Randomness and Physics

Formalizing Randomness

Generating PRNs

Random Numbers

Random numbers (or random bits) are a crucial ingredient in many algorithms. For example, all industrial strength primality testing algorithms rely on the availability of random bits. Modern cryptography is unimaginable without randomness.

Since computers are deterministic devices (more or less), it is actually not all that easy to produce random bits using a computer: whatever program we run will produce the same bits if we run it again.

To make matters worse, it is rather difficult to even say exactly what is meant by a sequence of random bits. Of course, intuitively we all know what randomness means, right?

Predictions

Randomness is closely associated with the inability to predict what is going to happen. You roll a die, all 6 outcomes are possible.

And yet, rolling a die 6000 times, one would expect the number of 6’s to be somewhere around 1000, say, in the interval [950, 1050]. Some could even calculate that if we expand the interval to [913, 1087] then the likelihood of hitting it is 0.997.

Flipping Coins

Another often used random bit generator: flipping coins.
How Random Is It?

Is the randomness in a coin-toss real or is it actually confined to just the initial conditions?

Persi Diaconis, a Stanford mathematician and highly accomplished professional magician, supposedly can consistently produce ten consecutive heads flipping a coin – by carefully controlling the initial conditions.

Lava Lamps

Radioactivity is another great source of randomness – except that no one likes to keep a lump of radioactive material and a Geiger-Müller counter on their desk. Solution: keep the radioactive stuff someplace else and get the random bits over the web.

True random bits from www.fourmilab.ch.

Krypton-85

Huge Difference

The last system (and also the lava lamps, see below) is very different from the others: if our current understanding of physics is halfway correct, there is no way to predict certain events in quantum physics, like radioactive decay. It is fundamentally impossible (even if we could establish initial conditions correctly, which we cannot thanks to Herr Heisenberg).

The other, purely mechanical systems such as dice and coins, we encounter deterministic chaos: given sufficiently precise descriptions of the initial conditions, and sufficient compute power, one could in principle compute the outcomes (if we think of them as classical systems).

In principle only, not in practice.

Lorenz Attractor

Here is a famous example discovered by Lorenz in the 1963, in an attempt to study a hugely simplified model of heat convection in the atmosphere.

\[ x' = \sigma(y - x) \]
\[ y' = rx - y - xz \]
\[ z' = xy - bz \]

These are not spatial coordinates, \( x \) stands for the amplitude of convective motion, \( y \) for temperature difference between rising and falling air currents, and \( z \) between temperature in the model and a simple linear approximation.

For certain values of the parameters we get the following behavior.
In the olden days, the RAND Corporation used a kind of electronic roulette 
wheel to generate a million random digits (rate: one per second).

In 1955 the data were published under the title:
*A Million Random Digits With 100,000 Normal Deviates*

"Normal deviates" simply means that the distribution of the random numbers
is bell-shaped rather than uniform. But the New York Public Library shelved
the book in the psychology section.

The RAND guys were surprised to find that their original sequence had several
defects and required quite a bit of post-processing before it could pass muster
as a random sequence. This took years to do.


Incidentally, Noll and Cooper at Silicon Graphics discovered one day that the
pretty lava lamps were completely irrelevant: they could get even better
random bits with the lens cap on (there is enough noise in the circuits to get
good randomness).

Another way to use light, very much unlike the original lava lamp system, is to
exploit an elementary quantum optical process: a photon hitting a
semi-transparent mirror either passes or is reflected.

The Quantis systems was developed at the University of Geneva, the first
practical model was released in 1998.

Note that quantum physics is the only part of physics that claims that the
outcome of certain processes is fundamentally random (which is why Einstein
was never very fond of quantum physics).


http://www.idquantique.com/random-number-generation/quantis-random-number-generator/

### Hilbert’s 6th Problem

Mathematical Treatment of the Axioms of Physics.

The investigations on the foundations of geometry suggest the
problem: To treat in the same manner, by means of axioms,
those physical sciences in which already today mathematics plays
an important part; in the first rank are the theory of probabilities
and mechanics.
Unpredictability

So far, all the examples we have seen are based on the notion of unpredictability in physics: the behavior of some physical systems is so complicated that the only way to determine the state at time \( t \) is “run” the system till time \( t \).

Now suppose you use random bits in some algorithm and you want to prove your algorithm to be correct. Without a formal, mathematical definition you cannot even start with the proof.

And try to convince a proof checker that it’s correct when the first line reads “flip a coin 10000 times.”

Obstructions to Randomness

It is somewhat easier to consider infinite binary sequences \( \alpha \in 2^\omega \) rather than finite ones. Infinity is often an excellent approximation for finiteness.

Let’s turn around for the moment and look at properties that would disqualify a sequence from being random in any intuitive sense of the word. Here are two obvious potential problems.

- **Bias (or skew):** the probability of a 0 is not 1/2.
- **Correlation:** the \( i + 1 \)st bit is not independent from the \( i \)th bit.

Fortunately, we don’t need to achieve perfection in either category: there are algorithms that can turn a slightly biased and/or correlated sequence into an unbiased and uncorrelated one.

Here is a simple though not entirely satisfactory method to eliminate bias due to von Neumann. Dealing with correlated bits is harder, we won’t get involved.

Removing Bias

Suppose we have an imperfect source of random bits (real world sources typically fall into this category) that are already independent but that the probability of a 0 is \( 1/2 + \epsilon \). To eliminate this bias, John von Neumann suggested to following algorithm:

- Read the bits, two at a time.
- Skip 00 and 11.
- For 01 and 10 output the first bit.

The probabilities of all 2-blocks are easily computed since we assume independence:

\[
\begin{align*}
00 & \quad (1/2 + \epsilon)^2 = 1/4 + \epsilon + \epsilon^2 \\
01 & \quad (1/2 + \epsilon)(1/2 - \epsilon) = 1/4 - \epsilon^2 \\
10 & \quad (1/2 - \epsilon)(1/2 + \epsilon) = 1/4 - \epsilon^2 \\
11 & \quad (1/2 - \epsilon)^2 = 1/4 - \epsilon^2
\end{align*}
\]

The resulting sequence has no bias, as needed. Of course, it’s a bit shorter.

Density

It is easy to define the density of a finite binary word \( x \) of length \( n \):

\[
D(x) = 1/n \sum_{i} x_i
\]

But how about infinite sequences?

**Definition (Density)**

Let \( \alpha \in 2^\omega \) and define the **density** of \( \alpha \) up to \( n \) to be \( D(\alpha, n) = \frac{1}{n} \sum_{i<n} \alpha_i \).

The **limiting density** of \( \alpha \) is

\[
D(\alpha) = \lim_{n} D(\alpha, n),
\]

Note that there is a huge problem with this definition: limits are precisely defined in analysis, and there is not much reason to assume that this particular limit should exist.

Defining Randomness

All we have so far are various references to common sense and physics. Physics would be fine if we had a coherent, precise theory of the field. We don’t.

Hilbert’s problem number 6 is still open.

Of course, there are excellent physical theories available, in particular relativity theory and quantum theory, but things get dicey when one tries to prove strict mathematical results. In essence, we can only argue relative to some axiomatization.

Still, we can home in on a few central properties of randomness – properties that we feel intuitively to be associated with randomness rather than being able to derive them from first principles. But one has to be careful, intuition is not always reliable.

The Law of Large Numbers

The LoLN says that if we repeat an experiment often, the observed average does in fact converge to the expected value; almost certainly.

For example, for an unbiased coin we should approach the the limiting density of 1/2. Don’t ask what an unbiased coin is.

Also note that we should not expect the averages to be exactly equal to the expectation.

For example, performing a one-dimensional random walk with steps \( \pm 1 \) we should expect to be up to \( O(\sqrt{n}) \) from the origin after \( n \) steps.
A Random Walk

We can look at density not just for single bits but for arbitrary finite blocks $w \in 2^m$: the number of occurrences of a particular block $w$ in $x_1x_2 \ldots x_n$ should approach $n/2^m$.

A good way of visualizing this is to have $\alpha$ trace a path in a de Bruijn graph. Recall that the de Bruijn graph of order $k$ is defined as

$$ B_k = \left\langle 2^k, E \right\rangle $$

$$ E = \{ (au, ub) \mid a, b \in 2, u \in 2^{k-1} \} $$

If we slide a window of width $k$ over $\alpha$ we obtain a sequence of nodes in $B_k$, and the sequence is in fact a path in the graph.

De Bruijn Graph of Order 3

De Bruijn graphs are Hamiltonian, and the Hamiltonian cycles are called de Bruijn sequences: they contain every block of length $k$ exactly once and have length $2^k + k - 1$.

Any finite or infinite sequence of bits (of length at least $k$) traces a path in the graph. If the sequence is unbiased the path hits each node equally often in the limit.

Example

The first 30 bits of the binary expansion of $\sqrt{5}$ produces

$001
dots
010
dots
101
dots
111
dots
100
dots
110$

After 59 bits all edges lie on the path.

In practice, counting blocks up to a certain size is a very good test for randomness ($\chi^2$ test).

Decimation

How about using Roman military traditions to define randomness?

In 1919 Richard von Mises suggested a notion of randomness based on the limiting density of the sequence itself and certain derived subsequences.

The idea is that “reasonable” subsequences of the given sequence should also have limiting density $1/2$.

Definition

An infinite sequence $\alpha \in 2^\omega$ is **Mises random** if the limiting density of any subsequence $(x_{i_j})$ is $1/2$ where the subsequence is selected by a Auswahlregel.

Auswahlregeln

So what on earth is a Auswahlregel, a selection rule?

Intuitively, the following all should have limiting density $1/2$:

$$ x_0, x_1, x_2, \ldots, x_n, \ldots $$

$$ x_{0i}, x_{1i}, x_{2i}, \ldots, x_{ni}, \ldots $$

$$ x_1, x_4, x_7, \ldots, x_{3n+1}, \ldots $$

$$ x_0, x_1, x_4, x_{n1}, \ldots $$

In fact, we might want for any reasonable strictly monotonic function $f : \mathbb{N} \to \mathbb{N}$ that

$$ x_{f(0)}, x_{f(1)}, x_{f(2)}, \ldots, x_{f(n)}, \ldots $$

has limiting density $1/2$. 
Mises’ Definition

The actual definition used by Mises is a bit more technical. He considers functions
\[ \Phi : 2^\ast \to \mathbb{B} \]
and selects \( \alpha_n \) to be in the subsequence if \( \Phi(\alpha[n]) \) holds.
However, there is one big caveat: the function \( \Phi \) must be defined without any knowledge of \( \alpha \); otherwise we can simply pick a subsequence of all 0’s.

Now suppose we have a countable system of Auswahlregeln and our sequence passes all these tests. In other words, for all \( \Phi \) we have
\[ D(\alpha_k) = 1/2. \]
Then \( \alpha \) is Mises-random. One can show that for any countable collection of Auswahlregeln there are always uncountably many sequences that are random in this sense.

Sounds all eminently reasonable.

Kolmogorov-Randomness

Kolmogorov suggested to use incompressibility as a measure of randomness.

Definition
An infinite sequence \( \alpha \in 2^\ast \) is Kolmogorov-random if for some constant \( c: K(\alpha[n]) \geq n – c \) for all \( n \).

So the prefixes \( \alpha[n] \) are algorithmically incompressible with the same constant \( c: K(\alpha[n]) \) is the shortest description of itself. Chaitin’s \( \Omega \) is an example of a sequence that is random in this sense.

Again, incompressibility is very similar in spirit to the notion of randomness: there is no rhyme nor reason, one has to have a full record to reconstruct the sequence.

The Importance of Prefix Complexity

It is important that the notion of incompressibility is based on Chaitin prefix complexity, not the more intuitive Kolmogorov complexity.

Theorem (Martin-Löf)
Let \( f \) be computable such that \( \sum 2^{-f(i)} \) diverges. Then for any \( \alpha \in 2^\ast \) there are infinitely many \( n \) such that \( C(\alpha[n]) < n – f(n) \).

The proof is quite involved. Note that \( f(n) = \log n \) satisfies the hypothesis, so we can infinitely often shorten the description in plain Kolmogorov-Chaitin complexity by a logarithmic amount.

Martin-Löf Randomness

Here is another approach to randomness: construct a collection of test that are supposed to detect non-randomness; a sequence is random if it survives all the tests.

As we will see, computability plays a major role in describing the tests, without it, our tests would rule out all sequences.

It is helpful to think of an infinite sequence \( \alpha \in 2^\ast \) as an infinite branch in the full infinite binary tree \( 2^\ast \).

The initial segments are ordinary finite binary words and we can try to express conditions on \( \alpha \) by placing conditions on prefixes \( \alpha[n] \).

Ville’s Counterexample

Unfortunately, in 1939 J. Ville showed that for any countable system of Auswahlregeln there is always a sequence \( \alpha \) that passes all the tests (i.e., the limiting density is 1/2 for all these subsequences) but that is nonetheless biased towards 1.
More precisely, it was known that a random sequence should have
\[ \limsup_n \sqrt{\frac{2n}{\log \log n}} (D(\alpha, n) - 1/2) = 1 \]
\[ \liminf_n \sqrt{\frac{2n}{\log \log n}} (D(\alpha, n) - 1/2) = -1 \]
and Ville’s example violated the second condition.

Cylinders and Measure

We will need to measure the size of sets \( S \subseteq 2^\ast \).
To this end let \( x \) be a finite word and define the cylinder generated by \( x \) and its measure by
\[
\text{cyl}(x) = \{ \alpha \in 2^\ast \mid x \sqsubseteq \alpha \} \\
\mu(\text{cyl}(x)) = 2^{-|x|}
\]
These cylinders form basic open sets in \( 2^\ast \) and any open set \( S \) can be written as the disjoint union of countably many of them. Hence we can measure \( S \):
\[
S = \bigcup_i \text{cyl}(x_i) \\
\mu(S) = \sum_i 2^{-|x_i|}
\]
### Intuition

Think of $2^\omega$ as the real interval $[0, 1] \subseteq \mathbb{R}$.

Given a binary word $x = x_1 x_2 \ldots x_n$, think of $\text{cyl}(x)$ as a real interval comprising all numbers with binary expansion $0 . x_1 x_2 \ldots x_n 0 x_1 x_2 \ldots$ which is an interval of length $2^{-n}$.

We will try to cover a given set of reals by a collection of intervals of shrinking size (a set of constructive measure zero).

### Example: Density

For example, to weed out sequences with limiting density at least $2/3$ we can use the following test sets:

$$K_n = \bigcup \{ \text{cyl}(x) \mid |x| \geq n \land D(x) \geq 2/3 \}$$

Then any sequence in $K = \bigcap_n K_n$ has limiting density at least $2/3$ and should be eliminated from our pool of potential random sequences.

Of course, we need to perform countably many such tests to make sure that the limiting density is not larger than $1/2 + \varepsilon$ for any $\varepsilon > 0$ (actually, not; see below).

### General Case

More generally, we consider tests analogous to the density test: we want a descending chain of open sets

$$K_n \supseteq K_1 \supseteq K_2 \supseteq \ldots \supseteq K_n \supseteq \ldots$$

where $\mu(K_n) \leq 2^{-n}$ so that these sets are becoming “small” as $n$ increases and their intersection has constructive measure zero.

We eliminate all sequences in this set $K = \bigcap_n K_n$.

For the math guys: $K$ is a $G_\delta$ null set.

Note that it is not so easy to show that the $K_n$'s from the example above are sufficiently small.

### The Catch

We want to declare a sequence $\alpha$ to be random if it survives this kind of test for various choices of $(K_n)$. For example, we would want to apply all possible density tests, for blocks of arbitrary lengths.

Unfortunately, we cannot simply allow arbitrary tests $(K_n)$; if we do, then all sequences are eliminated.

To see why, let $\alpha \in 2^\omega$ arbitrary and define a special test $K_n^\alpha = \text{cyl}(\alpha[n])$. Then $K_n^\alpha = \bigcap_n K_n^\alpha = \{\alpha\}$.

So, if we want to get anything useful out of this, we need to limit the permissible tests $(K_n)$.

### Computability to the Rescue

The key in designing a usable randomness test lies in imposing a computability constraint.

**Definition**

A **sequential test** has the additional property that

$$K = \{ (n,x) \mid x \sqsubset \alpha \in K_n \subseteq 2^* \}$$

is semidecidable.

As a consequence, there are only countably many sequential tests.

And, we certainly can no longer design a test that eliminates an arbitrary sequence $\alpha$ (unless $\alpha$ is computable).

### Applying a Sequential Test

By definition, we can effectively enumerate all the pairs $(n,x)$ in $K = \{ (n,x) \mid x \sqsubset \alpha \in K_n \}$. 

What does it mean for $\alpha$ to pass this test?

We need $\alpha \in \bigcap_n K_n$ where

$$K_n = \bigcup_{(n,x) \in K} \text{cyl}(x)$$

So for every $n$, we have to find an $x$ such that $(n,x) \in K$ and $x \sqsubset \alpha$.

Of course this is not effective, even assuming that we have $\alpha$ as an oracle. But it’s not terribly far off.
Taking our definitions at face value, we would have to check all possible sequential tests to check for randomness.

Here is an amazing result that shows that in essence we only need to deal with a single test. The proof uses the existence of a universal Turing machine and is not particularly difficult.

**Theorem (Universal Test)**

There is a universal sequential test $U$ such that for any sequential test $K$ we have $K_{n+c} \subseteq U_n$ for some constant $c$.

---

**Definition**

An infinite sequence $\alpha$ is **Martin-Löf random** if it passes a universal sequential test.

This definition is very strongly supported by the empirical fact that any practical test of randomness in ordinary probability theory can be translated into a sequential test. So, we are just dealing with all of these tests at once (plus all conceivable others).

As it turns out, our last two approaches produces the same notion of randomness.

**Theorem**

A sequence is Martin-Löf random iff it is Kolmogorov-random.

---

**Exercise**

Give a detailed proof that limiting density can be checked by a suitable test set. In particular, make sure that semi-decidability holds.

**Exercise**

Show how to check for the frequencies of blocks of arbitrary finite length in a sequential test.

**Exercise**

Show that every Martin-Löf random sequence fails to be computable. Assume a random sequence is computable and show how to construct a test that rejects it.

**Exercise**

Show that there are uncountably many Martin-Löf random sequences (in fact, they form a set of measure 1).

---

**Pseudo-Randomness**

Anyone attempting to produce random numbers by purely arithmetic means is, of course, in a state of sin.

*John von Neumann*

In the real world, one often makes do with a pseudo-random number generator (PRNG) based on iteration: the pseudo-random sequence is the orbit of a some initial element under some function.

\[
\begin{align*}
x_0 &= \text{chosen somehow, at random :-) } \\
x_{n+1} &= f(x_n)
\end{align*}
\]

where $f$ is easily computable, typically using arithmetic and some bit-plumbing.
The Lasso

Of course, we are taking a huge step away from real randomness here; this sequence would perish miserably when exposed to a Martin-Löf test. The function \( f \) typically operates on some finite domain such as 64-bit words. Every orbit necessarily looks like so:

![Diagram](image)

So we can only hope to make \( f \) fast and guarantee long periods.

Still, there are many applications where this type of pseudo-randomness is sufficient.

One has to be very careful with cryptography, though!

The Seed

Needless to say, running a PRNG twice with the same seed \( x_0 \) is going to produce exactly the same "random" sequence.

This can be a huge advantage, because it makes computations that are based on the random numbers reproducible (important for debugging and verification).

More generally, if we are willing to pay for a truly random seed, we would hope that the iterative PRNG would amplify the randomness: we provide \( m \) truly random bits and get back \( n \) high-quality pseudo-random bits where \( n \gg m \).

Hence PRNGs reduce the need for truly random bits but does not entirely eliminate them.

The Real Challenge

As von Neumann kindly pointed out, we’re hosed, no matter what we do.

So the real question is this: how simple and easy-to-compute can we make our sinful function \( f \) and still get pseudo-random numbers that are sufficiently good to drive certain algorithms?

In the worst case, we could resort to actual quantum randomness, but that is expensive in many ways.

As it turns out, often we can get away with murder.

Linear Congruential Generator

A typical example: a simple affine map modulo \( m \).

\[
x_{n+1} = ax_n + b \mod m
\]

The trick here is to choose the proper values for the coefficients. Can be found in papers and on the web.

A choice that works reasonably well is

\[
a = 1664525 \quad b = 1013904223 \quad m = 2^{32}
\]

Note that a modulus of \( 2^{32} \) amounts to unsigned integer arithmetic on a 32-bit architecture, so this is implementation-friendly.

Inverse Congruential Generator

Choose the modulus \( m \) to be a prime number and define the patched inverse of \( x \) to be

\[
\mathcal{F} = \begin{cases} 
0 & \text{if } x = 0, \\
 x^{-1} & \text{otherwise}.
\end{cases}
\]

Then we can define a pseudo-random sequence by

\[
x_{n+1} = a \mathcal{F} x_n + b \mod m
\]

Computing the inverse can be handled by the extended Euclidean algorithm.

Again, it is crucial to choose the proper values for the coefficients.

Multiplicative Congruential Generator

Omit the additive offset and use multiplicative constants only.

If need be, use a higher order recurrence.

\[
x_n = a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_k x_{n-k} \mod m
\]

For prime moduli one can achieve period length \( m^k - 1 \).

This is almost as fast and easy to implement as LCG (though there is of course more work involved in calculating modulio a prime).

Note, though, that there is more state: we need to store all of \( x_{n-1}, x_{n-2}, \ldots, x_{n-k} \).
Mersenne Twister

Fairly recent (1998) method by Matsumoto and Nishimura, seems to be the tool of choice at this point.

- Has huge period of $2^{19937} - 1$, a Mersenne prime.
- Is statistically random in all the bits of its output (after a bit of post-processing).
- Has negligible serial correlation between successive values.
- Only statistically unsound generators are much faster.

The method is very clever and not exactly obvious.

The algorithm works on bit-vectors of length $w$ (typically 32 or 64).

Let $k$ be the degree of the recursion, and choose $1 \leq m < k$ and $0 \leq r < w$.

The MT Recurrence

So we are trying to generate a sequence of bit-vectors $x_i \in 2^w$.

Define the join $(x, y)$ of $x, y \in 2^w$ to be the first $w - r$ bits of $x$ followed by the last $r$ bits of $y$.

$$\text{join}(x, y) = (x_1, x_2, \ldots, x_{w-r}, y_{w-r+1}, \ldots, y_w).$$

We can use the join operation to define the following recurrence:

$$x_{n+k} = x_{n+m} \cdot \text{join}(x_n, x_{n+1}) \cdot A.$$

Here $A$ is a sparse companion-type matrix that makes it easy to perform the vector-matrix multiplication.

Companion Matrix

The $w \times w$ matrix $A$ has the following form:

$$A = \begin{pmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
a_{w-1} & a_{w-2} & a_{w-3} & \cdots & a_0
\end{pmatrix}.$$  

Note that $z : A$ is not really a vector-matrix operation and can be handled in $O(w)$ steps.

This is just a convenient way to describe the necessary manipulations.

Good Parameters

Here is one excellent choice for the parameters:

$$w = 32 \quad k = 624 \quad m = 397 \quad r = 31$$

and the $A$ matrix is given by

$$a = 0x9908B0DF$$

which hex-number represents the entries in the last row of $A$.

Recall that the recurrence determines $x_{n+k}$ in terms of $x_{n+m}$, $x_n$ and $x_{n+1}$, so we need to store a bit of state: $x_n, \ldots, x_{n+k-1} = x_{n+623}$.

So we need to store 624 words, not too bad.

A Miracle

This particular choice of parameters achieves the theoretical upper bound for the period:

$$2^{w-k-r} = 2^{19937} \approx 4.315 \times 10^{6001}.$$  

After a little bit of post-processing of the sequence $(x_i)_{i \geq 0}$, this method produces very high quality pseudo-random numbers, and is not overly costly.