0,1,1,0,0,1,1,1,0,0,1
0,0,1,1,0,0,0,1,1,0,0,1,1
0,0,1,1,0,0,1,1,0,0,0,1,1
0,0,1,1,0,0,1,1,1,0,0,1,1
0,0,1,1,1,0,0,1,1,0,0,0,1
0,0,1,1,1,0,0,1,1,0,0,1,1
0,1,1,0,0,0,1,1,0,0,1,1,0
0,1,1,0,0,0,1,1,0,0,1,1,1
0,1,1,0,0,1,1,0,0,0,1,1,0
0,1,1,0,0,1,1,1,0,0,1,1,0
0,1,1,1,0,0,1,1,0,0,0,1,1
0,1,1,1,0,0,1,1,0,0,1,1,1
1,0,0,0,1,1,0,0,1,1,0,0,0
1,0,0,0,1,1,0,0,1,1,1,0,0
1,0,0,1,1,0,0,0,1,1,0,0,1
1,0,0,1,1,0,0,1,1,1,0,0,1
1,0,0,1,1,1,0,0,1,1,0,0,0
1,0,0,1,1,1,0,0,1,1,0,0,1
1,1,0,0,0,1,1,0,0,1,1,0,0
1,1,0,0,0,1,1,0,0,1,1,1,0
1,1,0,0,1,1,0,0,0,1,1,0,0
1,1,0,0,1,1,0,0,1,1,1,0,0
1,1,0,0,1,1,1,0,0,1,1,0,0
1,1,1,0,0,1,1,0,0,0,1,1,0
1,1,1,0,0,1,1,0,0,1,1,1,0
```

As an exercise, you can try with different values of c, or with more than 13 digits. If instead you do the same exercise in base 2, you would find that all 2^{13} = 8,192 digit configurations are represented. This brings us to the concept of information. How much information does one digit provide about the number x? In base b = 1,000, one digit obviously provides three times as much information as in base b = 10. In base b = 2, one digit provides little information (just one bit), compared to base b = 10. In base b = 1.3 (yes, the base does not need to be an integer, see chapter 10) a digit provides even less information. In my new system, a digit provides even far less information. Indeed, we have this surprising fact: **One digit in my system carries much less than one bit of information**, proving that the bit, contrarily to popular belief, is not the smallest unit of information. Just like atoms are not the smallest particle of matter. Note that in the above table with the 26 digit configurations, each digit $a(1)$, ... $a(13)$ is equal to 0 exactly 13 times, and equal to 1 exactly 13 times. What a superb pattern! It sounds like a fractional factorial table used in designs of experiments or clinical trials.

Finally, autocorrelations in the { $x(n)$ } or { $a(n)$ } sequences, are very strong in my new system. However, these autocorrelations depend on c (but not on x.) Out of curiosity, I tried to find some values of c that yield very little correlation (lag-1 autocorrelation)

between $x(n)$ and $x(n+1)$. Indeed, such c's are easy to find, but in all cases, it results in a strong lag-2 autocorrelation (correlation between $x(n)$ and $x(n+2)$). The findings are summarized in the chart below.



Scatterplot: Lag-1 (X-axis) vs Lag-2 (Y-axis) autocorrelations in the { x(n) } sequence, for 500 values of c

Yes, this is really a scatterplot, produced using about 500 values of c, all with $x = 0$. Yet all points are concentrated on the special blue curve. The details are in this spreadsheet.

Note that some points in the chart seem to be a little off the main curve. This is due to the fact that any irrational value of c that is very close to a simple rational number, is causing stability issues. The 500 values of c were generated using the function RAND() in Excel, thus some are bound to be very close to simple fractions such as 7/2. Also, these computed correlations are approximations, computed on the first 1,000 terms of the sequence { $x(n)$ }.

The lag-d autocorrelations $R(d)$ (d = 1, 2, and so on) can be efficiently computed on the first n terms using the formula

$$R_n(d) = \frac{12}{n} \cdot \sum_{k=1}^{n} \left(x(k+d) - \frac{1}{2} \right) \left(x(k) - \frac{1}{2} \right).$$

This formula actually applies to any number representation system that has a uniform equilibrium distribution on [0, 1]. The theoretical lag-d auto-correlation can be obtained using the above formula and letting n tends to infinity. Its value depends on c and d (but not on x.) For d = 1,we have:

$$R_\infty(1) = 12 \cdot \left(\int_0^1 y \cdot g(y) dy - \frac{1}{4} \right),$$

where g is the function that characterizes the system, in this case $g(x) = x + c - \text{INT}(x + c)$. The same applies to the system studied in the next section, but using a slightly different g.

Finally, another drawback of this system is that it won't map onto other systems, because of these holes that standard systems do not have. To the contrary, there is a mapping between the Logistic Map and the base-2 system, see chapter 10.

4. Towards a more general, better, hybrid system

One would think that by combining the base-2 system, together with the system described in the previous section, you would get a system with many desirable properties. This system is defined as follows:

$$g(x) = bx + c - \lfloor bx + c \rfloor,\ a(n) = \lfloor 2x(n) \rfloor,\ x \in [0,1],\ n = 1,2,\cdots$$

where the parameter b plays the same role as in the base-b system, and the parameter c is the same as in section 3. If $b = 1$, this is identical to the system described in section 3. If $b = 2$ and $c = 0$, it is identical to the base-2 system. I am not sure whether a simple, exact formula is available for $x(n)$.

I have found that when $b = 2$, based on 5,000 test seeds, all 64 possible combinations are represented for the first 6 binary digits of x. This means that this system probably has no hole, but unfortunately, it must have bad seeds: You cannot win both ways. On the plus side, it also means that one digit carries at least one bit of information, usually a desirable property in business applications. The equilibrium distribution for $x(n)$ is also the uniform distribution on [0, 1] assuming b is an integer. The sequence $\{ x(n) \}$ still has auto-correlations (depending on c), but that is also true for the base-2 system.

From now on, we focus exclusively on the case $b = 2$. Some values of c, for instance $c = \log(6) / \log(10)$, yield a lag-1 autocorrelation very close to 0 (a desirable property.) By contrast, in the standard base-2 system (corresponding to $c = 0$) the lag-1 autocorrelation is equal to 1/2. We discuss here two data science techniques that are useful in this context.

Faulty digits, ergodicity, and high precision computing

All the systems investigated here are *ergodic*, in the sense that they do have a unique equilibrium distribution that does not depend on the (good) seed. In such systems, there are two approaches to investigate statistical properties, for instance to analyze auto-correlations in the $\{ x(n) \}$ or $\{ a(n) \}$ sequences:

- Look at just one good seed, and compute a large number of digits for the seed in question. All good seeds behave the same way. Even if the digits are wrong, say beyond $n = 40$, it has no impact on global statistical properties in general, as the error process is similar to re-starting the sequence with a new seed every now and then.

- Look at a large number of good seeds, compute the first few digits, and average across those seeds.

The first approach can result in numbers (digits) that are entirely wrong, beyond $n = 40$, if you use standard arithmetic, due to cumulative errors spreading exponentially fast. This was the case here when using a value of b that is an even integer, due to some dynamics in the way arithmetic computations are performed in Excel (after $n = 40$, and strangely, only for even integer values of b, all $x(n)$'s suddenly vanished -- the implicit "re-seeding" mechanism failed.) The same may happen in any programming language. As a result, the second approach is favored.

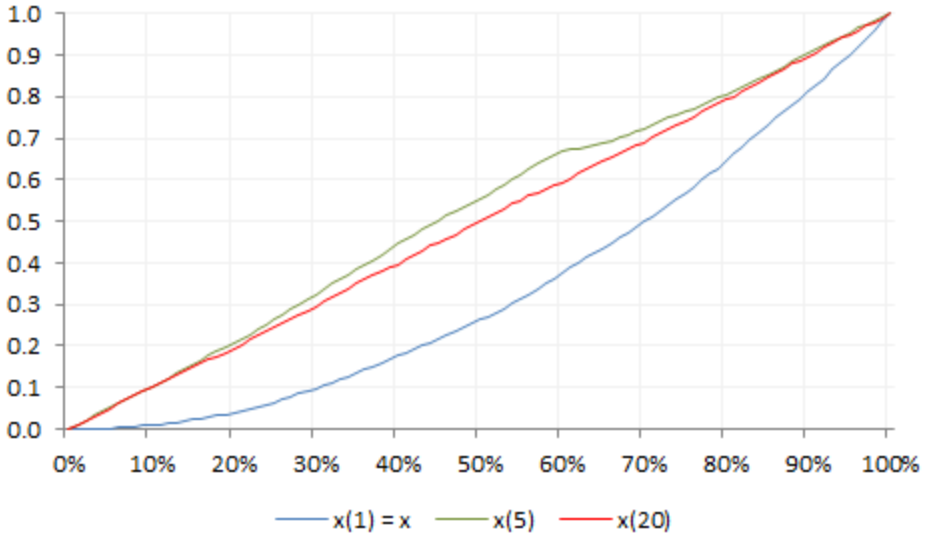
However, there is another workaround: It consists of using high precision computing, and this is described in chapter 8. After all, how do you expect, for a given number, to get 5,000 correct digits when machine accuracy (with standard arithmetic) is limited to 15 decimals or so?

Finding the equilibrium distribution with the percentile test

You can always find the equilibrium distribution (at least a very good approximation) by solving the stochastic integral equation associated with your system, and it typically has only one solution. It is actually not that hard to solve, even accessible to college students in their first year. It is illustrated in chapter 7. I solved quite a few of them (with exact solution), but in each case, I started with empirical results to "guess" the equilibrium distribution, before confirming the result with a theoretical proof. We discuss this empirical approach here, which relies on percentile computations and fitting the observed (empirical) percentiles distribution with a few models. The details are available in my Excel spreadsheet (see last sub-section in this chapter.)

In this case, for $b = 2$ and $c = \log(6) / \log(10)$, I proceeded as follows:

- Generate 5,000 seeds, assumed to be good seeds, using the RAND() function in Excel. I chose to have those seeds NOT uniformly distributed on [0, 1], to see how fast the convergence to the uniform equilibrium distribution occurs, as n increases. The seed x is also (by construction) equal to x(1).
- I computed the percentiles of $x(n)$ for $n = 1$ (non-uniform by construction), $n = 5$, and $n = 20$, using the percentile function in Excel. It is obvious even to the naked eye, that when $n = 20$, we have reached a very stable distribution, which turns out to be uniform (as expected.) The only percentile chart that corresponds to a uniform distribution is when your curve is a diagonal, and this is the case here (see red line below). The result is shown in the chart below.



Percentiles of x(n) computed across 5,000 seeds: n = 5 is close to uniform, n = 20 is almost uniform

Note that the fast convergence is due to using $b = 2$. If you use $b = 1$ instead (corresponding to the system described in section 3) you will need to use a much larger n to notice convergence, which brings again the issue of numerical accuracy. However, when $b = 1$, the system is numerically stable, you may as well use one seed ($x = 0$ will do, see section 3) and a large n.

Central limit theorem, random walks, Brownian motions, stock market modeling

The famous central limit theorem applies in this context in two different ways:

- The sequences { $x(n)$ } and { $a(n)$ }, when properly scaled, can be used as building blocks to design random walk models and Brownian stochastic processes (see chapter 1), with applications to Fintech. More on this in chapter 0 (read the section on "Connection with Brownian Motions" in that chapter.) The connection to Brownian motions is a consequence of the classic Central Limit Theorem as explained in chapter 1, with some cumulative distributions converging to a Gaussian distribution regardless of the initial distribution.
- If you look at the above chart, we started with $x = x(1)$ being distributed according to some arbitrary (yet not uniform) distribution. Regardless of the arbitrary distribution used to sample the seed x, the limit distribution is always the same for good seeds. Not a Gaussian one, but uniform in this case. In many number representation systems (see table in section 1) the limit distribution -- called *equilibrium distribution* in this article -- is not a uniform one (and never a Gaussian one either since x is in [0, 1]) yet is does not depend on the distribution of the seed. You could consider this as the Central Limit Theorem for number representation systems.

Data set and Excel computations

All the data (simulations) and Excel computations (formulas) related to this section, can be downloaded here.

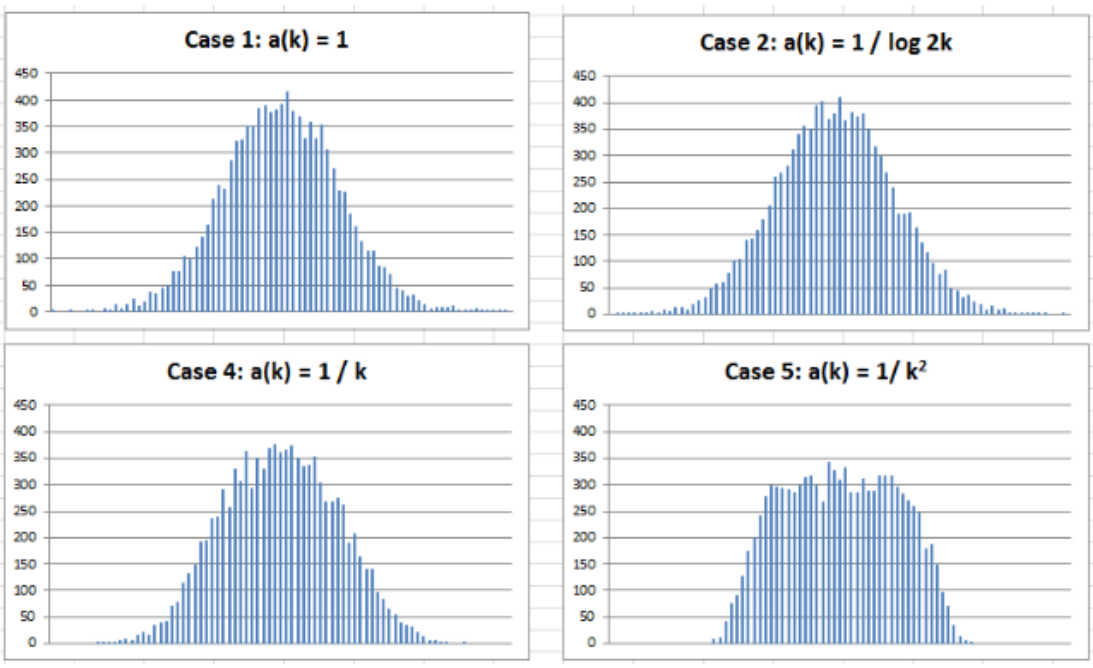
12. The Central Limit Theorem Revisited

The central limit theorem explains the convergence of discrete stochastic processes to Brownian motions, and has been cited a few times in this book. Here we also explore a version that applies to deterministic sequences. Such sequences and treated as stochastic processes in this book.

In this article, we revisit the most fundamental statistics theorem, talking in layman terms. We investigate a special but interesting and useful case, which is not discussed in textbooks, data camps, or data science classes. This material is part of a series about off-the-beaten-path data science and mathematics, offering a fresh, original and simple perspective on a number of topics. Previous articles in this series can be found here and also here.

The theorem discussed here is the central limit theorem. It states that if you average a large number of well-behaved observations or errors, eventually, once normalized appropriately, it has a standard normal distribution. Despite the fact that we are dealing here with a more advanced and exciting version of this theorem (discussing the Liapounov condition), this article is very applied, and can be understood by high school students.

In short, we are dealing here with a not-so-well-behaved framework, and we show that even in that case, the limiting distribution of the “average” can be normal (Gaussian.). More precisely, we show when it is and when it is not normal, based on simulations and non-standard (but easy to understand) statistical tests.



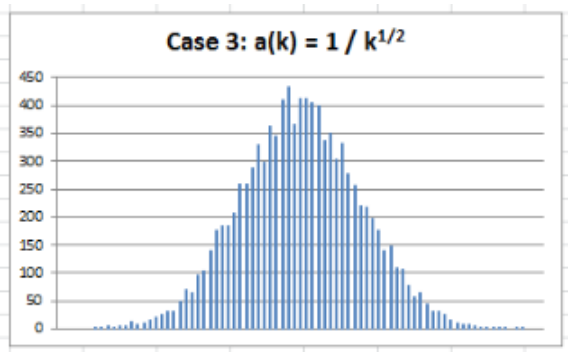


Figure 1: Cases #1, 2 and 3 show convergence to the Gaussian distribution
(Click here for a higher resolution picture)

1. A special case of the Central Limit Theorem

Let's say that we have *n* independent observations X(1),..., X(*n*) and we compute a weighted sum

$$S = a(1)\,X(1) + \ldots + a(n)\,X(n).$$

Under appropriate conditions to be discussed later, (S - E(S)) / Stdev(S) has a normal distribution of mean 0 and variance 1. Here E denotes the expectation and Stdev denotes the standard deviation, that is, the square root of the variance.

This is a non-basic case of the central limit theorem, as we are dealing with a weighted sum. The classic, basic version of the theorem assumes that all the weights a(1),...,a(*n*) are equal. Furthermore, we focus here on the particular case where

1. The highest weights (in absolute value) are concentrated on the first few observations,
2. The weight a(*k*) tends to 0 as *k* tends to infinity.

The surprising result is the fact that even with putting so much weight on the first few observations, depending on how slowly a(*k*) converges to 0, the limiting distribution is still Gaussian.

About the context
You might wonder: how is this of any practical value in the data sets that I have to process in my job? Interestingly, I started to study this type of problems long ago, in the context of *k*-NN (nearest neighbors) classification algorithms. One of the questions, to estimate the local or global intensity of a stochastic point process, and also related to density estimation techniques, was: how many neighbors should we use, and which weights should we put on these neighbors to get robust and accurate estimates? It turned out that putting more weight on close neighbors, and increasingly lower weight on far away neighbors (with weights slowly decaying to zero based on the distance to the neighbor in question) was the solution to the problem. I actually found optimum decaying schedules for the a(*k*)'s, as *k* tends to infinity. You can read the detail here.

2. Generating data, testing, and conclusions

Let's get back to the problem of assessing when the weighted sum S = a(1) X(1) + ... + a(*n*) X(*n*), after normalization, converges to a Gaussian distribution. By normalization, I mean considering (S - E(S)) / Stdev(S), instead of S.

In order to solve this problem, we performed simulations as follows:

Simulations
Repeat *m* = 10,000 times:

- Produce *n* = 10,000 random deviates X(1),...,X(*n*) uniformly distributed on [0, 1]
- Compute S = a(1) X(1) + ... + a(*n*) X(*n*) based on a specific set of weights a(1),...,a(*n*)
- Compute the normalized S, denoted as W = (S - E(S)) / Stdev(S).

Each of the above *m* iterations provides one value of the limiting distribution. In order to investigate the limiting distribution (associated with a specific set of weights), we just need to look at all these *m* values, and see whether they behave like deviates from a Gaussian distribution of mean 0 and variance 1. Note that we found *n* = 10,000 and *m* = 10,000 to be large enough to provide relatively accurate results. We tested various values of *n* before settling for *n* = 10,000, looking at what (little) incremental precision we got from increasing *n* (say) from 500 to 2,000 and so on. Also note that the random number generator is not perfect, and due to numerical approximations made by the computer, indefinitely increasing *n* (beyond a certain point) is not the solution to get more accurate results. That said, since we investigated 5 sets of weights, we performed 5 x *n* x *m* = 500 million computations in very little time. A value of *m* = 10,000 provides about two correct digits when computing the percentiles of the limiting distribution (except for the most extreme ones), provided *n* is large enough.
The source code is provided in the last section.

Analysis and results
We tested 5 sets of weights, see Figure1:

- **Case 1**: a(*k*) = 1, corresponding to the classic version of the Central Limit Theorem, and with guaranteed convergence to the Gaussian distribution.
- **Case 2**: a(*k*) = 1 / log 2*k*, still with guaranteed convergence to the Gaussian distribution
- **Case 3**: a(*k*) = $k^{-1/2}$, the last exponent (-1/2) that still provides guaranteed convergence to the Gaussian distribution, according to the Central Limit Theorem with the Liapounov condition (more on this below.) A value below -1/2 violates the Liapounov condition.
- **Case 4**: a(*k*) = k^{-1}, the limiting distribution looks Gaussian (see Figure 1) but it is too thick to be Gaussian, indeed the maximum is also too low, and the kurtosis is now significantly different from zero, thus the limiting distribution is not Gaussian (though almost).

- **Case 5**: a(k) = k^{-2}, not converging to the Gaussian distribution, but instead to an hybrid continuous distribution, half-way Gaussian, half-way uniform.

Note that by design, all normalized S's have mean 0 and variance 1.

We computed (in Excel) the percentiles of the limiting distributions for each of the five cases. Computations are found in this spreadsheet. We compared the cases 2 to 5 with case 1, computing the differences (also called deltas) for each of the 100 percentiles. Since case 1 corresponds to a normal distribution, we actually computed the deltas to the normal distribution, see Figure 2. The deltas are especially large for the very top or very bottom percentiles in cases 4 and 5. Cases 2 and 3 show deltas close to zero (not statistically significantly different from zero), and this is expected since these cases also yield a normal distribution. To assess the statistical significance of these deltas, one can use the model-free confidence interval technique described here: it does not require any statistical or probability knowledge to understand how it works. Indeed you don't even need a table of the Gaussian distribution for testing purposes here (you don't even need to know what a Gaussian distribution is) as case 1 automatically provides one.

The Liapounov connection

For those interested in the theory, the fact that cases 1, 2 and 3 yield convergence to the Gaussian distribution is a consequence of the Central Limit Theorem under the Liapounov condition. More specifically, and because the samples produced here come from uniformly bounded distributions (we use a random number generator to simulate uniform deviates), all that is needed for convergence to the Gaussian distribution is that the sum of the squares of the weights -- and thus Stdev(S) as n tends to infinity -- must be infinite. This result is mentioned in A. Renyi's book *Probability Theory* (Dover edition, 1998, page 443.)

Note that in cases 1, 2, and 3, the sum of the squares of the weights is infinite. In cases 4 and 5, it is finite, respectively equal to $\pi^2/6$ and $\pi^4/90$ (see here for details.) I am very curious to know what the limiting distribution is for case 4.

3. Generalizations

Here we discuss generalizations of the central limit theorem, as well as potential areas of research

Generalization to correlated observations

One of the simplest ways to introduce correlation is to define a stochastic auto-regressive process using

$$Y(k) = p\, Y(k\text{-}1) + q\, X(k)$$

where X(1), X(2), ... are independent with identical distribution, with Y(1) = X(1) and where p, q are positive integers with $p + q = 1$. The Y(k)'s are auto-correlated, but clearly, Y(k) is a weighted sum of the X(j)'s ($1 <= j <= k$), and thus, S = a(1) Y(1) + ... +

a(*n*) Y(*n*) is also a weighted sum of the X(*k*)'s, with higher weights on the first X(*k*)'s. Thus we are back to the problem discussed in this article, but convergence to the Gaussian distribution will occur in fewer cases due to the shift in the weights.

More generally, we can work with more complex auto-regressive processes with a covariance matrix as general as possible, then compute S as a weighted sum of the X(*k*)'s, and find a relationship between the weights and the covariance matrix, to eventually identify conditions on the covariance matrix that guarantee convergence to the Gaussian distribution.

Generalization to non-random (static) observations

Is randomness really necessary for the central limit theorem to be applicable and provable? What about the following experiment:

- Compute all unordered sums S made up of *n* integers, 0 or 1, with repetitions allowed. For instance, if *n* = 2, the four possibilities are 0+0, 0+1, 1+0, 1+1. For an arbitrary *n*, we have 2^n possibilities.
- Normalize S as usual. For normalization, here use E(S) = *n*/2 and Stdev(S) = SQRT(*n*) / 2.

Do these 2^n normalized values of S (generated via this non-random experiment) follow a Gaussian distribution as *n* tends to infinity? Ironically, one way to prove that this is the case (I haven't checked if it is the case or not, but I suspect that it is) would be to randomly sample *m* out of these 2^n values, and then apply the central limit theorem to the randomly selected values as *m* tends to infinity. Then by increasing *m* until *m* is as large as 2^n we would conclude that the central limit theorem also holds for the non-random (static) version of this problem. The limiting distribution definitely has a symmetric, bell-like shape, just like the Gaussian distribution, though this was also the case in our above "case 4" example -- yet the limit was not Gaussian. More on this in the next chapter.

Other interesting stuff related to the Central Limit Theorem

There is a lot of interesting stuff on the Wikipedia entry, including about the Liapounov condition. But the most interesting things, at least in my opinion, were the following:

- The area S of a convex hull of *n* points X(1),...,X(*n*) also converges to a normal distribution, once standardized, that is when considering (S - E(S)) / Stdev(S).
- Under some conditions, the result below applies, with C being a universal constant:

$$\left|\mathbb{P}\left(a \leq \frac{X_1 + \cdots + X_n}{\sqrt{n}} \leq b\right) - \frac{1}{\sqrt{2\pi}} \int_a^b \mathrm{e}^{-t^2/2}\,\mathrm{d}t\right| \leq \frac{C}{n}$$

- If instead of a weighted average S, we consider the maximum M = max(X(1),...,X(*n*)), then we also have a limiting distribution for (M - E(M)) / Stdev(M) after proper standardization. This is known as the Fisher–Tippett–Gnedenko theorem in extreme value theory. The limit distribution is not Gaussian.

What would happen if instead of the maximum or weighted average, we consider the empirical percentiles?

- The digits for the vast majority of numbers, in all number representation systems, can be used to emulate Brownian motions, thanks to the central limit theorem. For details, see chapter 9.

Another potential generalization consists of developing a central limit theorem that is based on L^1 rather than L^2 measures of centrality and dispersion, that is, the median and absolute deviations rather than the mean and variance. This would be useful when the observations come from a distribution that does not have a mean or variance, such as Cauchy.

Also, does the limit distribution in case 4 depend on the distribution of the X(*k*)'s -- in this case uniform -- or is it a universal distribution that is the same regardless of the X(*k*)'s distribution? Unfortunately, the answer is negative: after trying with the square of uniform deviates for the X(*k*)'s, the limit distribution was not symmetric, and thus different from the one obtained with uniform deviates.

4. Appendix: source code and chart

Below is the source code (Perl) used to produce the simulations:

```
$seed=100;
$c=-0.5; # the exponent in a(k) = k^c
$n=10000;
open(OUT,">out.txt");
for ($m=0; $m<10000; $m++) {
  $den=0;
  $num=0;
  $ss=0;
  for ($k=1; $k<=$n; $k++) {
    $r=rand();
    $aux=exp($c*log($k)); # k^c
    $num+=$r*$aux;
    $den+=$aux;
    $ss+=($aux*$aux);
  }
  $dev=$num/$den;
  $std=sqrt(1/12) * (sqrt($ss)/$den); # 1/12 for Uni[0,1]
  $dev2=($dev-0.5)/$std;
  print OUT "$m\t$dev2\n";
}
close(OUT);
```

Also, Figure 2 below is referenced earlier in this article.

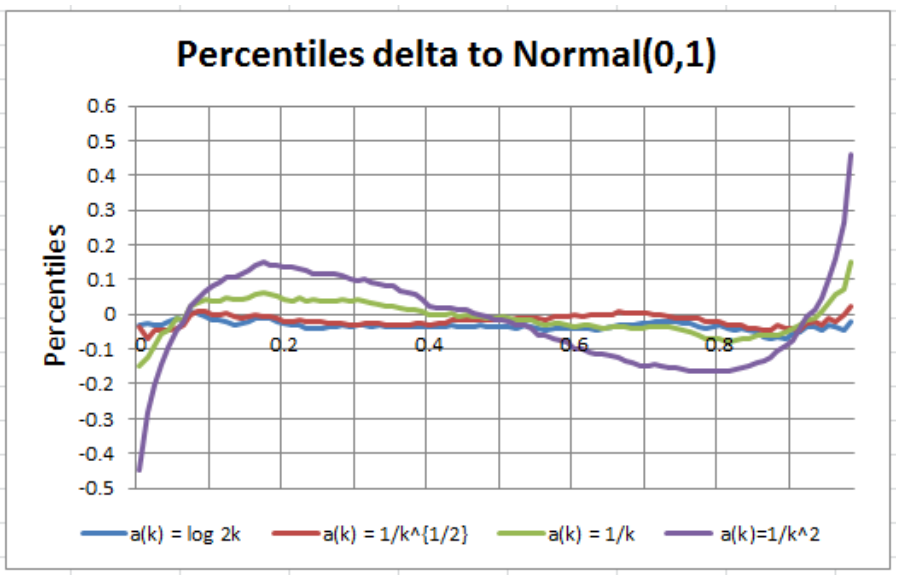


Figure 2: Two of the weight sets clearly produce non-Gaussian limit distributions

13. How to Detect if Numbers are Random or Not

We explore here some deterministic sequences of numbers, behaving like stochastic processes or chaotic systems, together with another interesting application of the central limit theorem.

In this article, you will learn some modern techniques to detect whether a sequence appears as random or not, whether it satisfies the central limit theorem (CLT) or not -- and what the limiting distribution is if CLT does not apply -- as well as some tricks to detect abnormalities. Detecting lack of randomness is also referred to as signal versus noise detection, or pattern recognition.

It leads to the exploration of time series with massive, large-scale (long term) auto-correlation structure, as well as model-free, data-driven statistical testing. No statistical knowledge is required: we will discuss deep results that can be expressed in simple English. Most of the testing involved here uses big data (more than a billion computations) and data science, to the point that we reached the accuracy limits of our machines. So there is even a tiny piece of numerical analysis in this article.

Potential applications include testing randomness, Monte Carlo simulations for statistical testing, encryption, blurring, and steganography (encoding secret messages into images) using pseudo-random numbers. A number of open questions are discussed here, offering the professional post-graduate statistician new research topics both in theoretical statistics and advanced number theory. The level here is state-of-the-art, but we avoid jargon and some technicalities to allow beginners and non-statisticians to understand and enjoy most of the content. An Excel spreadsheet, attached to this document, summarizes our computations and will help you further understand the methodology used here.

Interestingly, I started to research this topic by trying to apply the notorious (CLT, see chapter 12) to non-random (static) variables -- that is, to fixed sequences of numbers that look chaotic enough to simulate randomness. Ironically, it turned out to be far more complicated than using CLT for regular random variables. So I start here by describing what the initial CLT problem was, before moving into other directions such as testing randomness, and the distribution of the largest gap in seemingly random sequences. As we will see, these problems are connected.

1. Central Limit Theorem for Non-Random Variables

Here we are interested in sequences generated by a periodic function f(x) that has an irrational period T, that is f(x+T) = f(x). Examples include f(x) = sin x with $T = 2\pi$, or f(x)

= {αx} where α > 0 is an irrational number, { } represents the fractional part and $T = 1/\alpha$. The k-th element in the infinite sequence (starting with $k = 1$) is f(k). The central limit theorem can be stated as follows:

Under certain conditions to be investigated -- mostly the fact that the sequence seems to represent or simulate numbers generated by a well-behaved stochastic process -- we would have:

$$U(n) = \frac{\sqrt{n}}{\sigma} \cdot \left(\frac{1}{n} \sum_{k=1}^{n} f(k) - \mu \right) \rightarrow N(0,1) \text{ with } \mu = \frac{1}{T} \int_0^T f(x)dx \text{ and } \sigma^2 = \frac{1}{T} \int_0^T (f(x) - \mu)^2 dx$$

In short, $U(n)$ tends to a normal distribution of mean 0 and variance 1 as n tends to infinity, which means that as both n and m tends to infinity, the values $U(n+1)$, $U(n+2)$ ... $U(n+m)$ have a distribution that converges to the standard bell curve.

From now on, we are dealing exclusively with sequences that are equidistributed over [0, 1], thus μ = 1/2 and σ = SQRT(1/12). In particular, we investigate f(x) = {αx} where α > 0 is an irrational number and { } the fractional part. While this function produces a sequence of numbers that seems fairly random, there are major differences with truly random numbers, to the point that CLT is no longer valid. The main difference is the fact that these numbers, while somewhat random and chaotic, are much more evenly spread than random numbers. True random numbers tend to create some clustering as well as empty spaces. Another difference is that these sequences produce highly auto-correlated numbers.

As a result, we propose a more general version of CLT, redefining $U(n)$ by adding two parameters a and b:

$$U(n) = \frac{n^a (\log n)^b}{\sigma} \cdot \left(\frac{1}{n} \sum_{k=1}^{n} f(k) - \mu \right)$$

This more general version of CLT can handle cases like our sequences. Note that the classic CLT corresponds to $a = 1/2$ and $b = 0$. In our case, we suspect that $a = 1$ and b is between 0 and -1. This is discussed in the next section.

Note that if instead of f(k), the k^{th} element of the sequence is replaced by f(k^2) then the numbers generated behave more like random numbers: they are less evenly distributed and less auto-correlated, and thus the CLT might apply. We haven't tested it yet. You will also find an application of CLT to non-random variables, as well as to correlated variables, in chapter 12.

2. Testing Randomness: Max Gap, Auto-Correlations and More

Let's call the sequence f(1), f(2), f(3) ... f(n) generated by our function f(x) an α-sequence. Here we compare properties of α-sequences with those of random numbers on [0, 1] and we highlight the striking differences. Both sequences, when n tends to infinity, have a mean value converging to 1/2, a variance converging to 1/12 (just like

any uniform distribution on [0, 1]), and they both look quite random at first glance. But the similarities almost stop here.

Maximum gap

The maximum gap among *n* points scattered between 0 and 1 is another way to test for randomness. If the points were truly randomly distributed, the expected value for the length of the maximum gap (also called longest segment) is known and is equal to

$$\frac{1}{n}\sum_{k=1}^{n}\frac{1}{k} \sim \frac{\log(n)}{n}$$

See this article for details, or the book Order Statistics published by Wiley, page 135. The max gap values have been computed in the spreadsheet (see section below to download the spreadsheet) both for random numbers and for α-sequences. It is pretty clear from the Excel spreadsheet computations that the average maximum gaps have the following expected values, as *n* becomes very large:

- Maximum gap for random numbers: log(*n*)/*n* as expected from the above theoretical formula
- Maximum gap for α-sequences: *c*/*n* (*c* is a constant close to 1.5; the result needs to be formally proved)

So α-sequences have points that are far more evenly distributed than random numbers, by an order of magnitude, not just by a constant factor! This is true for the 8 α-sequences (8 different values of α) investigated in the spreadsheet, corresponding to 8 "nice" irrational numbers (more on this in the research section below, about what a "nice" irrational number might be in this context.)

Auto-correlations

Unlike random numbers, values of f(*k*) exhibit strong, large-scale auto-correlations: f(*k*) is strongly correlated with f(*k*+*p*) for some values of *p* as large as 100. The successive lag-*p* auto-correlations do not seem to decay with increasing values of *p.* To the contrary, it seems that the maximum lag-*p* auto-correlation (in absolute value) seems to be increasing with *p*, and possibly reaching very close to 1 eventually. This is in stark contrast with random numbers: random numbers do not show auto-correlations significantly different from zero, and this is confirmed in the spreadsheet. Also, the vast majority of time series have auto-correlations that quickly decay to 0. This surprising lack of decay could be the subject of some interesting number theoretic research. These auto-correlations are computed and illustrated in the Excel spreadsheet (see section below) and are worth checking out.

Convergence of *U*(*n*) to a non-degenerate distribution

Figures 2 and 3 in the next section (extracts from our spreadsheet) illustrate why the classic central limit theorem (that is, *a* = 1/2, *b* =0 for the *U*(*n*) formula) does not apply to α-sequences, and why *a* = 1 and *b* = *0* might be the correct parameters to use instead. However, with the data gathered so far, we can't tell whether *a* = 1 and *b* = 0 is correct, or whether *a* = 1 and *b* = -1 is correct: both exhibit similar asymptotic behavior,

and the data collected is not accurate enough to make a final decision on this. The answer could come from theoretical considerations rather than from big data analysis. Note that the correct parameters should produce a somewhat horizontal band for $U(n)$ in figure 2, with values mostly concentrated between -2 and +2 due to normalization of $U(n)$ by design. And $a = 1$, $b = 0$, as well as $a = 1$, $b = -1$, both do just that, while it is clear that $a = 1/2$ and $b = 0$ (classic CTL) fails as illustrated in figure 3. You can play with parameters a and b in the spreadsheet, and see how it changes figure 2 or 3, interactively.

One issue is that we computed $U(n)$ for n up to 100,000,000 using a formula that is ill-conditioned: multiplying a large quantity n by a value close to zero (for large n) to compute $U(n)$, when the precision available is probably less than 12 digits. This might explain the large, unexpected oscillations found in figure 2. Note that oscillations are expected (after all, $U(n)$ is supposed to converge to a statistical distribution, possibly the bell curve, even though we are dealing with non-random sequences) but such large-scale, smooth oscillations, are suspicious.

3. Excel Spreadsheet with Computations

Click here to download the spreadsheet. The spreadsheet has 3 tabs: One for α-sequences, one for random numbers -- each providing auto-correlation, max gap, and some computations related to estimating a and b for $U(n)$ -- and a tab summarizing n = 100,000,000 values of $U(n)$ for α-sequences, as shown in figures 2 and 3. That tab, based on data computed using a Perl script, also features moving maxima and moving minima, a concept similar to moving averages, to better identify the correct parameters a and b to use in $U(n)$.

Confidence intervals (CI) can be empirically derived to test a number of assumptions, as illustrated in figure 1: in this example, based on 8 measurements, it is clear that maximum gap CI's for α-sequences are very different from those for random numbers, meaning that α-sequences do not behave like random numbers.

1.49	1.72	1.50	1.10	1.64	1.73	1.40	1.94
9.78	14.89	12.51	9.28	9.60	9.25	9.33	9.10

***Figure 1**: max gap times n (n = 10,000), for 8 α-sequences (top) and 8 sequences of random numbers (bottom)*

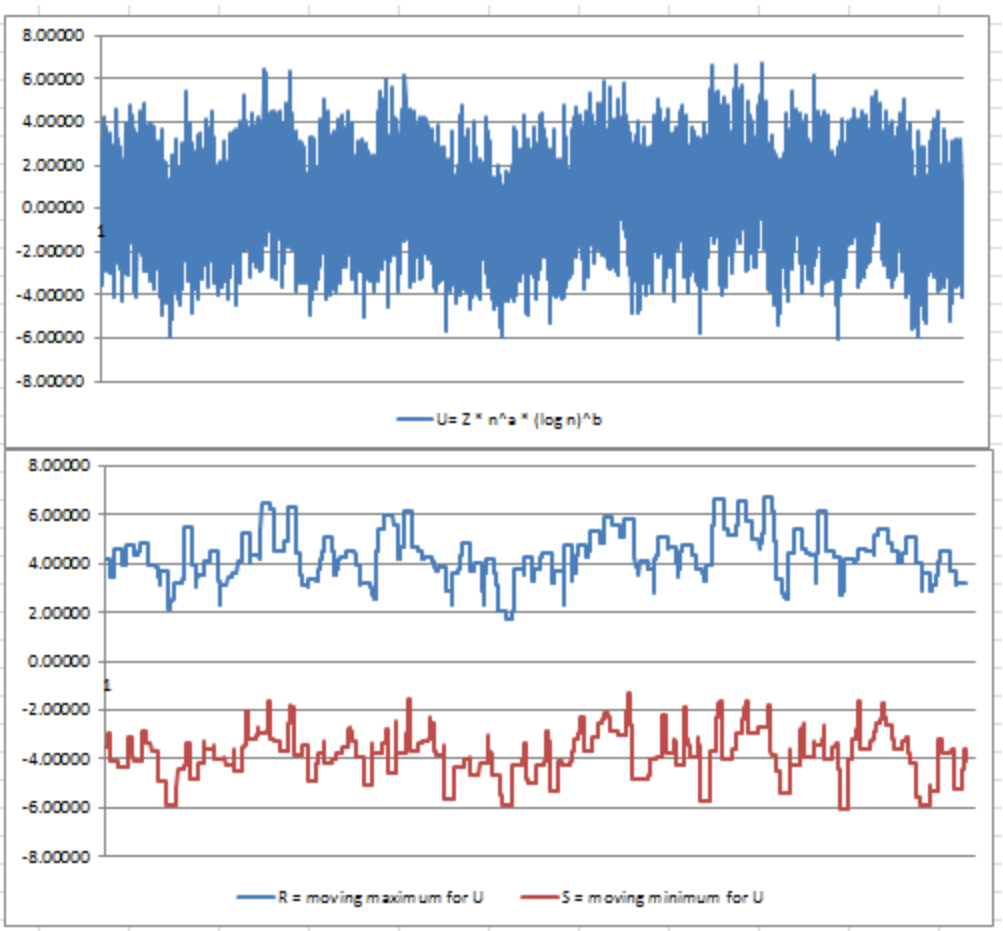


***Figure 2**: U(n) with a = 1, b = 0 (top) and moving max / min bands (bottom) for α-sequences*

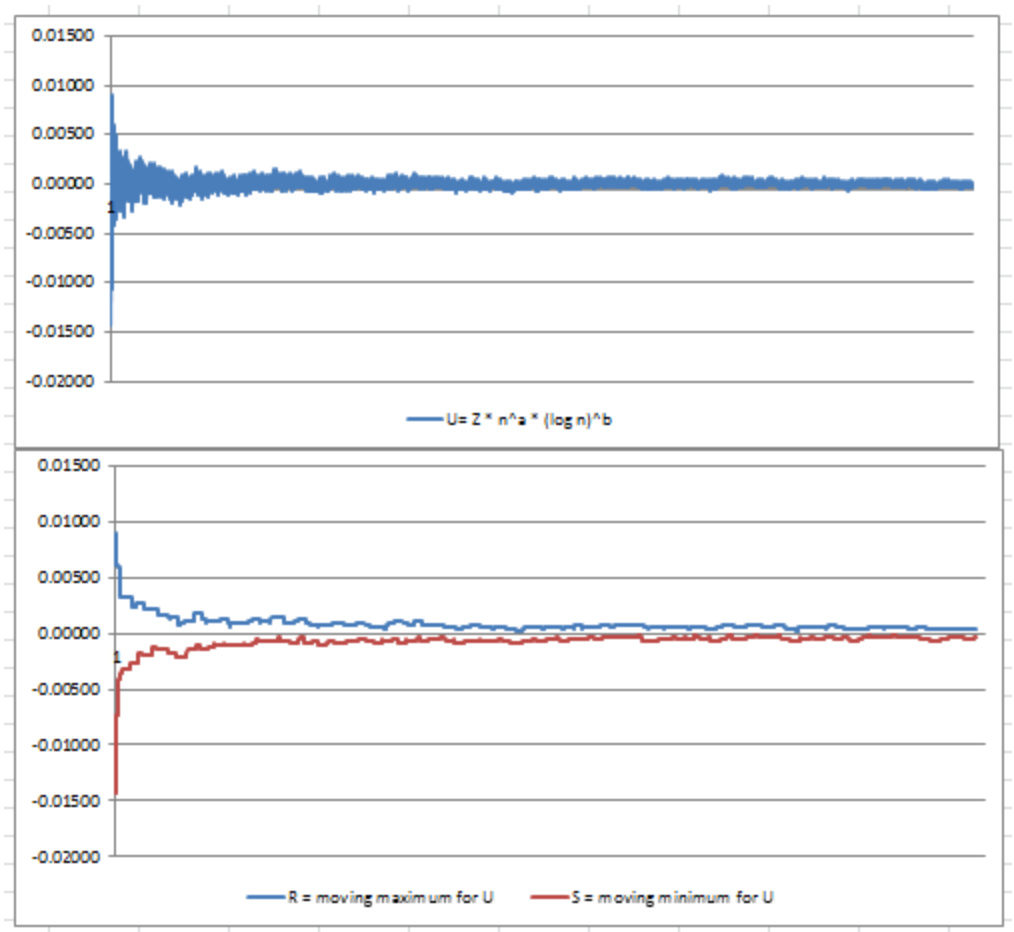


***Figure 3**: U(n) with a = 0.5, b = 0 (top) and moving max / min bands (bottom) for α–sequences*

4. Potential Research Areas

Here we mention some interesting areas for future research. By sequence, we mean α-sequence as defined in section 2, unless otherwise specified.

- Using f(k^c) as the k^{th} element of the sequence, instead of f(k). Which values of c > 0 lead to equidistribution over [0, 1], as well as yielding the classic version of CLT with a = 1/2 and b = 0 for $U(n)$? Also what happens if f(k) = {$\alpha p(k)$} where $p(k)$ is the k^{th} prime number and { } represents the fractional part? This sequence was proved to be equidistributed on [0, 1] (this by itself.is a famous result of analytic number theory, published by Vinogradov in 1948) and has a behavior much more similar to random numbers, so maybe the classic CLT applies to this sequence? Nobody knows.

- What is the asymptotic distribution of the moments and distribution of the maximum gap among the n first terms of the sequence, both for random numbers on [0, 1] and for the sequences investigated in this article? Does it depend on the parameter α? Same question for minimum gap and other metrics used to test

randomness, such as point concentration, defined for instance in the article On Uniformly Distributed Dilates of Finite Integer Sequences?

- Does $U(n)$ depend on α? What are the best choices for α, to get as much randomness as possible? In a similar context, SQRT(2)-1 and (SQRT(5)-1)/2 are found to be good candidates: see this Wikipedia article (read the section on additive recurrence.) Also, what are the values of the coefficients a and b in $U(n)$, for α-sequences? It seems that a must be equal to 1 to guarantee convergence to a non-degenerate distribution. Is the limiting distribution for $U(n)$ also normal for α-sequences, when using the correct a and b?

- What happens if α is very close to a simple rational number, for instance if the first 500 digits of α are identical to those of 3/2?

Generalization to higher dimensions

So far we worked in dimension 1, the support domain being the interval [0, 1]. In dimension 2, $f(x) = \{\alpha x\}$ becomes $f(x, y) = (\{\alpha x\}, \{\beta y\})$ with α, β, and α/β irrational; $f(k)$ becomes $f(k,k)$. Just like the interval [0, 1] can be replaced by a circle to avoid boundary effects when deriving theoretical results, the square [0, 1] x [0, 1] can be replaced by the surface of the torus. The maximum gap becomes the maximum circle (on the torus) with no point inside it. The range statistic (maximum minus minimum) becomes the area of the convex hull of the n points. For a famous result regarding the asymptotic behavior of the area of the convex hull of a set of n points, see chapter 12 and check out the sub-section entitled "Other interesting stuff related to the Central Limit Theorem." Note that as the dimension increases, boundary effects become more important.

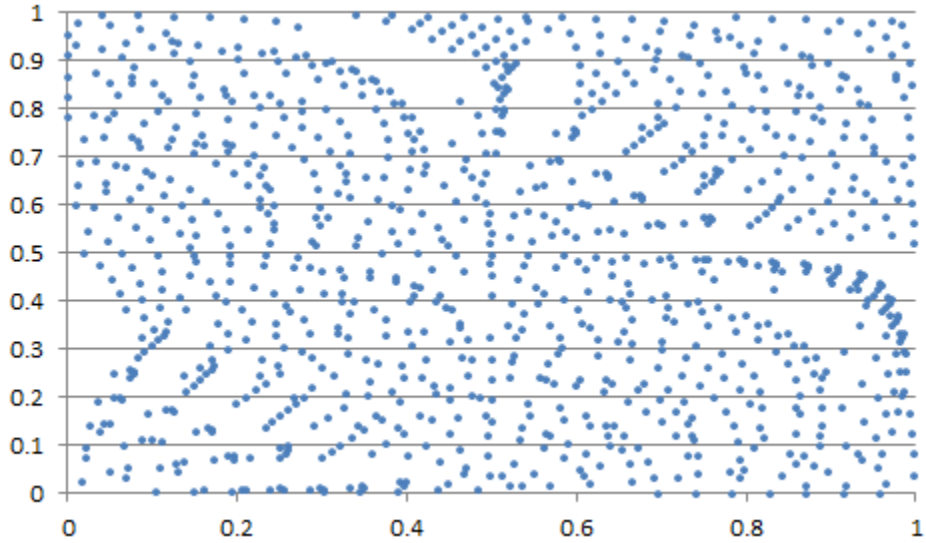


***Figure 4**: bi-variate example with c = 1/2, α = SQRT(31), β = SQRT(17) and n = 1000 points*

Figure 4 shows an unusual example in two dimensions, with strong departure from randomness, at least when looking at the first 1,000 points. Usually, the point pattern looks much more random, albeit not perfectly random, as in Figure 5.

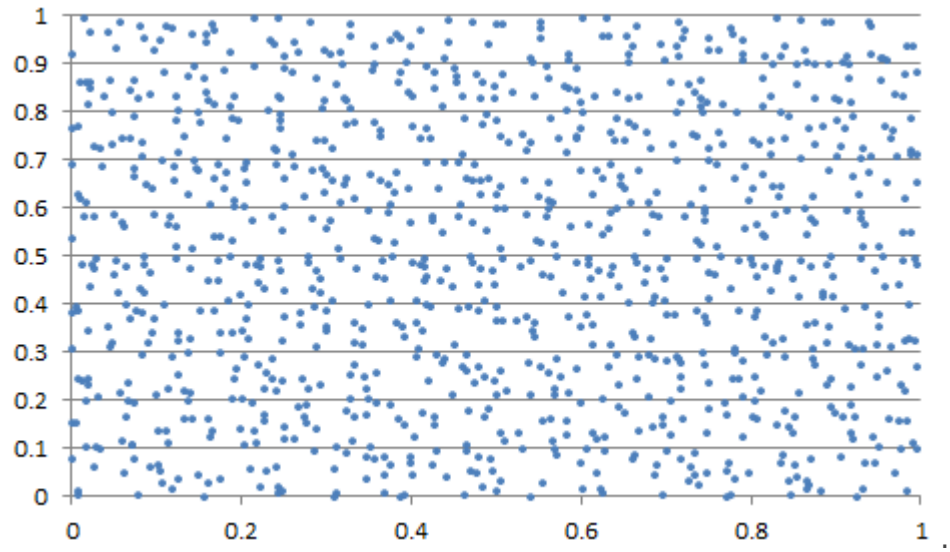


***Figure 5**: bi-variate example with c = 1/2, α = SQRT(13), β = SQRT(26) and n = 1000 points*

Computations are found in this spreadsheet. Note that we've mostly discussed the case $c = 1$ in this chapter. The case $c = 1/2$ creates interesting patterns, and the case $c = 2$ produces more random patterns. The case $c = 1$ creates very regular patterns (points evenly spread, just like in one dimension.)

14. Distribution of Arrival Times for Extreme Events

Time series, as discussed in the first chapters, are also stochastic processes. Here we discuss a topic rarely investigated in the literature: the arrival times, as opposed to the extreme values (a classic topic), associated with extreme events in time series.

Most of the articles on extreme events are focusing on the extreme values. Very little has been written about the arrival times of these events. This article fills the gap.

We are interested here in the distribution of arrival times of successive records in a time series, with potential applications to global warming assessment, sport analytics, or high frequency trading. The purpose here is to discover what the distribution of these arrival times is, in absence of any trends or auto-correlations, for instance to check if the global warming hypothesis is compatible with temperature data obtained over the last 200 years. In particular it can be used to detect subtle changes that are barely perceptible yet have a strong statistical significance. Examples of questions of interest are:

- How likely is it that 2016 was the warmest year on record, followed by 2015, then by 2014, then by 2013?
- How likely is it, in 200 years' worth of observations, to observe four successive records four years in a row, at any time during the 200 years in question?

The answer to the first question is that it is very unlikely to happen just by chance..

Despite the relative simplicity of the concepts discussed here, and their great usefulness in practice, none of the material below is found in any statistics textbook, as far as I know. It would be good material to add to any statistics curriculum.

1. Simulations

I run a number of simulations, generating 100 time series each made up of millions of random, independent Gaussian deviates, without adding any trend up or down. The first few hundred points of one of these time series is pictured in Figure 1.

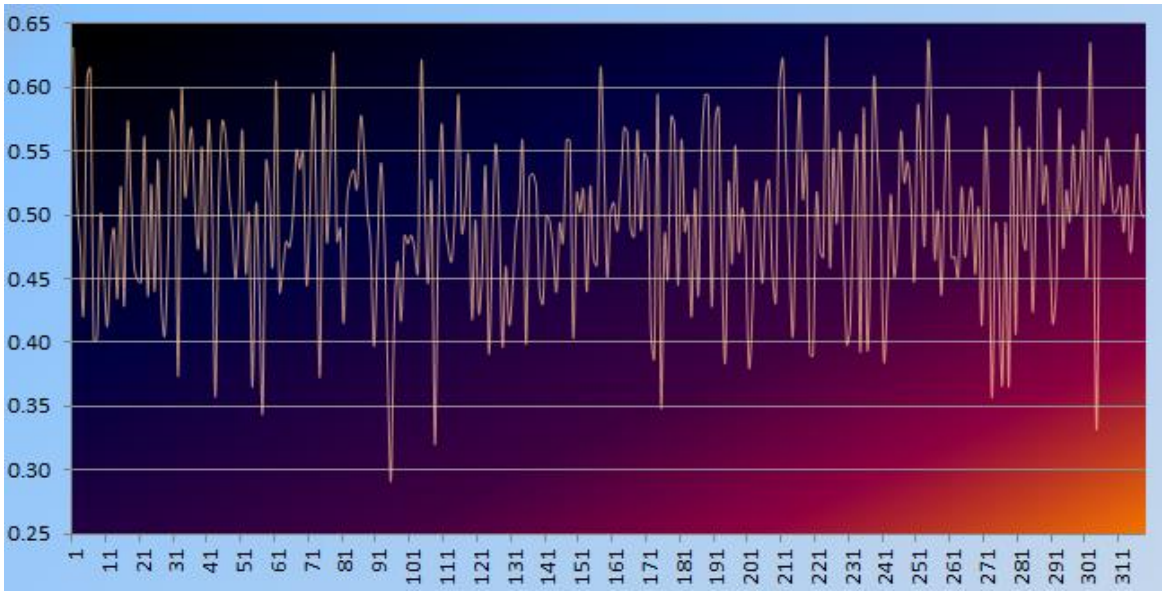


***Figure 1**: Example of time series with no periodicity and no trend (up or down)*

I computed the median, 25- and 75-percentiles for the first few records, see Figure 2. For instance, the median time of occurrence of the first record (after the first measurement) is after 2 years, if your time unit is a year. The next bigger record is expected 8 years after the first measurement, and the next bigger one 21 years after the first measurement (see Figure 2.) Even if you look at the 25-percentile, it really takes a lot of years to beat the previous 4 or 5 records in a row. In short, it is nearly impossible to observe increasing records four years in a row, unless there is a trend that forces the observed values to become larger over time.

Record #	Median	P.25	P.75	Min
1	2.0	1.0	3.3	1
2	8.0	3.8	21.0	2
3	21.0	7.8	91.0	3
4	65.5	23.3	243.8	4
5	201.0	62.5	755.3	7
6	539.5	149.3	3,360.5	9
7	1,185.5	300.5	7,167.3	12
8	4,956.0	950.3	25,033.5	22
9	12,317.5	2,388.3	93,059.8	50
10	35,644.5	5,869.0	300,711.3	89

***Figure 2**: Time of arrivals of successive records (in years if you time unit is a year)*

This study of arrival times for these records should allow you to detect even very tiny trends, either up or down, better than traditional models of change point detection hopefully. However it does not say anything about whether the increase is barely perceptible or rather large.

Note that the values of these records is a subject of much interest in statistics, known as extreme value theory. This theory has been criticized for failure to predict the amount of damage in modern cataclysms, resulting in big losses for insurance companies. Part of the problem is that these models are based on hundreds of years' worth of data (for instance to predict the biggest flood that can occur in 500 years) but over such long periods of time, the dynamics of the processes at play have shifted. Note that here, I focus on the arrival times or occurrences of these records, not on their intensity or value, contrarily to traditional extreme value theory.

Finally, arrival times for these records do not depend on the mean or variance of the underlying distribution. Figure 2 provides some good approximations, but more tests and simulations are needed to confirm my findings. Are these median arrival times the same regardless of the underlying distribution (temperature, stock market prices, and so on) just like the central limit theorem (see chapter 12) provides a same limiting distribution regardless of the original, underlying distribution? The theoretical statistician should be able to answer this question. I didn't find many articles on the subject in the literature, though this one is interesting. In the next section, I try to answer this question. The answer is positive.

2. Theoretical Distribution of Records over Time

This is an interesting combinatorial problem, and it bears some resemblance to the Analyticbridge Theorem. Let R(n) be the value of the n^{th} record (n = 1, 2,...) and T(n) its arrival time.

For instance, if the data points (observed values) are X(0) = 1.35, X(1) = **1.49**, X(2) = 1.43, X(3) = **1.78**, X(4) = 1.63, X(5) = 1.71, X(6) = 1.45, X(7) = **1.93**, X(8) = 1.84, then the records (highlighted in bold) are R(1) = 1.49, R(2) = 1.78, R(3) = 1.93, and the arrival times for these records are T(1) = 1, T(2) = 3, and T(3) = 7.

To compute the probability P(T(n) = k) for n > 0 and k = n, n+1, n+2, etc., let's define T(n, m) as the arrival time of the n-th record if we only observe the first m+1 observations X(0), X(1), ..., X(m). Then P(Tn) = k) is the limit of P(T(n, m) = k) as m tends to infinity, assuming the limit exists. If the underlying distribution of the values X(0), X(1), etc. is continuous, then, due to the symmetry of the problem, computing P(T(n, m) = k) can be done as follows:

1. Create a table of all (m+1)! (factorial m+1) permutations of (0, 1, ... , m).
2. Compute N(n, m, k), the number of permutations of (0, 1, ..., m) where the n-th record occurs at position k in the permutation (with 0 < k <= m). For instance, if m = 2, we have 6 permutations (0, 1, 2), (0, 2, 1), (1, 0, 2), (1, 2, 0), (2, 0, 1) and (2, 1, 0). The first record occurs at position k = 1 only for the following three permutations: (0, 1, 2), (0, 2, 1), and (1, 2, 0). Thus, N(1, 2, 1) = 3. Note that the first element in the permutation is assigned position 0, the second one is assigned position 1, and so on. The last one has position m.
3. Then P(T(n, m) = k) = N(n, m, k) / (m+1)!

As a result, the distribution of arrival times, for the records, is universal: it does not depend on the underlying distribution of the identically and independently distributed observations X(0), X(1), X(2) etc.

It is easy (with or without using my above combinatorial framework) to find that the probability to observe a record (any record) at position k is $1/(k+1)$ assuming again that the first position is position 0 (not 1). Also, it is easy to prove that $P(T(n) = n) = 1/(n+1)!$. Now, $T(1) = k$ if and only if $X(k)$ is a record among $X(0), ..., X(k)$ and $X(0)$ is the largest value among $X(0), ..., X(k-1)$. Thus:

$$P(T(1) = k) = 1 / \{ (k+1)k \}$$

This result is confirmed by my simulations. For the general case, recurrence formulas can be derived.

3. Useful Results

None of the arrival times $T(n)$ for the records has a finite expectation. Figure 3 displays the first few values for the probability that the n^{th} record occurs at position $T(n) = k$, the first element in the data set being assigned to position 0. The distribution of these arrival times does not depend on the underlying distribution of the observations.

		n								
		1	2	3	4	5	6	7	8	9
	1	1								
	2	1	1							
	3	2	3	1						
	4	6	11	6	1					
k	5	24	50	35	10	1				
	6	120	274	225	85	15	1			
	7	720	1764	1624	735	175	21	1		
	8	5040	13068	13132	6769	1960	322	28	1	
	9	40320	109584	118124	67284	22449	4536	546	36	1

(k+1)! P(T(n) = k)

		n								
		1	2	3	4	5	6	7	8	9
	1	0.500								
	2	0.167	0.167							
	3	0.083	0.125	0.042						
	4	0.050	0.092	0.050	0.008					
k	5	0.033	0.069	0.049	0.014	0.001				
	6	0.024	0.054	0.045	0.017	0.003	0.000			
	7	0.018	0.044	0.040	0.018	0.004	0.001	0.000		
	8	0.014	0.036	0.036	0.019	0.005	0.001	0.000	0.000	
	9	0.011	0.030	0.033	0.019	0.006	0.001	0.000	0.000	0.000

P(T(n) = k)

***Figure 3**: P(T(n) = k) at the bottom, (k+1)! P(T(n) = k) at the top*

These probabilities were computed using a small script that generates all (k+1)! permutations of (0, 1, ..., k) and checks, among these permutations, those having a record at position k: for each of these permutations, we computed the total number of records. If N(n, k) denotes the number of such permutations having n records, then P(T(n) = k) = N(n,k) / (k+1)!.

Despite the fact that the above table is tiny, it required hundreds of millions of computations for its production.

15. Miscellaneous Topics

We investigate topics related to time series as well as other popular stochastic processes such as spatial processes.

1. How and Why: Decorrelate Time Series

When dealing with time series, the first step consists of isolating trends and periodicities. Once this is done, we are left with a normalized time series, and studying the auto-correlation structure is the next step, called model fitting. The purpose is to check whether the underlying data follows some well-known stochastic process with a similar auto-correlation structure, such as ARMA processes, using tools such as Box and Jenkins. Once a fit with a specific model is found, model parameters can be estimated and used to make predictions.

A deeper investigation consists of isolating the auto-correlations to see whether the remaining values, once decorrelated, behave like white noise, or not. If departure from white noise is found (using a few tests of randomness), then it means that the time series in question exhibits unusual patterns not explained by trends, seasonality or auto correlations. This can be useful knowledge in some contexts such as high frequency trading, random number generation, cryptography or cyber-security. The analysis of decorrelated residuals can also help identify change points and instances of slope changes in time series, or reveal otherwise undetected outliers.

So, how does one remove auto-correlations in a time series?
One of the easiest solution consists at looking at deltas between successive values, after normalization. Chances are that the auto-correlations in the time series of differences X(*t*) - X(*t*-1) are much smaller (in absolute value) than the auto-correlations in the original time series X(*t*). In the particular case of true random walks (see Figure 1), auto-correlations are extremely high, while auto-correlations measured on the differences are very close to zero. So if you compute the first order auto-correlation on the differences, and find it to be statistically different from zero, then you know that you are not dealing with a random walk, and thus your assumption that the data behaves like a random walk is wrong.

Auto correlations are computed as follows. Let X = X(*t*), X(*t*-1), ... be the original time series, Y = X(*t*-1), X(*t*-2), ... be the lag-1 time series, and Z = X(*t*-2), X(*t*-3), ... be the lag-2 time series. The following easily generalizes to lag-3, lag-4 and so on. The first order correlation is defined as correl(X, Y) and the second order correlation is defined as correl(X, Z). Auto-correlations decrease to zero in absolute value, as the order increases.

While there is little literature on decorrelating time series, the problem is identical to finding principal components among X, Y, Z and so on, and the linear algebra framework used in PCA can also be used to decorrelate time series, just like PCA is used to decorrelate variables in a traditional regression problem. The idea is to replace X(*t*) by (say) X(*t*) + *a* X(*t*-1) + *b* X(*t*-2) and choose the coefficients *a* and *b* to minimize the absolute value of the first-order auto-correlation on the new series. However, we favor easier but more robust methods, for instance looking at the deltas X(*t*) - X(*t*-1), as these methods are not subject to over-fitting yet provide nearly as accurate results as exact methods.

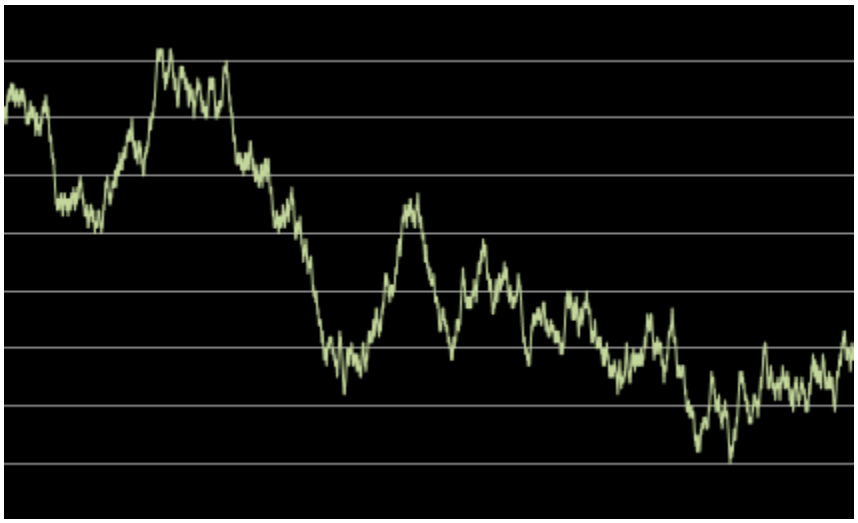


***Figure 1**: Auto-correlations in random walks are always close to +1*

Example

In figure 2, we simulated an auto-correlated time series as follows: X(*t*+1) = X(*t*) + U(*t*) where U(*t*) are independent uniform deviates on [-0.5, 0.5]. The resulting time series is a random walk (with no trend and no periodicity) with a lag-1 auto-correlation of 0.99 when measured on the first 100 observations. The lag-1 auto-correlation measured on the deltas (blue curve) of decorrelated observations is 0.00.

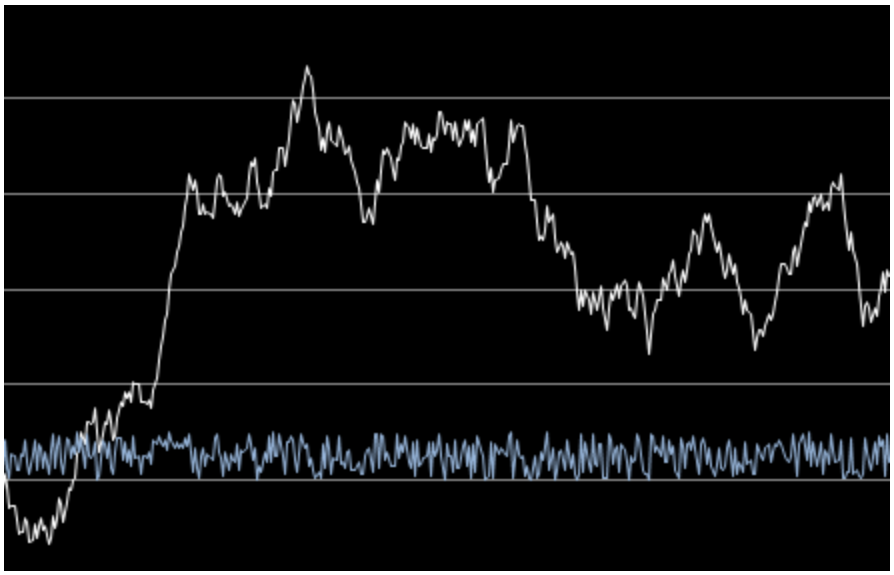


***Figure 2**: original (white) and decorrelated (blue) time series*

2. A Weird Stochastic-Like, Chaotic Sequence

This problem is not related to the previous one. It offers a nice sandbox for data scientists, new research material for people interested in probability theory, and an extremely challenging problem for mathematicians and number theorists, who have worked on very similar problems for more than a hundred years, without finding a solution to this day. Here we provide a quick overview on this subject.

We are interested in the following series:

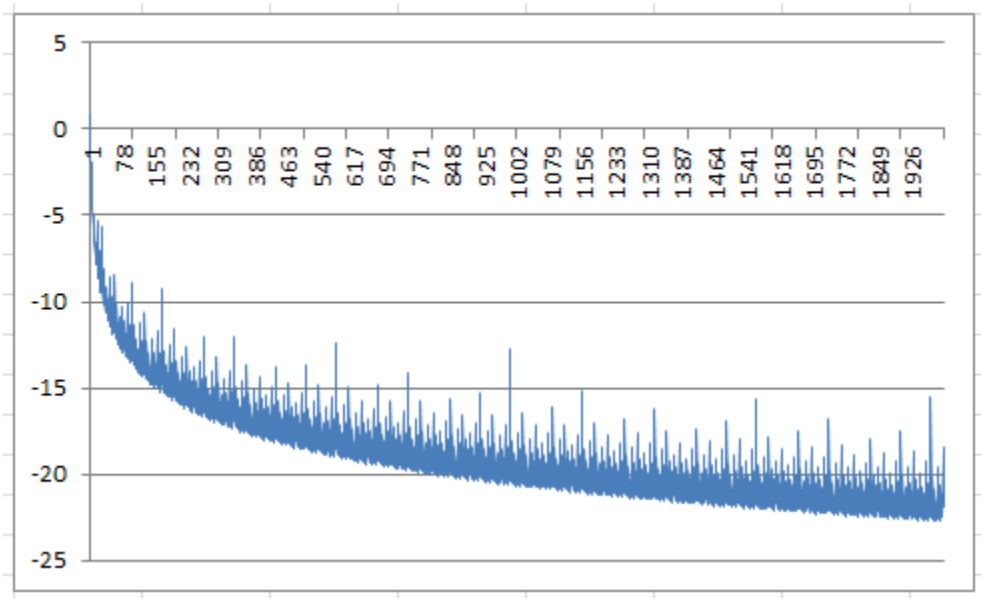
$$f(x) = \sum_{k=1}^{\infty} \frac{1}{k^a \cdot h(kx)}, \text{ with } h(y) = y - \lfloor y \rfloor,$$

where *a* is a parameter greater than 1, *x* is a positive real number, and the brackets represent the integer part function. This series is of course divergent if *x* is a rational number. For which values of the parameter *a* and (irrational) number *x* is this series convergent? The problem is closely related to the convergence of the Flint Hills series, see also here. Convergence depends on how well irrational numbers can be approximated by rational numbers, and in particular, on the irrationality measure of *x*.

If you replace the *h*(*kx*)'s by independent random variables uniformly distributed on [0, 1], then the series represents a random variable with infinite expectation, regardless of *a*. Its median might exist, depending on *a* (if *X* is uniform on [0, 1] then 1/*X* has a median equal to 2, but its expectation is infinite.) For an irrational number *x*, the sequence { *kx* } is equidistributed modulo 1, thus replacing *h*(*kx*) by a random variable uniformly distributed on [0, 1], seems to make sense at first glance, to study convergence. It amounts to reshuffling the terms and adding some little noise that should not impact convergence, or does it? The key here is to study the extreme value

distribution (see chapter 14): when and how close a uniform deviate approaches 0 over a large number of terms) and how well these extrema (corresponding to spikes in the chart below) are dampened by the factor k^a, as the number of terms is increasing.

For the non-random case, it seems that we have convergence if *a* is equal to 3 or higher. What about *a*= 2? No one knows. Below is a chart showing the first 2,000 terms of the series (after taking the logarithm), when *a* = 3 and *x* = SQRT(2).



It shows that the terms in the series rapidly approach zero (the logarithm being below -20, most of the time after the first 1,500 terms), yet there are regular spikes, also decreasing in value, but these spikes are what will make the series either converge or diverge, and their behavior is governed by the parameter *a*, and possibly by *x* as well (*x* is assumed to be irrational here.)

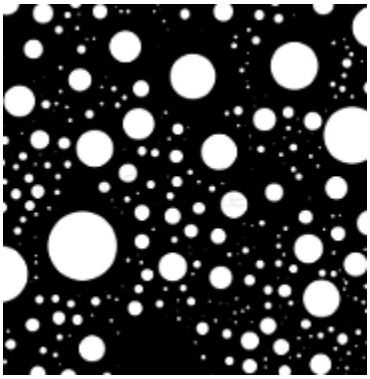
Exercise
In the random case, replace the uniform distribution on [0, 1] by a uniform distribution on [-1, 1]. Now the series has positive and negative terms. How does it impact convergence?

3. Stochastic Geometry, Spatial Processes, Random Circles: Coverage Problem

This should remind you of your probability classes during college years. Can you solve this problem in 30 minutes? This could make for an interesting job interview question.

Problem
Points are randomly distributed on the plane, with an average of *m* points per unit area. A circle of radius *R* is drawn around each point. What is the proportion of the plane covered by these (possibly overlapping) circles?

What if circles can have two different radii, either *R* = *r*, or *R* = *s*, with same probability? What if *R* is a random variable, so that we are dealing with random circles? Before reading further, try to solve the problem yourself.



Solution
The points are distributed according to a Poisson point process of intensity *m*. The chance that an arbitrary point *x* in the plane is not covered by any circle, is the chance that there is zero point from the process, in a circle of radius *R* centered at *x*. This is equal to exp(-*m* πR^2). Thus the proportion of the plane covered by the circles is

$$p(m, R) = 1 - \exp(-m\pi R^2)$$

If circles have radii equal to *r* or *s*, it is like having two independent (superimposed) Poisson processes, each with intensity *m* / 2, one for each type of circle. The chance that *x* is not covered by any circle is thus a product of two probabilities,

$$1 - p(m, R) = \exp\left(-\frac{m\pi r^2}{2}\right) \times \exp\left(-\frac{m\pi s^2}{2}\right) = \exp\left(-m\pi \cdot \frac{r^2 + s^2}{2}\right)$$

If *R* is a positive random variable and *E* denotes its expectation, then the general solution is

$$p(m, R) = 1 - \exp\left(-m\pi E[R^2]\right)$$

You can easily simulate a large number of these circles over a broad area, and then, pick up 1,000 random points and see how many of them are covered by at least one circle, to check whether your solution is correct or not. You can also use these simulations to get an approximation for exp(-π).

Application
The formula for *p*(*m*, *R*) and confidence intervals obtained for its value via simulations under the assumption of pure randomness (Poisson process), can be used to check if a process is really random. For instance, are Moon's craters spread randomly? They

might not, if large meteorites break up in small pieces before impact, resulting in clustered craters. In that case, the area covered by the (overlapping) craters might be smaller than theoretically expected.

4. Additional Reading

The following topics have not been included in this book, but are accessible via the following links:

- Mars Craters: An Interesting Stochastic Geometry Problem
- The Art and (Data) Science of Leveraging Economic Bubbles
- Original Pattern Found in the Stock Market
- Formula to Compute Digits of SQRT(2) - Application to lottery systems
- Simulation of Cluster Processes Evolving over Time - With video
- Twin Points in Point Processes

16. Exercises

These exercises, many with hints to solve them, were initially part of Chapter 9. We moved them here as they cover topics discussed in several chapters in this book.

[1] Randomly pick up 20 numbers *b*(1) ... *b*(30) in {0, 1, 2}. Compute

$$x = \sqrt{b(1) + \sqrt{b(2) + \sqrt{b(3) + \cdots + \sqrt{b(20)}}}}$$

Then compute the first 20 digits of *x*, denoted as *a*(1) ... *a*(20), in the nested square root system. Do we have *a*(1) = *b*(1), *a*(2) = *b*(2), and so on? Repeat this procedure with 10,000 numbers. Are the *a*(*n*)'s and *b*(*n*)'s always identical? For numbers with the last *b*(*n*)'s all equal to zero, you may run into problems.

---

[2] Create a summary table of all the number representation systems discussed here. For each system, include the parameters (base *b* for decimal system), sub-cases (*b* integer versus not integer), the equilibrium and digits distribution as well as their domain, the functions *g*(*x*), *h*(*x*), and *f*({*a*(*n*)}), constraints on the seed (for instance *x* must be in [0, 1]) and typical examples of bad seeds.

---

[3] Compute the digits of Pi, 3/7, and SQRT(2) in base π, and their empirical distribution. Compute at least 30 correct digits (for larger *n*, you may need to use high precision computing, see chapter 8.) Use a statistical test to assess whether these numbers have the same digits distribution, and the same auto-correlation structure, in base π. The exact distribution, assuming that these numbers are good seeds in the base π system, can be derived using the stochastic integral equation P(*X* < *y*) = P(*g*(*X*) < *y*) to get the equilibrium distribution for *x*(*n*), and then derive the digits distribution for *a*(*n*). You can do the same for base 10. Base 10 is actually easier since the theoretical distributions are known to be uniform, and there is no auto-correlation (they are equal to zero) and no need for high precision computing if you use tables of pre-computed digits, instead of the iterative algorithm mentioned in this article. Yet, even in base 10, you can still test whether the auto-correlations are zero, to confirm the theoretical result.

---

[4] This is a difficult problem, for mathematicians or computer scientists. Study the system defined by

$$g(x) = c^x - \lfloor c^x \rfloor, \text{ with } h(x) = \lfloor c^x \rfloor$$

Part I. Which values of *c* and *x,* with *x* in [0, 1], yield a chaotic system? If *c* is big enough, the system seems chaotic, otherwise it is not. What is the threshold (for *c*) to induce full chaos for most seeds? Show that auto-correlations between *a*(*n*+1) and *a*(*n*) are high for good seeds, even for *c* = 4, but are decreasing as *c* is increasing.

Part II. Answer the following questions:

- What is the distribution of the digits for this system, for a good seed, depending on c?
- Are all the digits always in {0, 1,..., INT(c)} ?
- Can you find the equilibrium distribution, and the function $x = f(\{a(n)\})$?

This is known as the iterated exponential map. Note that $g(x)$ is sometimes denoted as c^x mod 1. See exercise 10.4.11 in the book "Non Linear Dynamics and Chaos", by Steven Strogatz (2nd Edition, 2015.)

---

[5] This problem is similar to the previous one, but a little less difficult. Study the system defined by

$$g(x) = 1/x - \lfloor 1/x \rfloor, \text{ with } h(x) = \lfloor 1/x \rfloor$$

with the seed x in [0, 1]. Show that to get a chaotic system, the seed must be an irrational number. In that case, the digits can be any small or large positive integer (there is no upper bound), but most of the time, that is, for most n's, $a(n)$ is a small integer. Indeed, what is the digits distribution? What about the equilibrium distribution? How is this system related to continued fractions? Are the $x(n)$'s auto-correlated?

Solution: The equilibrium distribution, solution of the stochastic integral equation, is given by

$$P(X < y) = \frac{\log(1+y)}{\log 2}, \text{ with } 0 \le y \le 1.$$

You can check it out by plugging that distribution in the stochastic integral equation $P(X < y) = P(g(X) < y)$. To find out how to discover such a solution using empirical (data science) methods, read section 4 of chapter 7.The digits distribution is known as the Gauss-Kuzmin distribution. Besides rational numbers, an example of bad seed is x = (1 + SQRT(5)) / 2, with all the digits equal to 1.

---

[6] For the same g and h as in the previous exercise, what is the function $x = f(\{a(n)\})$? Can the sequence $\{a(n)\}$ of digits be arbitrary? Is it possible to find a number x (of course, it would be a bad seed) such that $a(n) = n$ for all n?

Solution: This system is actually the continued fractions system, thus we have

$$x = f(\{a(n)\}) = \cfrac{1}{a(1) + \cfrac{1}{a(2) + \cfrac{1}{a(3) + \cdots}}}$$

To answer the last question, try x = 0.697774657964008. The first digits (up to n = 11) are 1, 2, 3, ..., 11. Interestingly, if you allow the digits not to be integers, it opens the possibilities. For instance, $x = f(\{a(n)\})$ with $a(n) = 1/n$, yields $x = (\pi - 2)/2$. While this is a well-known result, it is not a valid representation in the standard continued fraction system (exercise: compute the digits of that number in the continued fraction system;

obviously, they will all be integers.) It is like allowing a digit in the base 10 system to not be an integer between 1 and 9.

[7] Equivalence between systems. This exercise is based on the exact formula available for the logistic map. If {($x(n)$)} is the sequence generated in base 2 with a given seed x, then the sequence {$y(n)$} with $y(n) = \sin^2(\pi x(n))$, corresponds to the standard logistic map with seed $y = \sin^2(\pi x)$. See exercise 10.3.10, in the book "Non Linear Dynamics and Chaos" by Steven Strogatz (2nd Edition, 2015.)

Conversely, if {($y(n)$)} is the sequence generated in the logistic map with a given seed y, then the sequence {$x(n)$)} (with $x(n)$ equal to either

- arcsin(SQRT($y(n)$)) / π, or
- 1-arcsin(SQRT($y(n)$)) / π

depending on whether the n^{th} digit of x = arcsin(SQRT(y)) / π is equal to 0 or 1 in base 2), corresponds to the base 2 system with seed x. These facts could possibly be used to prove that π-3 or π/4 is a good seed in base 2.

[8] About the various distributions attached to bad seeds. If x is a bad seed, the digits distribution can be anything. In any base, show that there is an infinite number of bad seeds that do not even have any digits distribution.

Solution: In base 2, consider the following seed: the first digit is 0. Digits #2 to #4 are all 1's. Digits #5 to #8 are all 0's. Digits #9 to #16 are all 1's. Digits #17 to #32 are all 0's, and so on.

[9] About sequences of bad seeds converging to a good seed, or the other way around. This is about the base 10 system. The bad seed x = 1/2 can be approximated by a sequence of (supposedly) good seeds, for instance $m\pi$ / (2$m\pi$ +1) as m tends to infinity, or by a sequence of bad seeds (m-1)/(2m) where m is a prime number. In the latter case, the exact distribution of the $x(n)$'s is known, for each m, since we are dealing with rational numbers. As m tends to infinity, does the distributions in question converge to the distribution associated with the limit x = 1/2? Likewise, we can use the m first factors of the expansion of π/4 as an infinite product, as successive approximations to π. All these approximations are rational numbers, and thus bad seeds. Does the (known) distributions of {$x(n)$)} attached to these rational numbers (bad seeds) converge to the distribution attached to the limit π, that is, to a uniform distribution on [0, 1], as m tends to infinity?

[10] Prove that if base b is an even integer, n > 3, and x = π/4, then

$$x(n) = -\arcsin(\sin\lfloor b^{n-1}x \rfloor).$$

The solution can be found here.