MANCHESTER 1824

The University of Manchester

Cockcroft Institute Postgraduate Lectures
Numerical Methods and Lattice Design

Lecture 2: Monte Carlo Methods and Random Sampling

Hywel Owen

School of Physics and Astronomy, University of Manchester



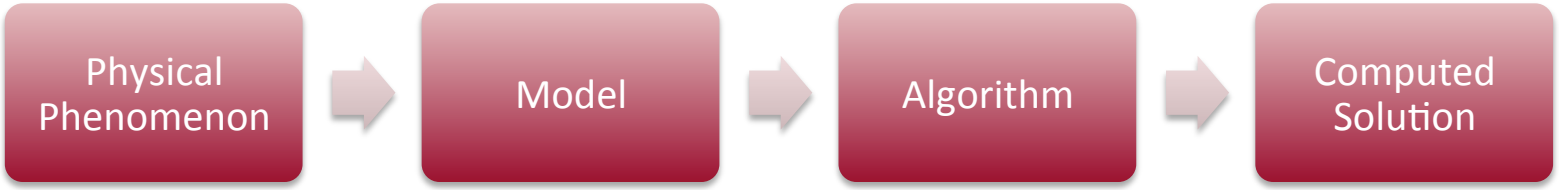
The Course Syllabus and Projects

- Recap on programming languages for physics; MATLAB and Python; summary of commands;
- Introduction to numerical computing; errors in computer calculations;
- Numerical integration methods; Euler's method; higher-order methods;
- Precision vs. accuracy; validation;
- Phase space; conserved quantities;
- Introduction to mappings and nonlinear systems;
- *Example: Methods for solving the linear and non-linear simple harmonic oscillator.*
- Introduction to Monte Carlo methods; Monte Carlo integration; classical problems;
- Pseudorandom and quasirandom sampling; methods of sampling; generation of distributions;
- Particle transport simulation; nuclear cross sections; particle histories; applications of Monte Carlo transport;
- *Example: Simulation of penetration of neutrons through shielding.*
- From mappings to linear optics; the concept of lattices;
- Transfer matrices and periodic solutions; propagation of linear optics parameters;
- Classic optical systems: the FODO, the double-bend achromat;
- Matching and optimisation; penalty/objective functions;
- Hill-climbing methods: Cauchy's method, Nelder-Mead, simulated annealing;
- Variables and constraints; under- and over-constrained problems;
- *Example: MAD8 matching of FODO Twiss values;*
- Multiple-configuration methods; genetic algorithms and evolutionary algorithms;
- A bestiary of codes; choosing the right code;
- Common pitfalls;
- *Example: Particle tracking in MAD8;*

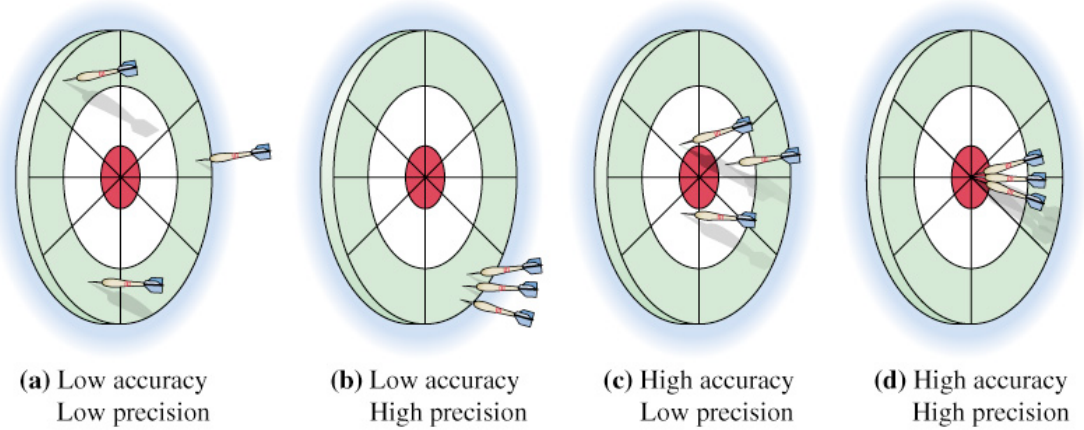
MANCHESTER 1824

The University of Manchester

Steps in a Model Simulation



Accuracy versus Precision



*The **accuracy** of a simulation is the degree of closeness of estimates of a quantity to their actual (true) value.*

*The **precision** of a simulation, also called reproducibility or repeatability, is the degree to which repeated simulations under unchanged conditions show the same results.*

Determinants of Accuracy and Precision

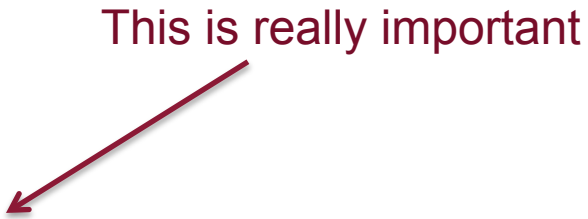
- **Modelisation** Error

- **Discretisation** Error
- Step Sizes
- Quantization

- **Computing** Error
- (numerical precision)

Errors in the Model of the Simple Harmonic Oscillator

- **Modelisation** Error
  - Restoring force truly linear?
  - Damping force truly linear?
  - Knowledge of the *actual* k, m and b in the model



- **Discretisation** Error
  - Evaluation is only carried out at specific times
  - The integration method uses those times to determine the *step size*

- **Computing** Error
  - Numerical precision
  - Truncation when using approximate formulae, e.g. series expansions
  - Truncation of constants

Monte Carlo Integration

- Integration over an interval can be written as an average:

$$S = \int_{x_1}^{x_2} f(x)dx = (x_2 - x_1) < f(x) >$$

- which can also be written as:

$$S \simeq (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

- Writing this as an algorithm, we may say:

$$x_i = (x_2 - x_1)u_i + x_1$$

$$f_i = f(x_i)$$

$$S \simeq S_N = (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{N} f_i$$

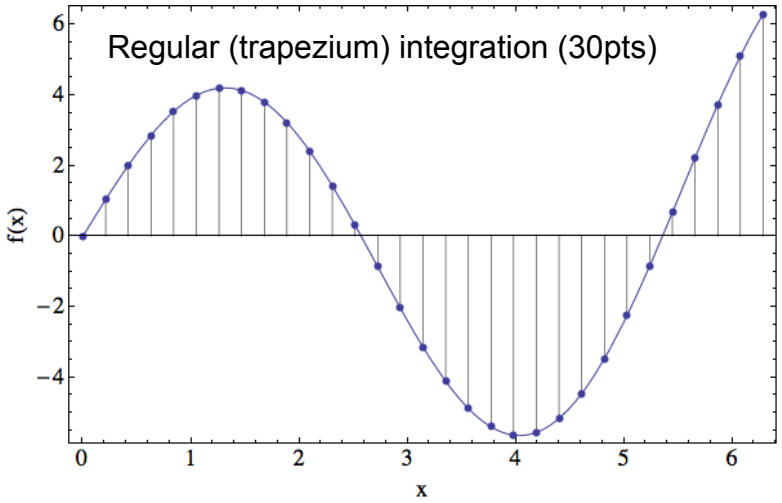
- or, progressively for each sample j:

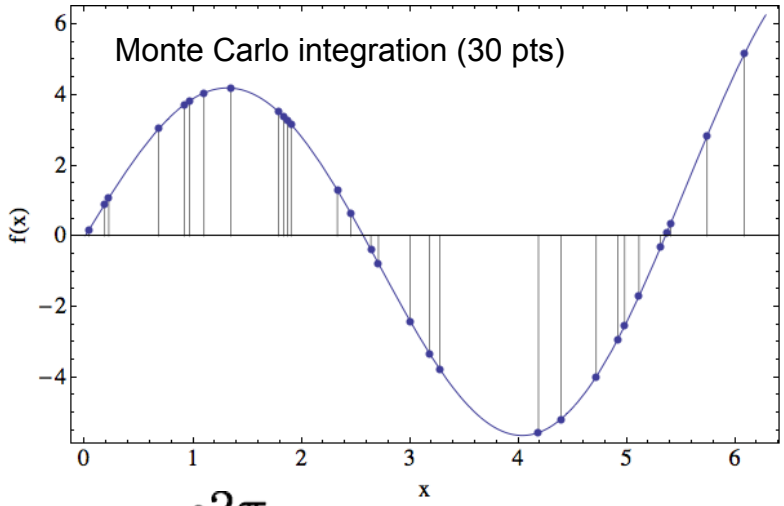
$$S \simeq S_j = (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{j} f_i$$

$$S \simeq (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

(Crude) Monte Carlo Integration

Consider $f(x) = x\cos x + 4\sin x$

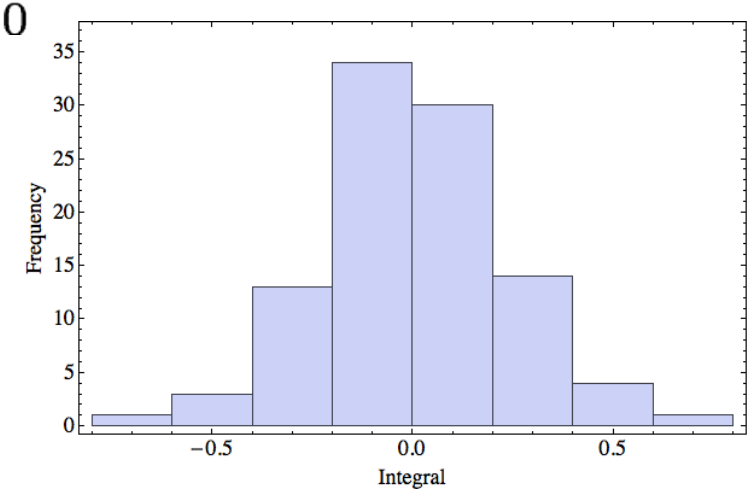




In this particular case, we know analytically

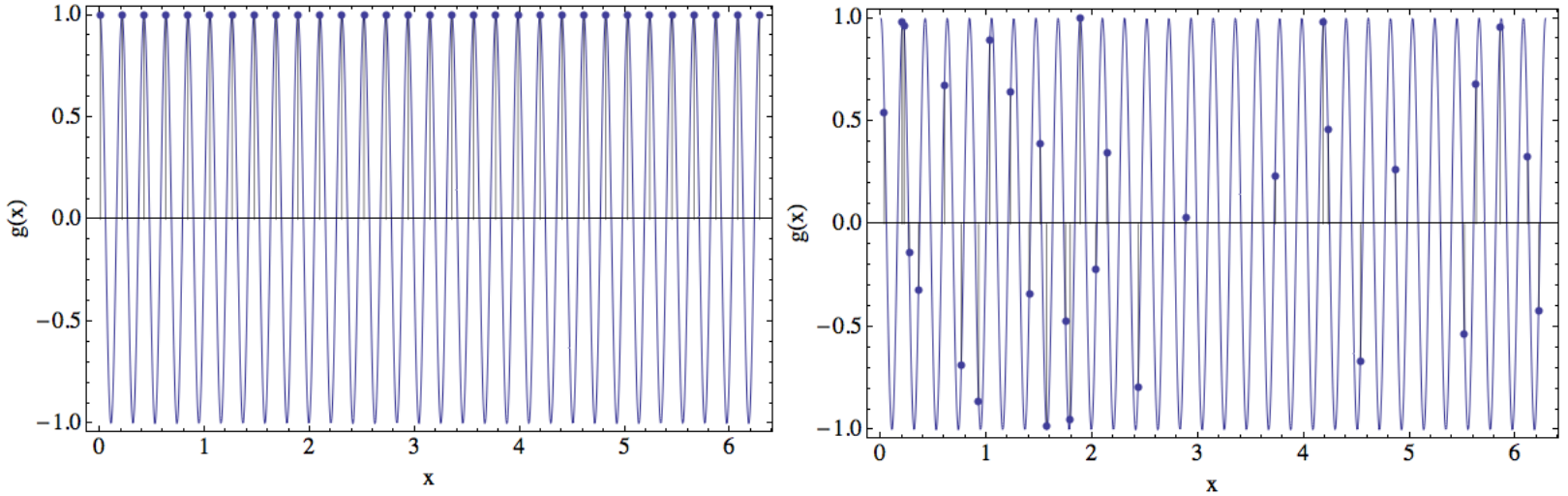
$$\int_0^{2\pi} x\cos x + 4\sin x dx = 0$$

For n=10^4 random points, we converge on the correct integral, as we should. (Integral performed 100 times)



Pathological Cases

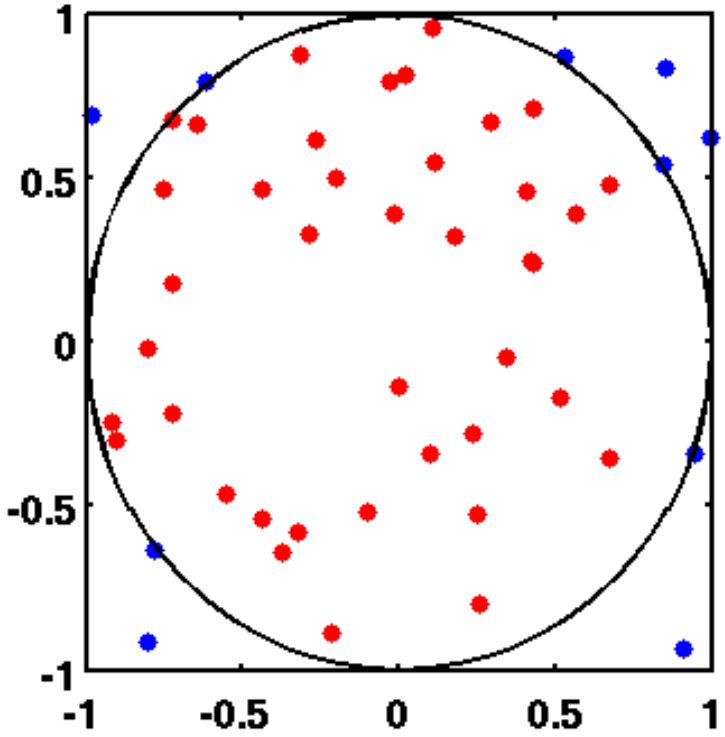
$$g(x) = \cos 30x$$ (30 pts)



Pathological behaviour can also occur with more complicated functions, which you may encounter without realising, e.g. in particular things like following particle interactions in matter.

Generalisation is Monte Carlo method, where we evaluate some complex function f(x) by repeated evaluation.

Finding the Area of a Circle: Hit-or-Miss Monte Carlo



- Pick two random numbers between -1 and +1

$$r^2 = x^2 + y^2 < R^2$$

- Determine for each pair whether

$$[P(r) < R] = \frac{\pi}{4}$$

So we need some way of generating the random numbers….

Random Numbers

1. There is no such thing as a ‘random’ number generator.

2. It is generally a **bad** idea to use the random number generator that comes with your favourite compiler. Using the code in **Numerical Recipes** is particularly bad.

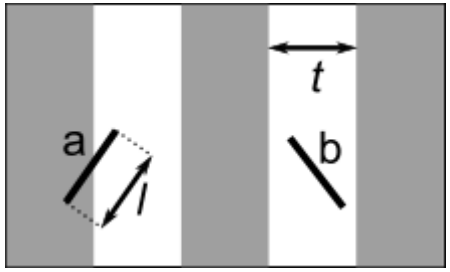
Buffon's Needle

Georges-Louis Leclerc, Comte de Buffon (1707-1788)

- Buffon was a noted naturalist who wrote a 36-volume Natural History, and from his studies observed that similar environments have distinct species (Buffon's Law) – posited that there therefore must have been improvement or degeneration of pre-existing species.
- A precursor to Darwin, who credited Buffon in the foreword to the *Origin of Species.*
- Also a contributor to early probability theory.
- The Buffon's Needle method is named after him.

Buffon's Needle: Statement of the Problem

- Buffon posed the following question:
- *Suppose we have a floor made of parallel strips of wood, each the same width t, and we drop a needle of length l onto the floor. What is the probability that the needle will lie across a line between two strips?*



- We generate (physically drop) many random samples, and measure the answer.

- Analytically, we can show that if *t*>*l*, then for *n* needles dropped with *h* of the needles crossing lines, the probability is:

$$\frac{h}{n} = \frac{2l}{t\pi},$$

- We can invert this equation to give:

$$\pi = \frac{2ln}{th}.$$

- In other words, we can drop some needles onto strip flooring, and from it get an estimate for π.

Trying out Buffon's Needle

- Mario Lazzirini did the experiment manually in 1901, manually throwing the needle 3408 times. In his case he used

$$l = \frac{5}{6}t$$

- for which the probability the needle will cross a line is

$$P = \frac{5}{3\pi}$$

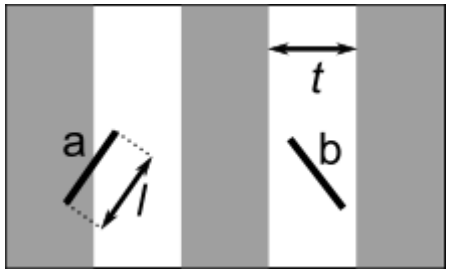
- For *n* throws giving *h* crossings, we may estimate π as

$$\pi \simeq \frac{5}{3}\frac{n}{h}$$

- Lazzirini found *h* = 1808, giving

$$\pi \simeq \frac{355}{113}$$

- (this is actually a slightly suspicious result: try to work out why)



Monte Carlo simulation

Buffon's Needle is an example of a Monte Carlo approach:

To find an approximate solution to a problem, we throw random samples at it and interpret the result we obtain.

The name Monte Carlo was coined by Stanislaw Ulam, who also incidentally (with Edward Teller) tried to patent the design of the hydrogen bomb.

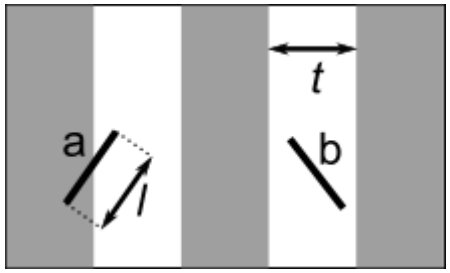
Ulam used this approach (along with Enrico Fermi) during the Manhattan Project, and chose the name as his uncle used to borrow money to go gambling at the Monte Carlo casino.

Stanislaw Ulam (1909-1984)

The casino at Monte Carlo

Problems with the Monte Carlo Method

- Of course, in order to perform a Monte Carlo simulation, we need a reliable source of random numbers.
- If not, then our answer could be wrong.



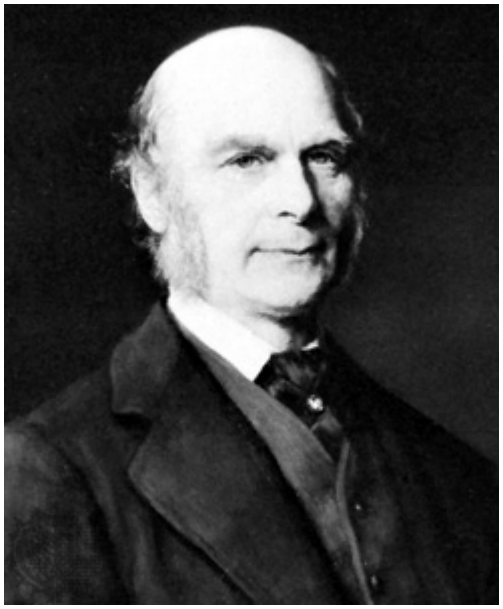
- For example, what if – when we throw our needles – we unwittingly put in a bias such that they more predominantly in line with our strips?
- If so, then we will systematically over-estimate π.

$$\pi \simeq \frac{5}{3}\frac{n}{h}$$

- In computer simulation, as well as physical simulation, we need a reliable way of generating random numbers.

Some methods of generating random numbers

- One way of generating random numbers is by using some physical process that appears to be truly random.
- Francis Galton proposed using dice.
- (By the way, Galton also drew the first weather map, did the first scientific studies of fingerprinting, and coined the terms *eugenics*, and *nature versus nurture*.)
- One simply throws a die many times, and gets a **uniformly-distributed random integer** between 1 and 6. **Uniform** means that each number (1,2,3,4,5,6) has an equal probability of being generated as a result.



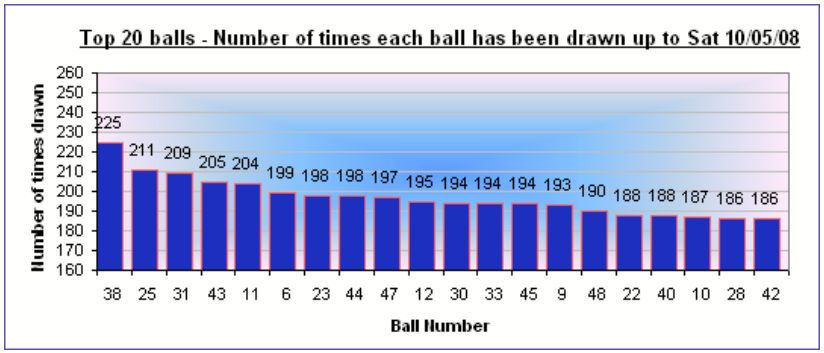
Francis Galton (1822-1911)

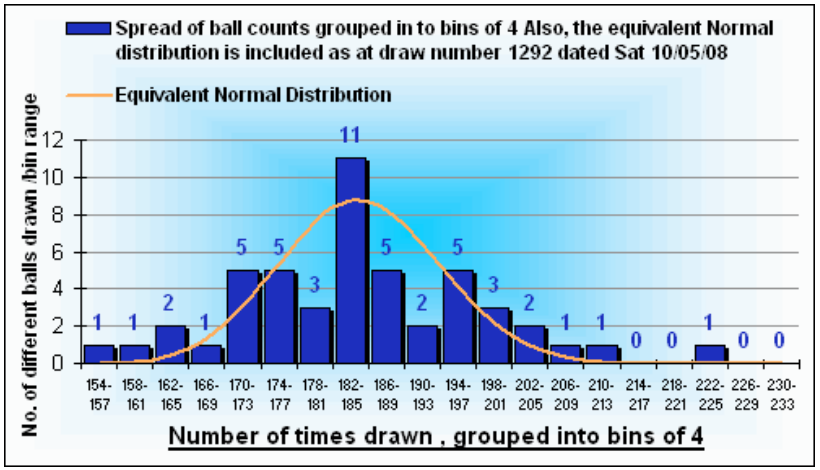
MANCHESTER 1824

The University of Manchester

Handling Little Balls

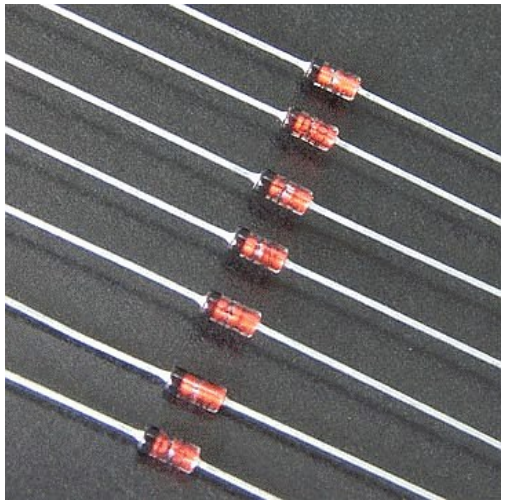
- In the national lottery, they use 49 little balls (80g weight each).
- Over all machines it is thought to be random, but ball number 38 does come up most often.
- The Royal Statistical Society said ‘Ball number 38 should be physically examined’, after an analysis by the University of Salford Centre for the Study of Gambling.
- Apparently, the lottery people examine their balls each draw anyway….





Other physical methods for generating random numbers

- Other physical methods are possible, which are more suited to use on computers – i.e. they generate random numbers faster.
- Electrical shot noise
- Thermal noise
- Avalanche noise (e.g. from a Zener diode)
- Nuclear decay/Geiger counter
- Atmospheric noise
- Drift between two clocks on a microprocessor (most hardware RNGs use this approach



- However, they all suffer from the same two problems:

1. They suffer from systematic bias (often hidden or progressive)
2. They are non-repeatable, precisely because they are random.

- Example of systematic bias:
- The dated bits of paper thrown into a bin used for the Vietnam draft were biased towards dates at the end of the year.

Why do you want true random numbers anyway?

- Actually, for most simulation purposes you **don't** want **true random numbers**. What you want are sequences which have no correlation with the process you are simulating.
- This is a subtle concept, which we will come back to later.

- To allow repeatability, which allows you to debug your simulation, and to repeat your great result so you can show someone later, people usually use **pseudo-random numbers**.
- 'The generation of random numbers is too important to be left to chance' – Robert Coveyou, Manhattan Project, and one of the pioneers of pseudo-random number generation.

Pseudo-Random Numbers

- A **pseudo-random number generator** (PRNG) produces a sequence of numbers that exhibits **statistical randomness**. The output sequence is unbiased, i.e. the statistical measures are what you would expect from a random variable.
- However, the sequence is entirely deterministic. Given any number in the sequence, you can generate the next number. Therefore, you can repeat the entire sequence given a starting number. This number is known as the **seed**.

- The quintessential example of the PRNG is the Linear Congruential Generator (LCG), invented by Lehmer:

$$X_{n+1} = (aX_n + c) \mod m$$

- which has the following parameters:

- The modulus: $0 < m$
- The multiplier: $0 \leq a < m$
- The increment: $0 < c < m$
- The seed: $0 \leq X_0 < m$

Derrick H. Lehmer (1904-1991)

Features of the LCG

- An LCG sequence repeats over a full period *m* as long as:
  - *c* and *m* are relatively prime
  - *a*-1 is divisible by all prime factors of *m*
  - *a*-1 is a multiple of 4 if *m* is a multiple of 4 (this is the Hull-Dobell theorem)

- With these parameters, we can draw a random number from 0 to 1 of precision *m* using the formula:

$$s_n = \frac{X_n}{m}$$

The LCG formula

$$X_{n+1} = (aX_n + c) \mod m$$

$$0 < m$$

$$0 \leq a < m$$

$$0 < c < m$$

$$0 \leq X_0 < m$$

The LCG as the positions of the hour hand on a clock

$$X_{n+1} = (aX_n + c) \mod m$$

$0 < m$

$0 \leq a < m$

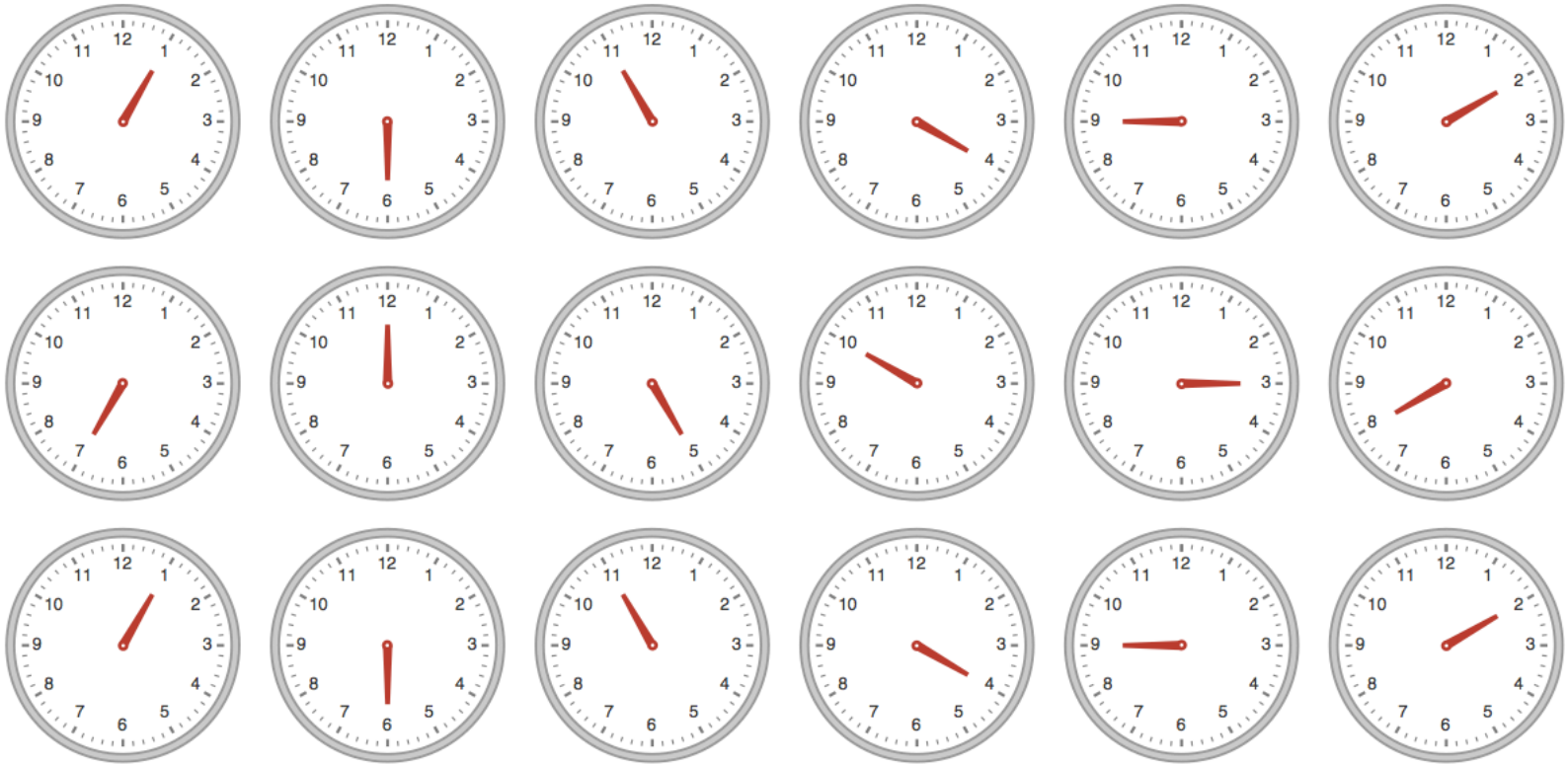
$0 < c < m$

$0 \leq X_0 < m$

$$m = 12$$

$$a = 1$$

$$c = 5$$

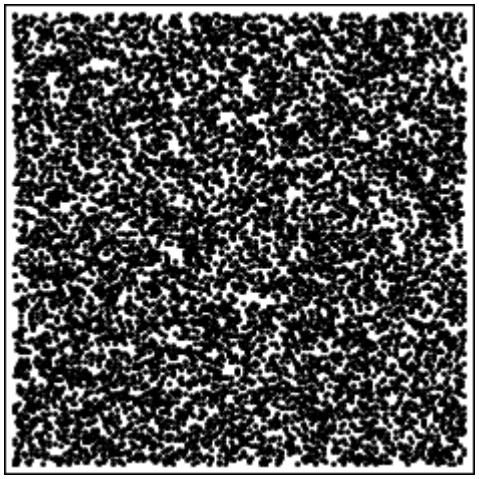


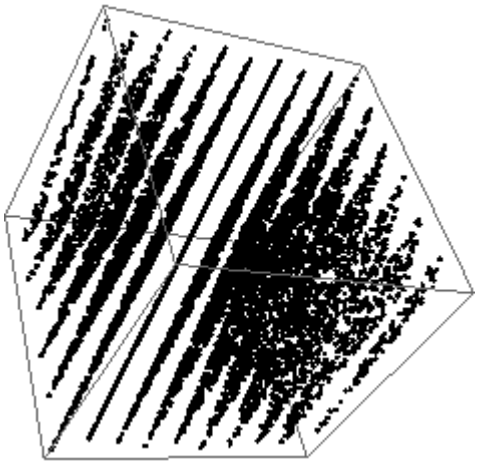
It seemed like a good idea at the time

- The RANDU algorithm is a particularly bad implementation:

$$V_{j+1} \equiv (65539 V_j) \mod 2^{31}$$

- (it was chosen because it runs fast)
- But actually, all LCGs produce points that lie on hyperplanes (Marsaglias' theorem) – there are correlations between the points. Looking at these planes is known as the **spectral test**.
- But LCGs are still widely used…





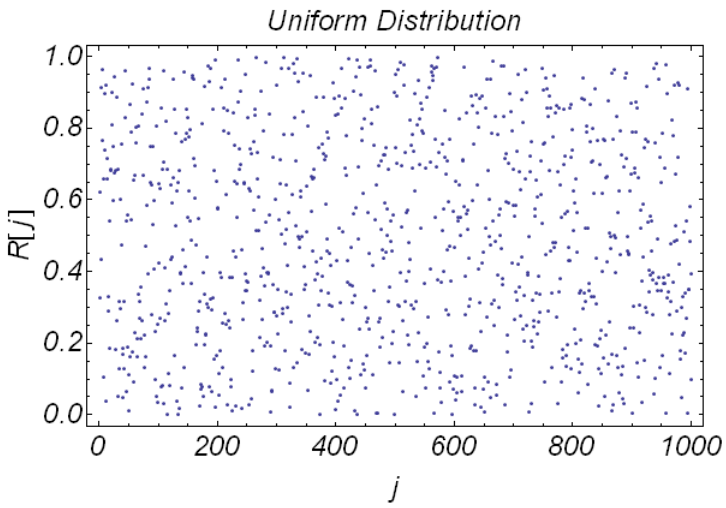
Compiler	m	a	c
Numerical Recipes	2^{32}	1664525	1013904223
GNU Compiler	2^{32}	69069	5
ANSI/IBM C	2^{32}	1103515245	12345
Borland	2^{32}	134775813	1
Microsoft VC++	2^{32}	214013	2531011

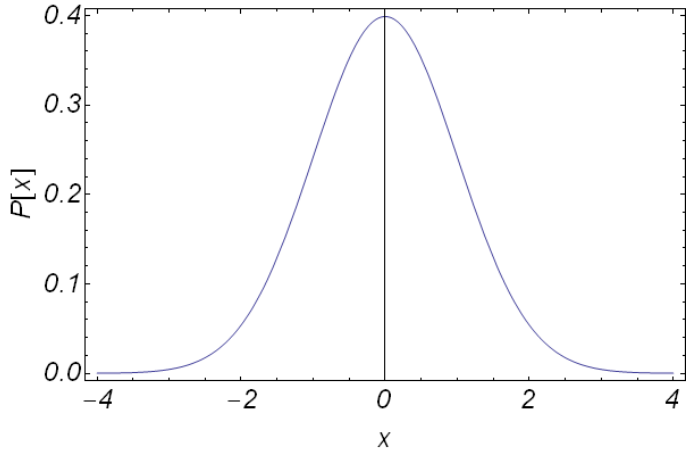
$$X_{n+1} = (aX_n + c) \mod m$$

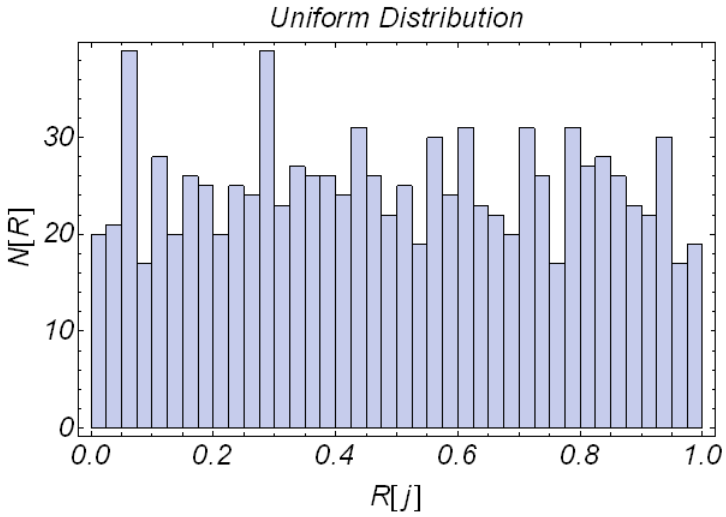
Creating Non-Uniform Distributions

- LCGs produce a sequence of numbers which is **uniformly distributed**.
- But how do we produce numbers which don't have a uniform distribution?
- For example, a Gaussian distribution?

$$X_{n+1} = (aX_n + c) \mod m$$



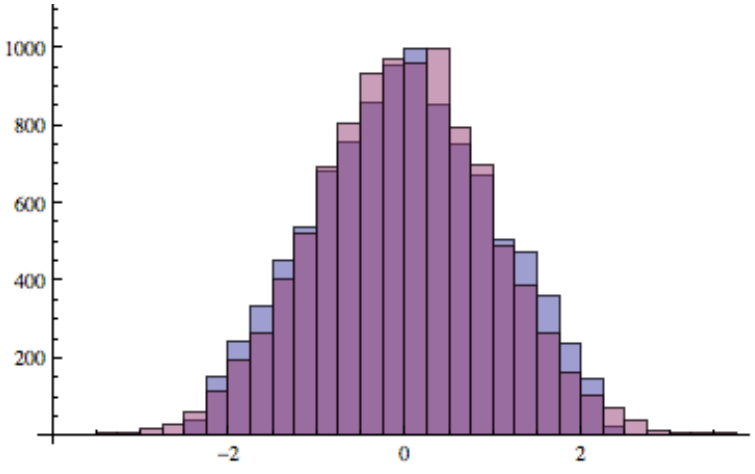




Using the Central Limit Theorem

- To create a distribution whose PDF is that of a normal distribution, we recall that independent, uniform random variates – when combined – tend to a normally-distributed variate.
- i.e. we can pick some uniformly-distributed numbers and sum them to get a Gaussian-distributed one.
- The important question is: how many numbers do I need to combine to get a decent Gaussian variate? The answer is 12
- But even 2 works pretty well near the mean

$$g = \frac{\sum_{i=1}^{n} u_i - n/2}{\sqrt{n/12}}$$



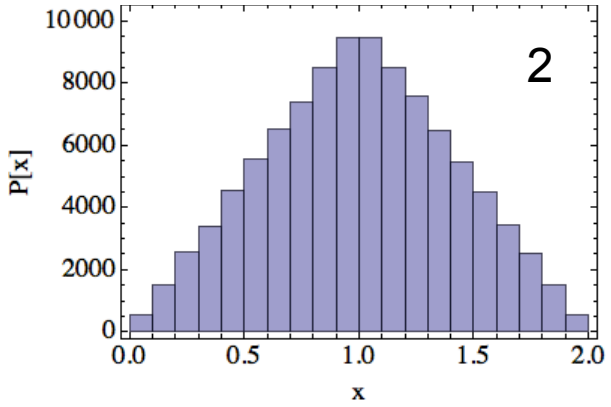
MANCHESTER 1824

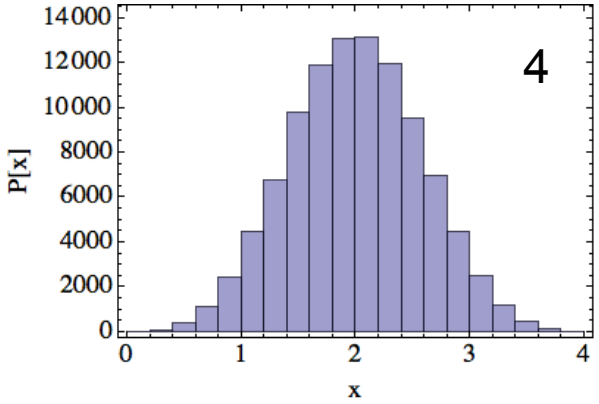
The University of Manchester

Approaching the Central Limit

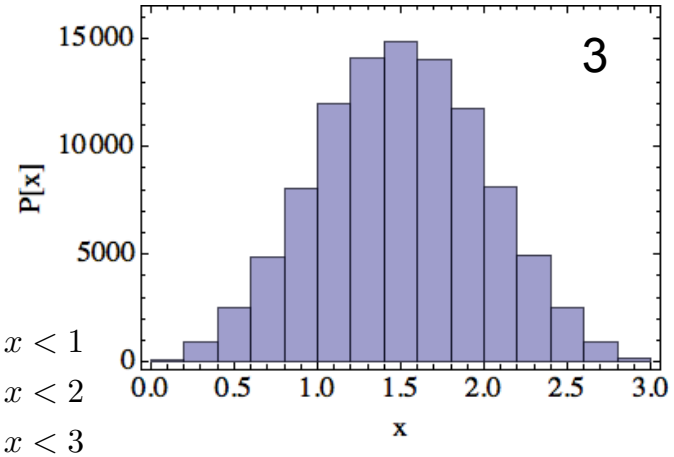
$$g = \frac{\sum_{i=1}^{n} u_i - n/2}{\sqrt{n/12}}$$

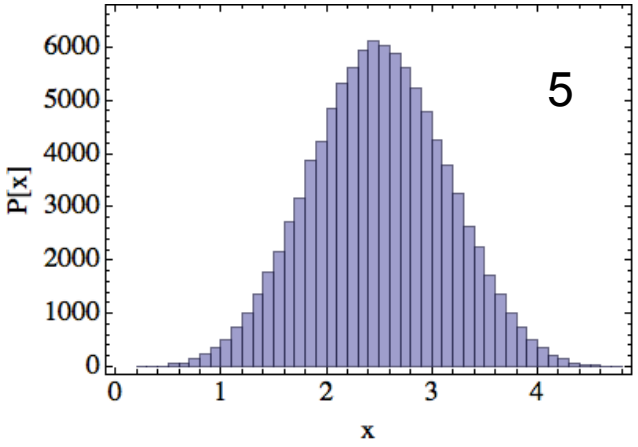
$$\frac{dP[x]}{dx} = \begin{cases} x & 0 < x < 1 \\ 2 - x, & 1 < x < 2 \end{cases}$$





$$\frac{dP[x]}{dx} = \begin{cases} \frac{1}{2}x^2 & 0 < x < 1 \\ \frac{3}{4} - (x - \frac{3}{2})^2, & 1 < x < 2 \\ \frac{1}{2}(3 - x)^2, & 2 < x < 3 \end{cases}$$





If we use n=12 the algorithm is very simple!
Any random variate can be the input – a uniform one is simply convenient

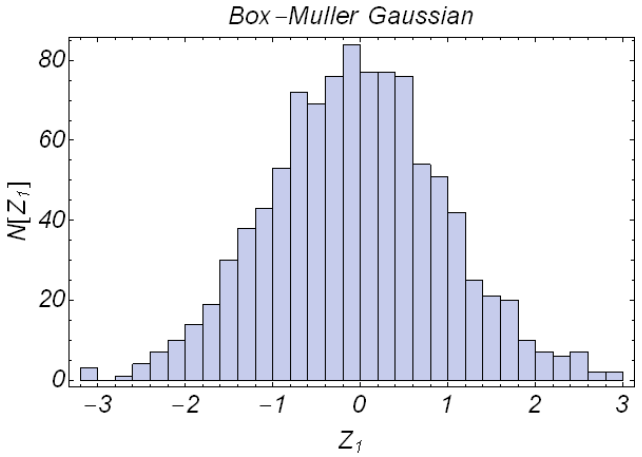
The Box-Muller Transformation

- The Box-Muller method is a handy way of generating Gaussians using two uniformly-distributed numbers.
- The result is a normal distribution with unit mean and standard deviation.
- Other means and s.d.'s can be generated simply: $X_j = \sigma Z_j + \mu$

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

$$Z_1 = R\sin(\Theta) = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$

Pair U_1, U_2 → Pair Z_1, Z_2



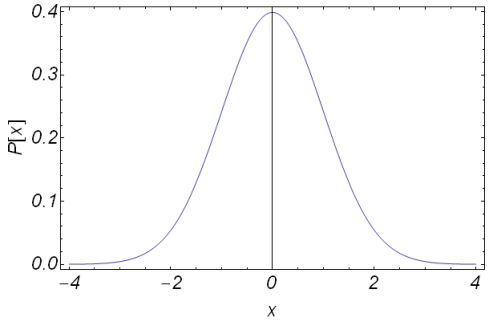
Box-Muller only works for Gaussians, but is handy because the formulae can be re-cast for very fast computation.

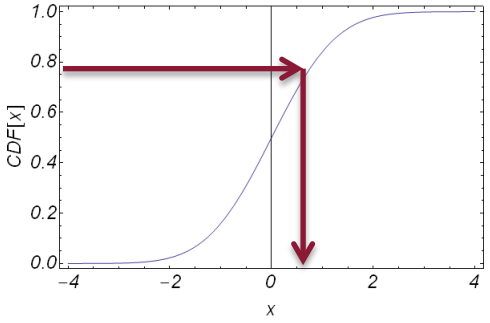
(Polar method and Marsaglia's Polar Method)

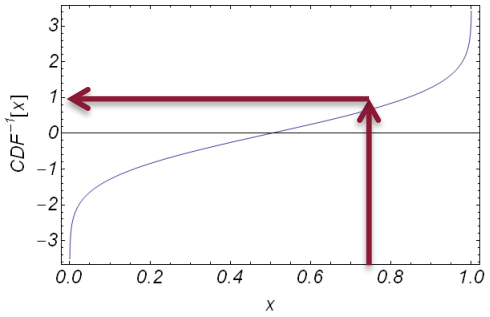
Box, G. E. P. and Muller, M. E. "A Note on the Generation of Random Normal Deviates." *Ann. Math. Stat.* **29**, 610-611 (1958)

Inverse distribution sampling/cumulative distribution sampling

The inverse distribution method is quite simple – integrate the **probability density function** of the distribution you want to sample from, and then invert it to form the **inverse cumulative distribution** function. Multiply a uniformly-distributed variable by the inverse CDF to get a distribution distributed according to your original PDF.

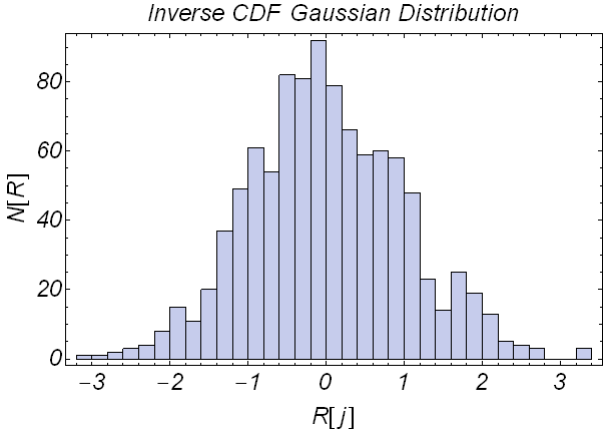






$$\mathrm{cdf}^{-1}(x) = \sqrt{2}\mathrm{erf}^{-1}(2x-1)$$

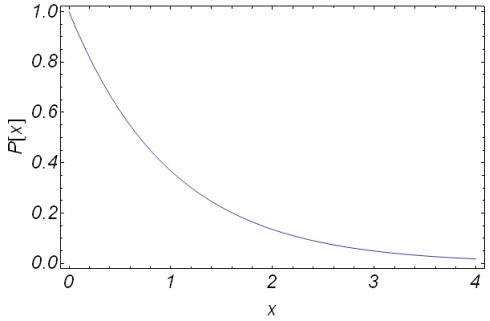
$$\mathrm{erf}^{-1}(x) = \frac{1}{2}\sqrt{\pi}\left(x + \frac{\pi}{12}x^3 + \frac{7\pi^2}{480}x^5 + \frac{127\pi^3}{40320}x^7 + \frac{4369\pi^4}{5806080}x^9 + \frac{34807\pi^5}{182476800}x^{11} + \cdots\right).$$



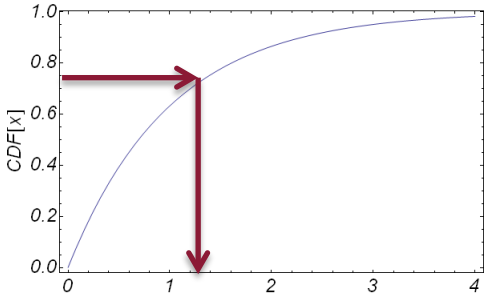
Works just great for a Gaussian (again….)

Sampling the Exponential Distribution

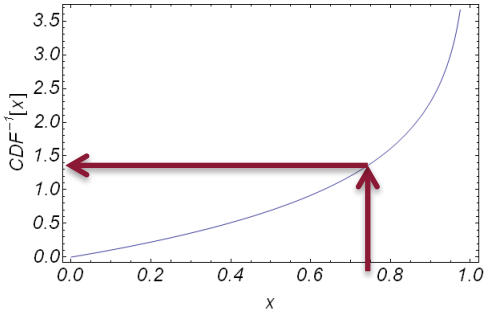
- Sampling functions is straightforward, provided the inverse CDF can be found.



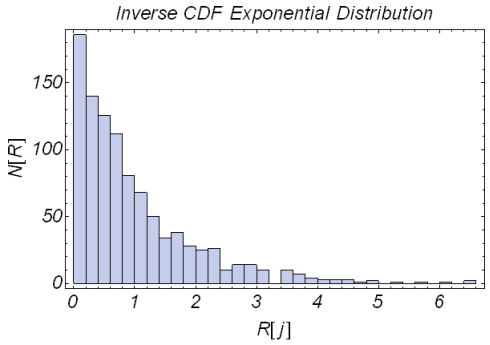
$$\mathrm{pdf}(x) = e^{-x}$$



$$\mathrm{cdf}(x) = \int_0^x \mathrm{pdf}(x) = 1 - e^{-x}$$

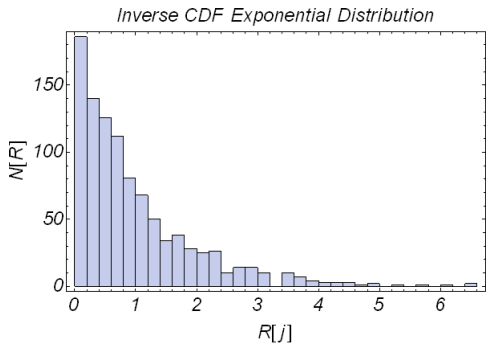


$$\mathrm{cdf}^{-1}(z) = -\ln(1-z)$$



This is really important! Pay attention here!

Exponential Distribution via Normal and Antithetic Variates



$$\mathrm{cdf}^{-1}(x) = -\log(1-x)$$

$$x_i = -\log(1-u_i)$$

$$x_i = -\log(u_i)$$

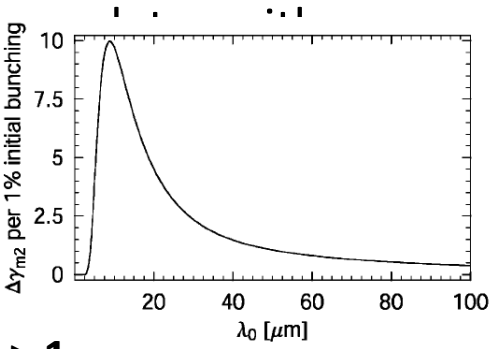
For mean free path λ

$$P[x] = e^{-x/\lambda}$$

$$s_i = -\lambda \log(u_i)$$

Do the correlations matter?

- The answer is yes…. and no. It all comes down to the **spatial periodicity** of the random number sequence.
- The answer is definitely yes if the spatial periodicity physical process of interest.
- **Example**: Microbunching gain
- PRSTAB **7**, 074401 (2004)

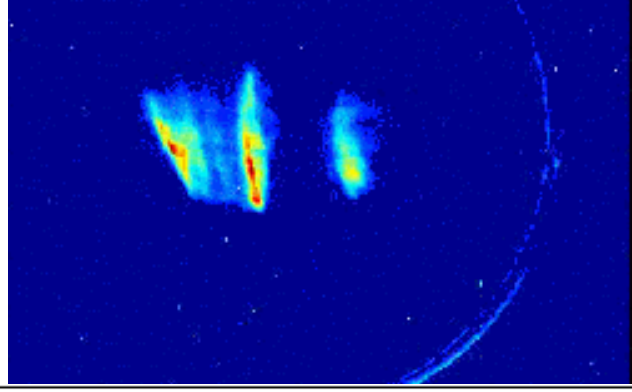


MANCHESTER 1824

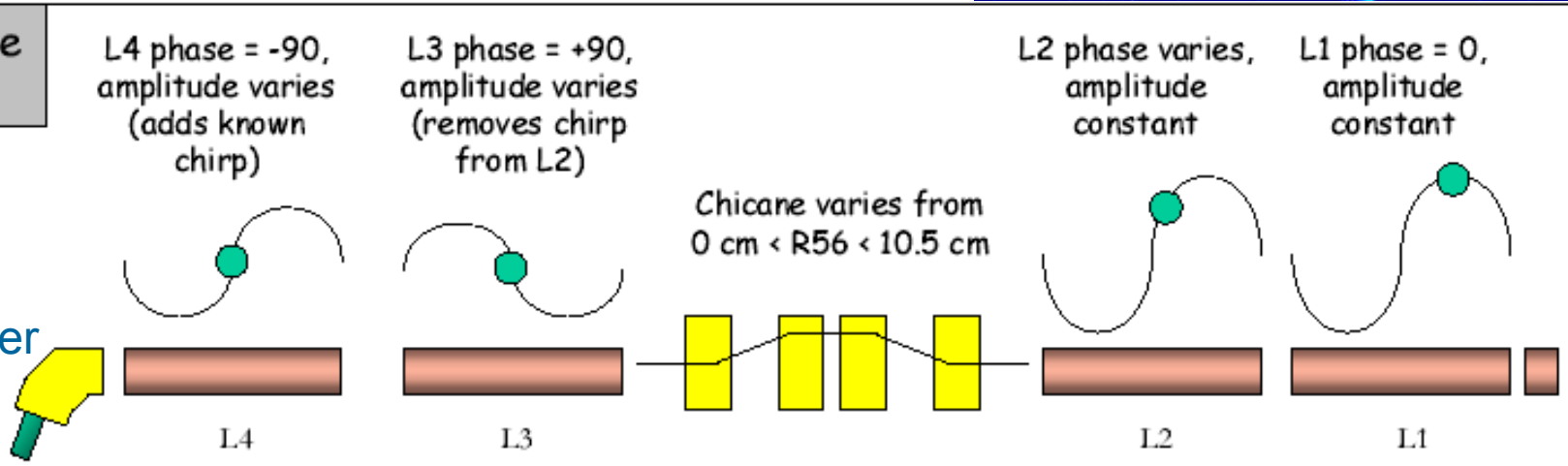
The University of Manchester

SDL Zero-Phasing Experiment

(W. Graves et al.)

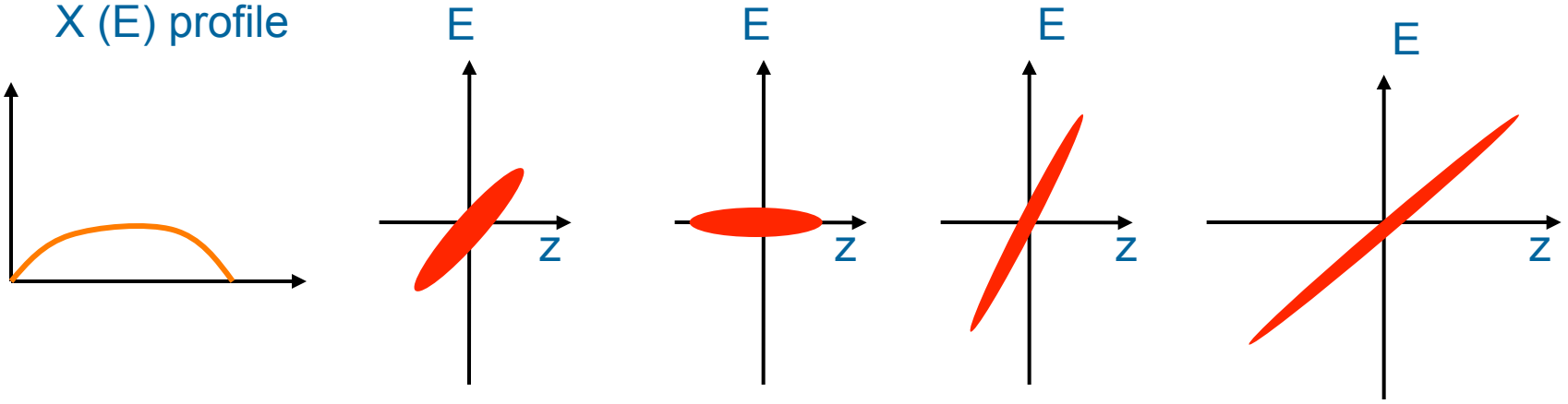


RF zero-phase time profile



65 MeV Energy spectrometer

X (E) profile

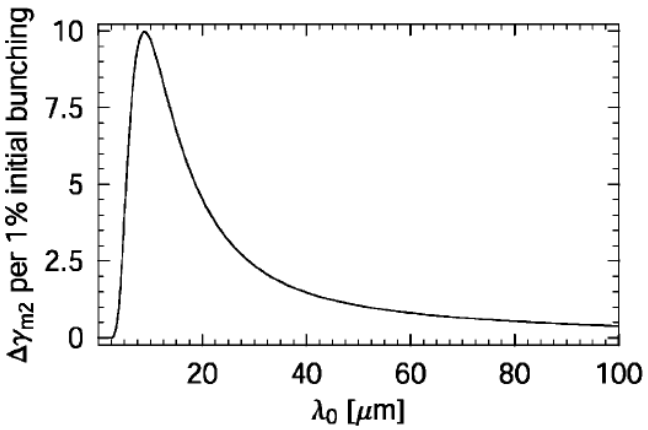


Real Particles vs. Macroparticles

- Simulations of microbunching rely on using macroparticles rather than actual particles.
  - The reason is typical bunches contain
  - $N \sim 10^9$ particles, too many for a code.

- If the particles are non-interacting then LCGs are usually ok – only the statistical measures are important.

- However, if the physical process of interest cares about spatial periodicities, then we have to be careful….



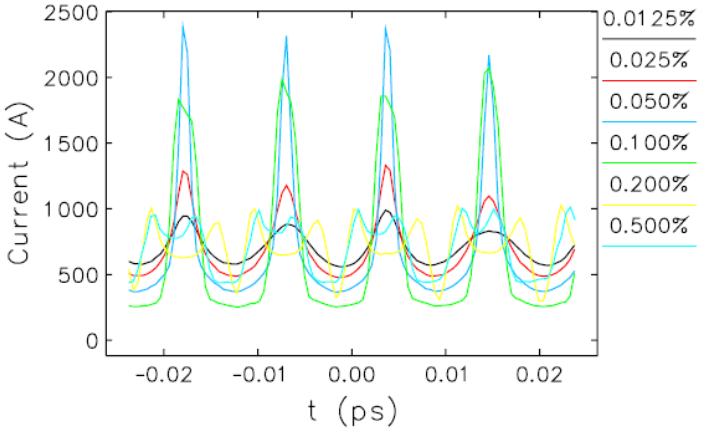


FIG. 8. (Color) Final longitudinal density for FERMI for 30 μm initial modulations of various amplitudes, showing only the central 100 bins.

M. Borland, "Modeling of the microbunching instability" *Phys. Rev. S.T.A.B.* **11**, 030701 (2008)

The von Neumann Method

- John von Neumann was a pioneering mathematician. He also patented a design for an atomic bomb (with Klaus Fuchs), and proposed Kyoto as the target for 'Fat Man'. Ironically, he died of cancer probably caused by watching the atomic tests at Bikini Atoll.

- He also invented the field of **cellular automata**, which are used for cryptography and pseudo-random number generation.

- Here, we look at his elegant method for generating a sample drawn from an arbitrary distribution, the **rejection method**.

János Lajos Neumann/ John von Neumann (1903-1957)

THE INSTITUTE FOR ADVANCED STUDY
SCHOOL OF MATHEMATICS
PRINCETON, NEW JERSEY

May 21, 1947

Mr. Stan Ulam
Post Office Box 1663
Santa Fe
New Mexico

Dear Stan,

Thanks for your letter of the 19th. I need not tell you that Klari and I are looking forward to the trip and visit at Los Alamos this Summer. I have already received the necessary papers from Carson Mark. I filled out and returned mine yesterday; Klari's will follow today.

I am very glad that preparations for the random numbers work are to begin soon. In this connection, I would like to mention this: Assume that you have several random number distributions, each equidistributed in $0, 1$: $(x^i), (y^i), (z^i), \ldots$. Assume that you want one with the distribution function (density) $f(\xi)\,d\xi$: (ξ^i). One way to form it is to form the cumulative distribution function: $g(\xi) = \int^{\xi} f(\xi)\,d\xi$ to invert it $h(x) = \xi \rightleftarrows x = g(\xi)$, and to form $\xi^i = h(x^i)$ with this $h(x)$, or some approximant polynomial. This is, as I see, the method that you have in mind.

An alternative, which works if ξ and all values of $f(\xi)$ lie in $0, 1$, is this: Scan pairs x^i, y^i and use or reject x^i, y^i according to whether $y^i \leqq f(x^i)$ or not. In the first case, put $\xi^j = x^i$ in the second case form no ξ^j at that step.

The second method may occasionally be better than the first one. In some cases combinations of both may be best; e.g., form random pairs

$$\xi = \sin x, \quad \eta = \cos x$$

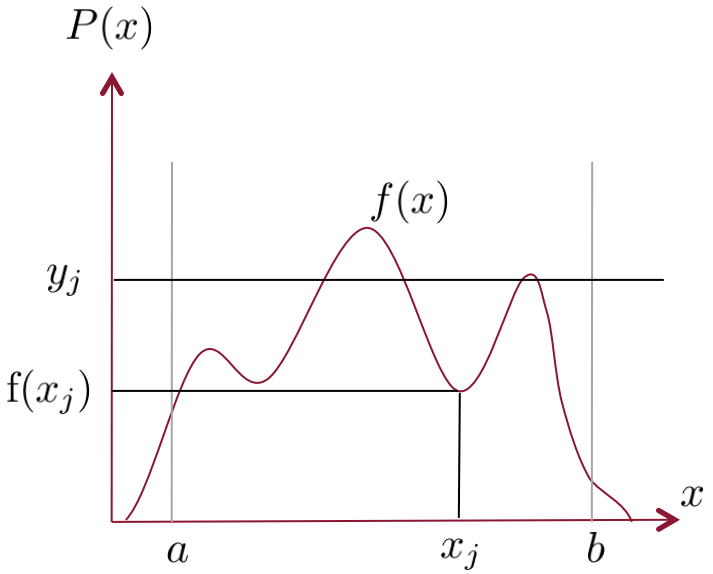
with x equidistributed between 0° and 300°. The obvious way consists of using the sin - cos - tables (with interpolation). This is clearly closely related to the first method. This is an alternative procedure:

Put $\xi = \dfrac{2t}{1+t^2}$, $\eta = \dfrac{1-t^2}{1+t^2}$, $t = \operatorname{tg} y$,

with y (which is $\frac{x}{2}$) equidistributed between 0° and 180°. Restrict y to 0° to 45°. Then the ξ, η will have to be replaced randomly by η, ξ and again by $\pm\xi, \pm\eta$. This can be done by using random digits $0, \ldots, 7$. It is also feasible with

The von Neumann Rejection Method

- The rejection method is actually very simple to understand.
- We draw a uniform variate sample x_j
- from a uniform distribution with bounds corresponding to the range of values we wish to draw samples from the PDF, $U(a,b)$



- We then calculate the PDF value for this x_j
- and compare this value with another uniform variate y_j drawn from $U(0,1)$

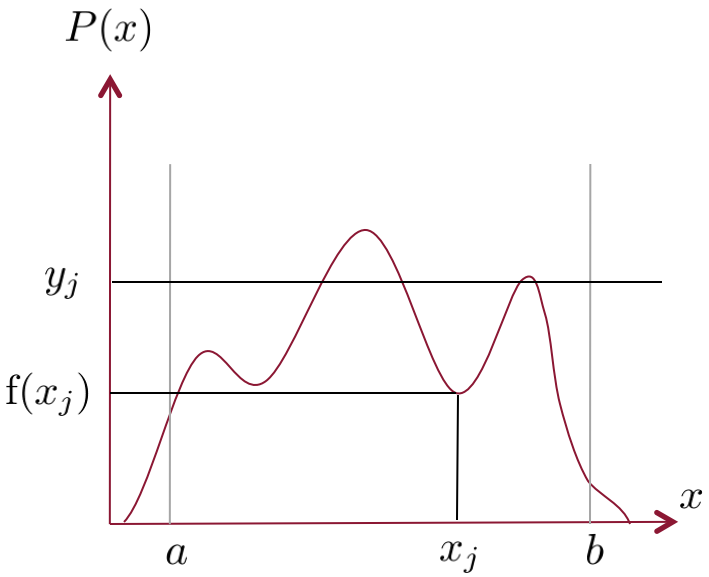
- If $f(x_j) < y_j$ then keep x_j
-

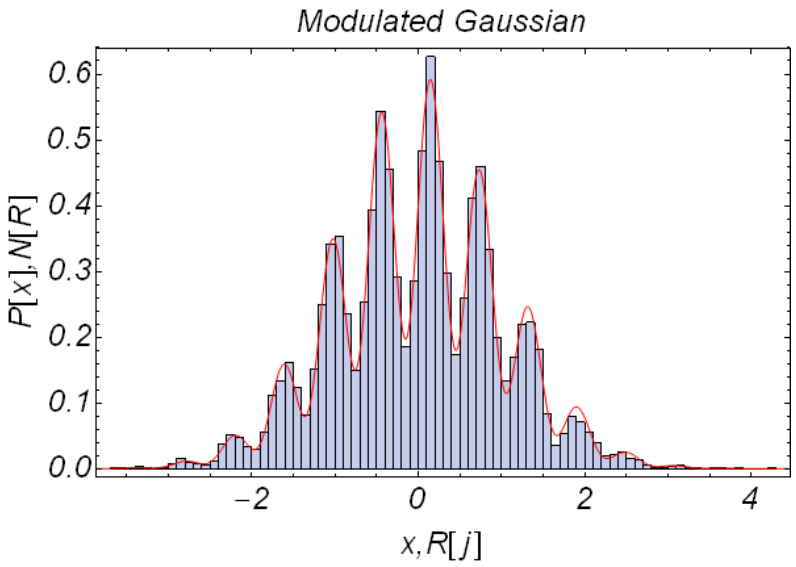
- The resulting set $\{x_j\}$ is distributed according to the PDF $f(x)$

A very elegant method, that is only limited by having to have limits a and b (for computational efficiency reasons)

In regions where there the PDF is close to zero, a large proportion of samples are rejected. This can be fixed by rescaling $f(x)$

Making a Modulated Gaussian

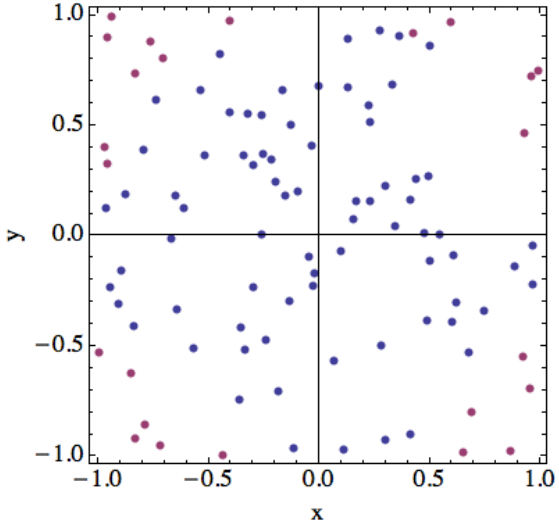




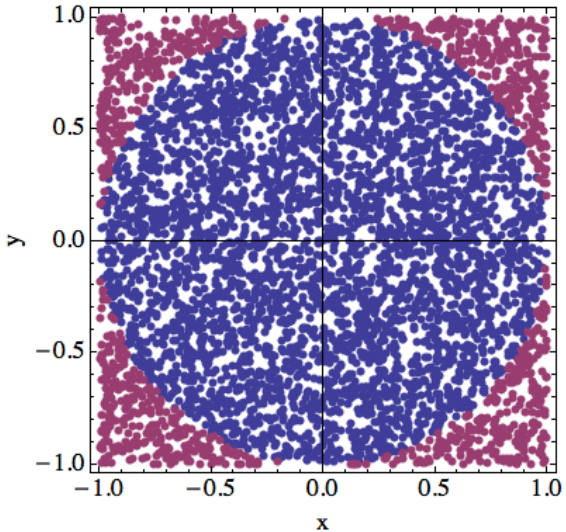
- The implementation works….
- but remember that the underlying pseudorandom samples can still be correlated if they are drawn using an LCG algorithm.

Rejection isn't that hard to bear…

- Actually, you already saw the rejection method in action in a simpler case.



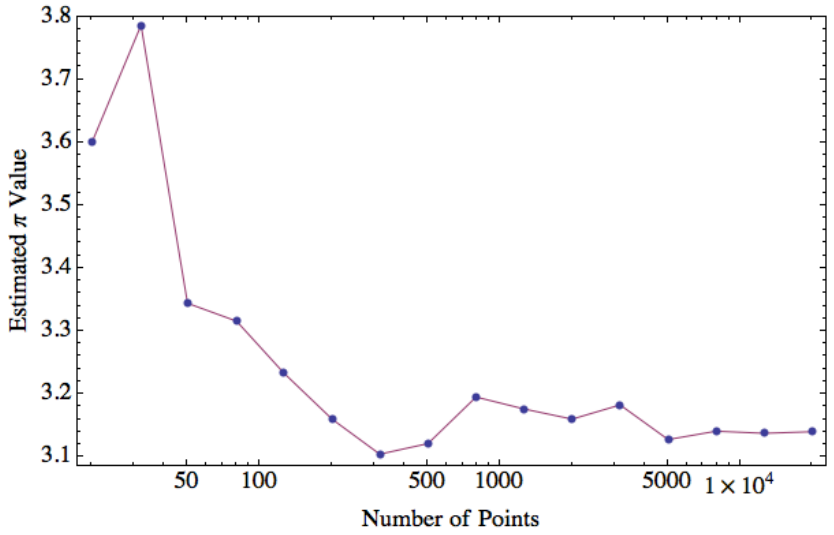
100 points



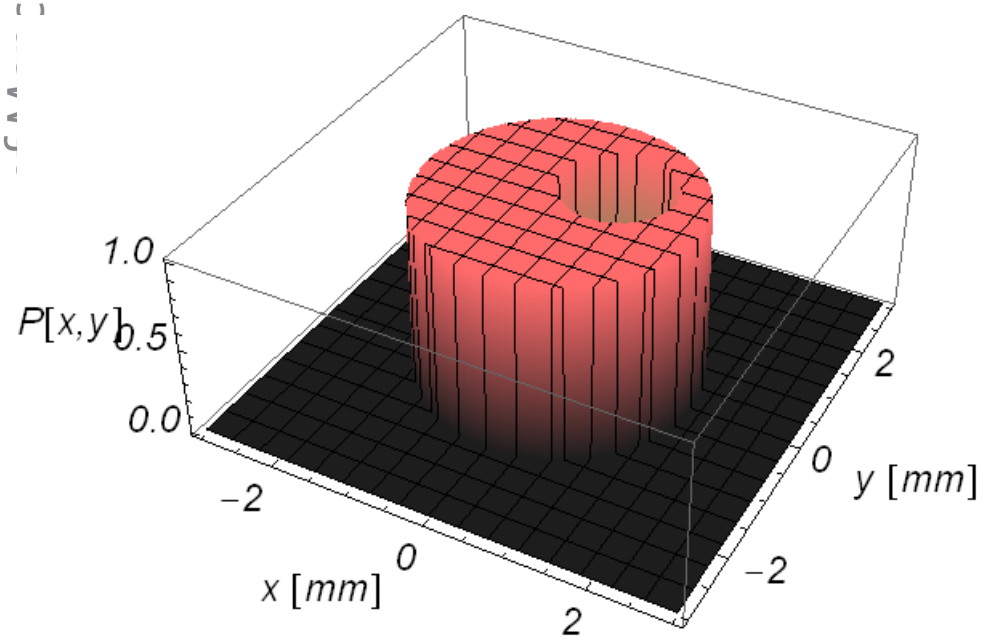
5000 points

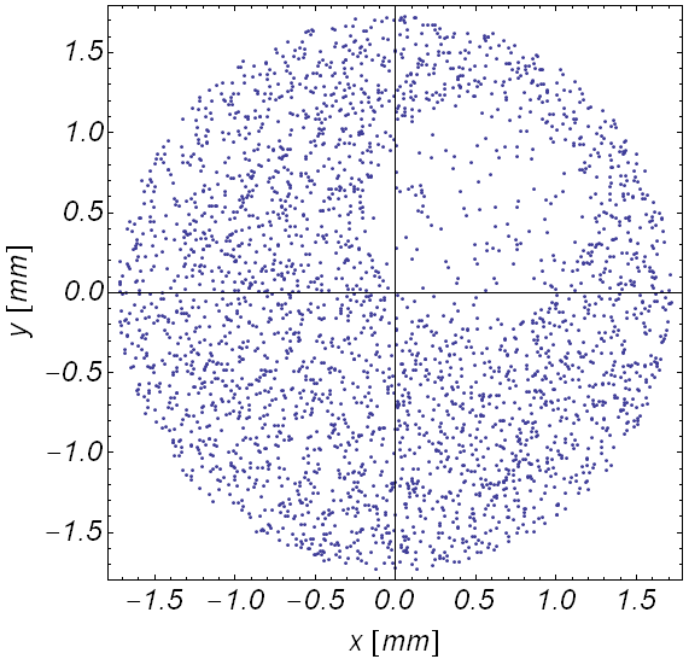
$$r^2 = x^2 + y^2 < R^2$$

$$[P(r) < R] = \frac{\pi}{4}$$



Complex distributions with rejection sampling





- A cathode emission with a hole in it. Easy with rejection sampling.

Replacements for the LCG Method

- I hope I've convinced you that using LCGs is usually flawed. There are better alternative methods, but of course you should be careful with those too.
- I'll just mention two popular ones.

- Marsaglia-Zaman (Multiply With Carry):

$$x_n = (ax_{n-r} + c_{n-1}) \bmod b,\ c_n = \left\lfloor \frac{ax_{n-r} + c_{n-1}}{b} \right\rfloor,\ n \geq r,$$

- Mersenne Twister (Matsumoto & Nishimura):
- (algorithm is hard to write down, but is an n-dimensional shift register with extra bits).

- Generally, the more complicated the random number generator, the more samples you need to pass tests of randomness.

- To get round problems with generating numbers, you can get them on a CD!

- For more information, please look at publications by Marsaglia, and at
- http://www.stat.fsu.edu/pub/diehard/

Metropolis-Hastings Markov Chain Monte Carlo

- A final method you might see.
- Based on the idea of Markov chains, in which each element in a sequence is only dependent on the previous element – a random walk.
- W. Hastings extended Nick Metropolis' method, which takes a random walk dependent on the PDF – it removes the limits on the distribution that the von Neumann method has.

Andrey Andreyevich Markov (1856-1922)

Despite the Russian Revolution, nothing particularly interesting happened to Dr. Markov

W.K. Hastings, *Biometrika* **57**, 97-109 (1970) – one of only three papers he published in his career.

One implementation of MCMC method

$$x_j$$

$$\downarrow$$

$$x' = x_j + kU(-1/2, 1/2)$$

$$\downarrow$$

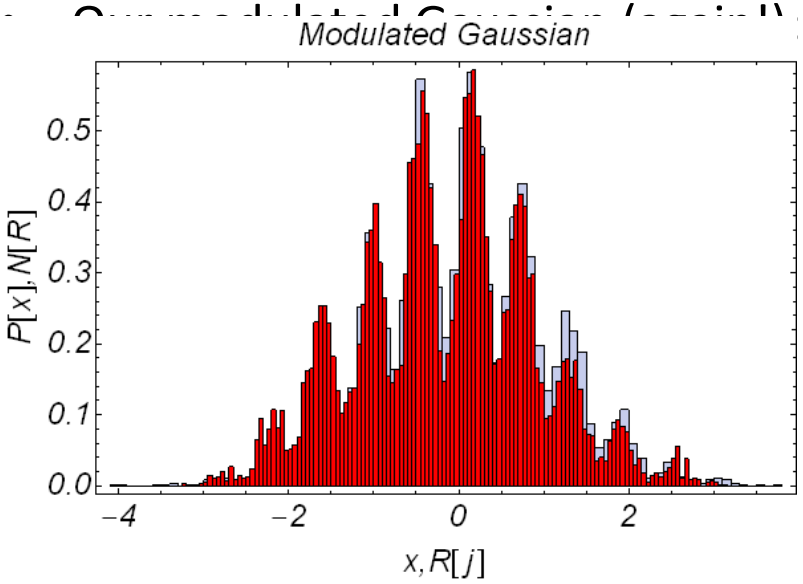
$$P[x'] > P[x_j] \text{ or } U[0,1] < \frac{P[x']}{P[x_j]} \text{ ?}$$

Y N

$$x_{j+1} = x' \qquad x_{j+1} = x_j$$

Note that we don't need to draw our step from a uniform distribution. We can draw it from any symmetric distribution.

- Very general, but quite inefficient, especially with a poor starting point; generally, the initial points are left out while the algorithm 'finds' the middle of the distribution.
- Our modulated Gaussian (again!)



Clumpiness

- Actually, pseudorandom numbers are actually not so good for some things.
- A pseudorandom macroparticle distribution will exhibit more 'clumpiness' than the real distribution.
- Apart from increasing the macroparticle number and looking for convergence in the simulation results (a brute force method), we can think of other ways of making samples than pseudorandom ones, that have:

1. The correct statistical properties (they match the real population)
2. They lack correlations that influence the simulation
3. They are more uniform – they are **low-discrepancy sequences**

- Low-discrepancy sequences are termed **quasi-random**. Simulations based on quasi-random are called **quasi-Monte Carlo** methods.

- A formula for discrepancy is complicated, but does exist.

van der Corput sequences

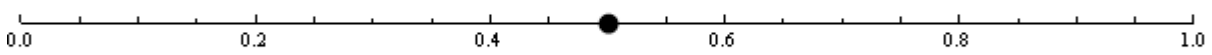
- In 1935 Johannes van der Corput asked himself how to progressively fill a unit interval with a sequence of numbers. This is the van der Corput sequence, an example of a low-discrepancy sequence.

- Easy to write down. This one is in base 2:

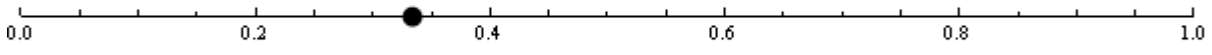
$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}, \frac{13}{16}, \frac{3}{16}, \frac{11}{16}, \frac{7}{16}, \frac{15}{16}, \cdots$$

- The formula is a bit tricky, but is based on using binary digits. Base 3,4,5…. are also possible.

Base 2



Base 3

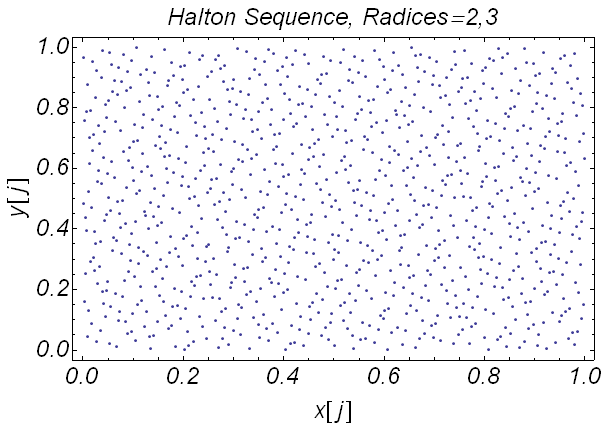


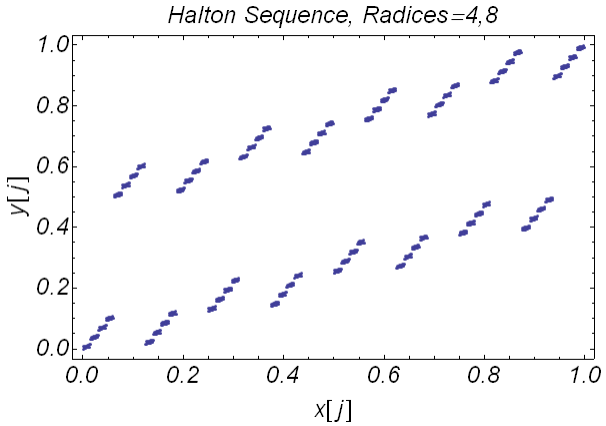
MANCHESTER 1824

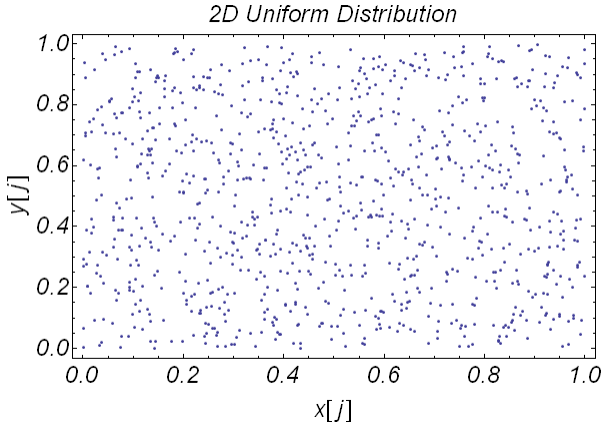
The University of Manchester

Halton Sequences

- A Halton sequence is just a series of points in n dimensions, where each coordinate is a van der Corput sequence.
- As you can see, the radices for each axis must be **relatively prime**, otherwise there will be correlations.
- Once you have that, you have a nice, uniform distribution, although of course it still has a strong spatial frequency determined by the radices.
- Again, you need to be careful, depending on the application. Halton is good for space charge, but can be bad for microbunching.
- Why should you care about the Halton sequence?
- Because it is used in pretty much every **quiet start** routine (e.g. Elegant), used for space charge, FEL modelling etc.







Microbunching again

- Similarly to the banding that you can have with the LCG, poor choices of the radices can give rise to banding, depending on the particle number.
- A very important point is that if you populate e.g. a 6D distribution with samples, you should not do it pairwise in each plane:

$$\{x_j, x'_j\} \qquad \{y_j, y'_j\}$$

- otherwise there will be correlations between e.g. x_j and y_j . Instead, you should construct a joint sequence of $\{x_j, x'_j, y_j, y'_j\}$

- The moral of the tale is to look at the distributions you are generating.

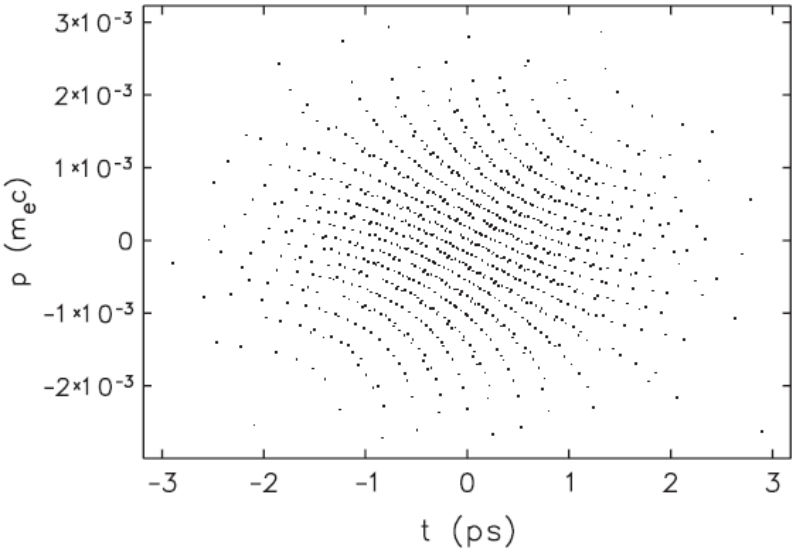


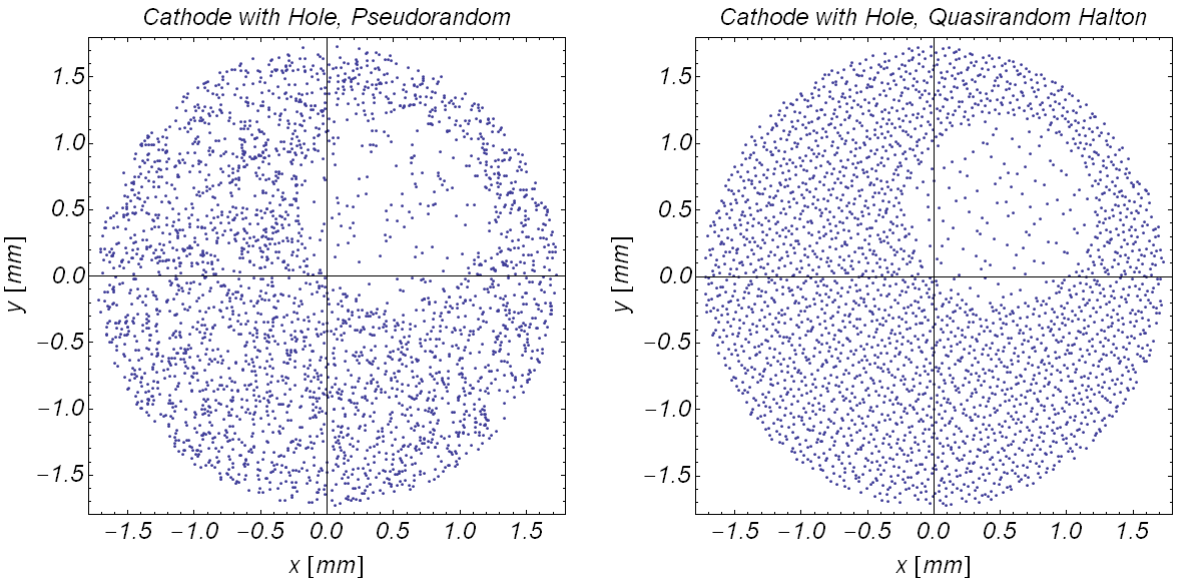
FIG. 2. Illustration of banding in longitudinal phase space when Halton radices of 11 and 13 are used for time and momentum coordinates, respectively. Banding becomes less evident as the number of particles is increased and when the ratio of the radices is far from unity.

M. Borland, “Modeling of the microbunching instability" *Phys. Rev. S.T.A.B.* **11**, 030701 (2008)

Other quasirandom sequences

- There are lots:
- Hammersley
- Faure
- Neiderreiter
- Sobol
- Scrambled van der Corput/Halton

- I've created a Mathematica notebook that you can play with to get an idea of how they all work.



Summary of Random Number Generation

- Random numbers are a big field in themselves, with many issues unresolved.
- Depending on the simulation you are doing, different types of distribution may be needed.
- The LCGs in most compilers have a number of deficiencies that are important in accelerator simulations.
- More generally, you should be aware of the issues in using distributions, both pseudorandom and quasirandom.