UCT

(Pseudo) Randomness

Klaus Sutner
Carnegie Mellon University
Spring 2022

The Roots: Gambling

Probability theory is actually one of the older theories in math, inspired by the pressing need to develop sound gambling strategies.

The first major contribution appears to be by Gerolamo Cardano, the *Liber de Ludo Aleae*, around 1564.

Later Galileo, Fermat and Pascal developed matters further.

Predictions

What is the big question about dice? The inability to predict the outcome of the next roll, even though one has observed a number of previous rolls. This is the core of randomness.

And yet, rolling a die 6000 times, one would expect the number of 6s 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.

Random Algorithms

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.

There is good reason to believe that a plausible formalization of the notion of "feasible computation" should allow for randomness. Alas, there is a bit of an issue:

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

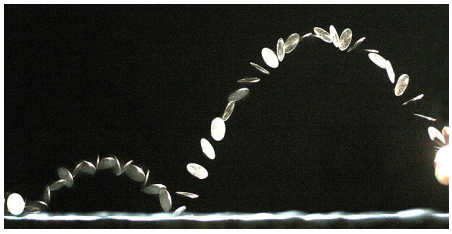
Determinism

Von Neumann is alluding to the big conundrum: digital computers are deterministic devices (more or less), and no such device can produce truly random bits (at least not if we identify a computer with a deterministic Turing machine).

As we will see, randomness and computability are not compatible. To understand why this is so, we need a precise mathematical definition of randomness. Intuition based on experience of physical phenomena is critical, but not sufficient.

And we will have to figure out how to fake randomness, how to produce bits that actually violate our definition (and in fact ridiculously so), but are sufficiently close to random to work just fine in algorithmic applications.

Flipping Coins



Another often used random bit generator: flipping coins. This is better for our purposes since we get random bits.

How Random Is It?

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

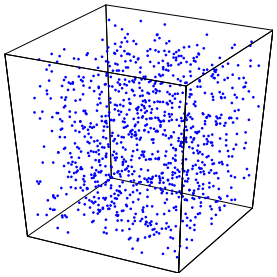
Persi Diaconis, a Stanford mathematician and highly accomplished professional magician, supposedly can consistently produce ten consecutive heads flipping a coin – by carefully controlling the initial conditions.

There is a great informal book on probability by Diaconis and Skyrms, *Ten Great Ideas About Change*.

Lava Lamps

Krypton-85

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.

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 exactly, which we cannot thanks to Herr Heisenberg.

It is most remarkable that we can still make statistical assertions about the behavior of large collections of radioactive atoms: we can predict when $1/2$ of them will have decayed.

U-238	4.5 bill
U-235	700 mill
U-234	25,000

Still, for any individual atom we have no information.

Deterministic Chaos

Other, purely mechanical systems such as dice and coins, are perfectly predictable at least in principle: if we know the initial conditions, then the laws of physics determine a unique trajectory and we can pre-compute the ultimate result.

Except, we can't: there are deterministic systems given by, say, non-linear differential equations with unique solutions, where the tiniest change in initial conditions causes the trajectories to diverge exponentially. As Poincaré realized in the late 1800s:

> . . . et nous avons des phénomènes aléatoires . . .

". . . and we have random phenomena . . . ". This was before relativity theory or quantum physics, everyone was still thinking in terms of a clockwork universe.

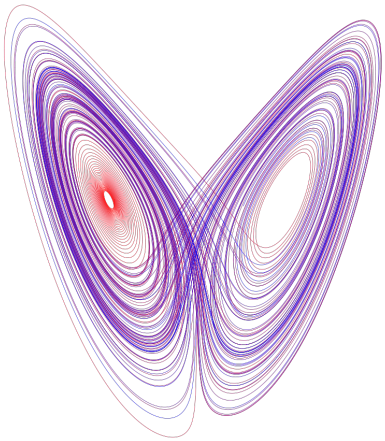
Lorenz Attractor

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.



Pre-History

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.

Available at http://www.rand.org/publications/classics/randomdigits.

Cloudflare

Needs lots of high-quality randomness for their internet services. They harvest their random bits from various physical sources:

- A wall of lava lamps in San Francisco.
- A chaotic pendulum in London.
- A lump of radioactive material in Singapore.

The raw random bits are then fed into a cryptographically secure pseudo-random number generator (CSPRNG) that generates more good random bits.

Fiat Lux

Incidentally, Noll and Cooper at Silicon Graphics were the first to use lava lamps as random number generators.

They 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).

The "Entropy Wall" at Cloudflare, by contrast, takes pictures of the whole wall, and then uses a clever hash function to turn the huge collection of pixels into a large random number.

Getting Serious

A more ambitious way to use light, is to exploit an elementary quantum optical process: a photon hitting a semi-transparent mirror, and either passes through or is reflected. Schrödinger already got great mileage out of this idea.

The Quantis systems was developed at the University of Geneva, the first practical model was released in 1998, http://www.idquantique.com/

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).

The Magic Device

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

Quantis TRNG

Features

- True quantum randomness
- High bit rate of 4Mbits/sec (up to 16Mbits/sec for PCI card)
- Low-cost device (1000+ Euros)
- Compact and reliable
- USB or PCI, drivers for Windows and Linux

Applications

- Numerical Simulations
- Statistical Research
- Lotteries and gambling
- Cryptography

The Central Issue

Producing random bits with a Quantis card is great, but there remains a nagging problem: what are the actual mathematical properties of randomness and probability? For example, what do we need to prove that a probabilistic algorithm produces certain performance guarantees? Could we execute such a proof on a theorem prover? Just try to tell the prover "flip a coin 10000 times."

It might be tempting to start with quantum physics and then derive all properties from there, but that is tricky since physics is not at the level of axiomatization that has become standard in mathematics. Also, it would be nice to have some direct, mathematical framework that pins down randomness, so that physics provides an example, rather than being the source of all concepts.

Hilbert's 6th Problem

Hilbert was the first to point out that an axiomatization of probability was needed, a task solved by Kolmogorov in 1933 (it's remarkable that this took three decades).

> 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.

This is in reference to Hilbert's seminal "Grundlagen der Geometrie" from 1899, the first modern axiomatization of a mathematical area. Well worth reading even today.

Unpredictability

Unpredictability is the key idea in randomness: suppose we are at time t_0, and we want to understand the behavior of some physical system S at time $t > t_0$. In many cases, S is so complicated that the only way to determine its state at time t is to "run" the system till time t. But we cannot run a simple computation that will determine the state of S at time t.

This phenomenon is often referred to as computational irreducibility.

Note that this irreducibility requirement bodes ill for computational approaches, if we had a way to compute truly random that would presumably ruin irreducibility. Of course, in many cases we can get by just making the computation sufficiently messy.

Towards Infinity

This may sound counterintuitive, but it is actually a little easier to try to define a random infinite binary sequences $\alpha \in \mathbf{2}^\omega$ instead of a single random bit, or a finite sequence (as in Kolmogorov-Chaitin complexity). Infinite sequences allow us to talk about limits, properties that emerge as one considers arbitrarily large initial segments.

For example, a finite binary word x of length n naturally has the density

$$D(x) = \frac{1}{n}\sum_i x_i$$

We can lift this to infinite sequences as follows.

Definition (Density)

Let $\alpha \in \mathbf{2}^\omega$ and define the density of α up to n to be $D(\alpha, n) = \frac{1}{n}\sum_{i<n} \alpha_i$. The limiting density of α is $D(\alpha) = \lim_n D(\alpha, n)$.

Obstructions to Randomness

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 in any technical sense. We will avoid these issues.

So what makes a sequence random? Based on intuition, we can come up with a wishlist.

- Bias (or skew): we want $D(\alpha) = 1/2$.
  Actually, we want this to be true not just for single bits, but for all blocks of length k, they should have limiting frequency 2^{-k}.
- Correlation: the $i+1$st bit in α is not independent from the ith bit.

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 limiting density of $1/2$. Don't ask what an unbiased coin is.

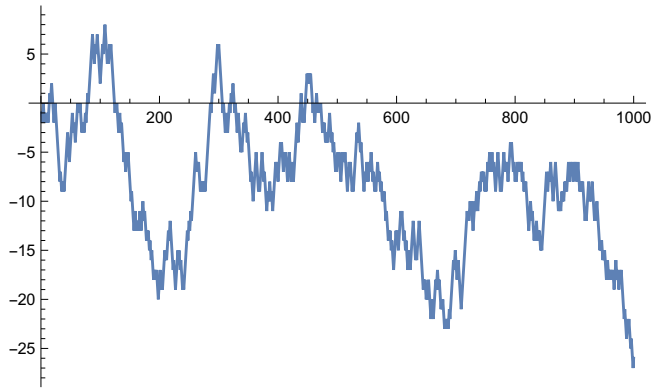
Also note that we should not expect the averages to be exactly equal to the expectation.

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

Exercise

What is an unbiased coin in the physical world?

A Random Walk



Aside: Fixing Things

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. It destroys bits, but removes slight 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 $1/2 + \varepsilon$ for some small, positive ε.

Removing Bias

- 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:

00	$(1/2+\varepsilon)^2$
$01, 10$	$(1/2+\varepsilon)(1/2-\varepsilon)$
11	$(1/2-\varepsilon)^2$

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

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 $\alpha \in \mathbf{2}^\omega$ 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 $1/2$:

$$x_0, x_1, x_2, \ldots, x_n, \ldots$$
$$x_0, x_2, x_4, \ldots, x_{2n}, \ldots$$
$$x_1, x_4, x_7, \ldots, x_{3n+1}, \ldots$$
$$x_0, x_1, x_4, \ldots, x_{n^2}, \ldots$$

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 $1/2$.

Ville's Counterexample

Unfortunately, in 1939 J. Ville showed that for any countable system of Auswahlregeln there is always a sequence α 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

$$\limsup_n \sqrt{\frac{2n}{\log\log n}}\left(D(\alpha, n) - 1/2\right) = 1$$

$$\liminf_n \sqrt{\frac{2n}{\log\log n}}\left(D(\alpha, n) - 1/2\right) = -1$$

and Ville's example violated the second condition.

Kolmogorov-Randomness

Kolmogorov suggested to use incompressibility as a measure of randomness.

Definition

An infinite sequence $\alpha \in \mathbf{2}^\omega$ is Kolmogorov-random if for some constant c: $C(\alpha[n]) \geq n - c$ for all n.

So the prefixes $\alpha[n]$ are algorithmically incompressible with the same constant c; $\alpha[n]$ is the shortest description of itself. Chaitin's Ω 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 $\alpha \in \mathbf{2}^\omega$ there are infinitely many n such that $K(\alpha[n]) < n - f(n)$.

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, again based on a collection of tests 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 $\alpha \in \mathbf{2}^\omega$ as an infinite branch in the full infinite binary tree $\mathbf{2}^\star$.

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

Cylinders and Measure

We will need to measure the size of sets $S \subseteq \mathbf{2}^\omega$.

To this end, write $x \sqsubset \alpha$ to indicate that the finite string x is a prefix of the infinite string α and define the cylinder generated by x and its measure by

$$\mathsf{cyl}(x) = \{\, \alpha \in \mathbf{2}^\omega \mid x \sqsubset \alpha \,\}$$
$$\mu(\mathsf{cyl}(x)) = 2^{-|x|}$$

These cylinders form basic open sets in $\mathbf{2}^\omega$ and any open set S can be written as the disjoint union of countably many of them. Hence we can measure S:

$$S = \bigcup_i \mathsf{cyl}(x_i)$$
$$\mu(S) = \sum_i 2^{-|x_i|}$$

Intuition

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

Given a binary word $x = x_1 x_2 \dots x_n$, think of $\mathsf{cyl}(x)$ as a real interval comprising all numbers with binary expansion

$$0.x_1 x_2 \dots x_n \mid z_0 z_1 z_2 \dots$$

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 family of test sets:

$$K_n = \bigcup \{\, \mathsf{cyl}(x) \mid |x| \geq n \wedge D(x) \geq 2/3 \,\}$$

Being in K_n is not a problem, we can fix the imbalance with later bits. But being in $K = \bigcap_n K_n$ is: any such sequence 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

$$K_0 \supseteq K_1 \supseteq K_2 \supseteq \dots \supseteq K_n \supseteq \dots$$

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_δ null set.

Note that it is not so easy to show that the K_ns from the example above are sufficiently small.

The Catch

We want to declare a sequence α 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 $\alpha \in \mathbf{2}^\omega$ arbitrary and define a special test $K_n^\alpha = \mathsf{cyl}(\alpha[n])$. Then $K^\alpha = \bigcap K_n^\alpha = \{\alpha\}$.

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 is a test (K_n) such that the following set is semidecidable:

$$\mathcal{K} = \{\, (n, x) \mid \mathsf{cyl}(x) \subseteq K_n \,\} \subseteq \mathbb{N} \times \mathbf{2}^\star$$

Think of fixing n, then we are enumerating the prefixes of sequences in K_n.

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

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

Applying a Sequential Test

Note that we can recover the K_n from $\mathcal{K}$:

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

In a sense this says that $\mathcal{K}$ is semidecidable (but we have not explained what recursion theory on $\mathbf{2}^\omega$ is).

So what does it mean for α to fail this test? We need $\alpha \in \bigcap K_n$, which means

$$\forall\, n\, \exists\, x \left((n,x) \in \mathcal{K} \wedge x \sqsubset \alpha\right)$$

Of course this is not effective, even assuming that we have α as an oracle. But it's not terribly far off, just Π_2.

Universal Tests

Taking our definitions at face value, we would have to check all possible sequential tests to check for randomness. Since there are universal Turing machines we can collect all tests into a single one.

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.

Hence there are uncountably many random sequences, in fact, they form a set of measure 1. It is also known that every set can be Turing reduced to a random sequence.

Martin-Löf Randomness

Definition

An infinite sequence α is Martin-Löf random if it passes a universal sequential test (and thus all sequential tests).

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 will all of these tests at once (plus all conceivable others).

As it turns out, the last approach produces the same notion of randomness as Kolmogorov-Chaitin style program-size complexity.

Theorem

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

How Bad is Randomness?

The definition may be elegant and right, but, sadly, it does not yield any methods to construct a random sequence. Aux contraire:

Theorem

Any Martin-Löf random sequence fails to be computable.

A typical example of a low-complexity random sequence is Chaitin's Ω which turns out to be Δ_2 in the arithmetical hierarchy. Close, but not computable.

Exercises

Exercise

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

Exercise

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

Generating Randomness

Recall von Neumann's dire warning:

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

True, but in many places it suffices to have nearly random bits, and the effort to get real random ones is just not warranted. "Purely arithmetic means" are often good enough.

One area, though, where things get very tricky is cryptography: we have to deal with an attacker that tries to exploit patterns in our pseudo-random bits. By contrast, there is no attacker in, say, probabilistic primality tests.

Pseudo-Randomness

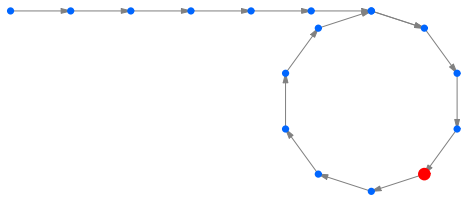
In the real world, one often makes do with a pseudo-random number generator (PRNG) based on iteration: the pseudo-random sequence is the orbit of a some initial element under some function.

$$x_0 = \text{ chosen somehow, at random :-)}$$
$$x_{n+1} = f(x_n)$$

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, perhaps without the transient:



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.

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 (critically 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 reduces 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.

$$x_{n+1} = a\,x_n + b \bmod m$$

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

$$a = 1664525 \qquad b = 1013904223 \qquad m = 2^{32}$$

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

Multiplicative Congruential Generator

Omit the additive offset and use multiplicative constants only.

If need be, use a higher order recurrence.

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \ldots a_k x_{n-k} \bmod m$$

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 modulo a prime).

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

Inverse Congruential Generator

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

$$\overline{x} = \begin{cases} 0 & \text{if } x = 0, \\ x^{-1} & \text{otherwise.} \end{cases}$$

Then we can define a pseudo-random sequence by

$$x_{n+1} = a\,\overline{x_n} + b \bmod m$$

Computing the inverse can be handled by the extended Euclidean algorithm. But note that this is computationally more expensive.

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

Mersenne Twister

Fairly recent (1997) method by Matsumoto and Nishimura, seems to be the tool of choice at this point (used in Sage, Maple, Matlab, GMP ...)[†].

- 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. Here are the key parameters:

w	length of bit-vectors used (typically 32 or 64)
k	order of the recurrence
r, m	$1 \le m < k$ and $0 \le r < w$

[†] There has been some grumbling lately about better methods.

The MT Recurrence

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

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

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

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

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

Here A is a sparse companion-type matrix that makes it easy to perform the vector-matrix multiplication. The arithmetic is over the two-element field $\mathbb{F}_2$.

Companion Matrix

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

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

Note that $z \cdot A$ is not really a vector-matrix operation, it can be handled in $O(w)$ steps. A is just convenient way to describe the recurrence from the last slide.

Good Parameters

Here is one excellent choice for the parameters:

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

and the A matrix is given by

$$\boldsymbol{a} = \text{0x9908B0DF} = 10011001000010001011000011011111$$

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: $x_n, \ldots, x_{n+k-1} = x_{n+623}$.

For our choice of parameters, we need to store 624 words, not too bad.

Initialization

Initialization is non-trivial here, we need k words before the actual algorithm can get started.

A standard method is to pick $x_0 \in \mathbf{2}^w$ (at random) and then define

$$x_i = c \cdot (x_{i-1} \oplus \mathsf{rshft}(x_{i-1}, w-2)) + i$$

where c is again cleverly chosen, typically $c = 1812433253$.

Initial conditions with lots of zeros should be avoided, it takes a while before good randomness appears in the generated sequence.

A Miracle

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

$$2^{wk-r} = 2^{19937} \approx 4.315 \times 10^{6001}$$

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.

Of course, the finite word of length about 4×10^{6001} has very low Kolmogorov-Chaitin complexity: we just specified a program that is much, much shorter.

My Favorite PRNG: ECA 30