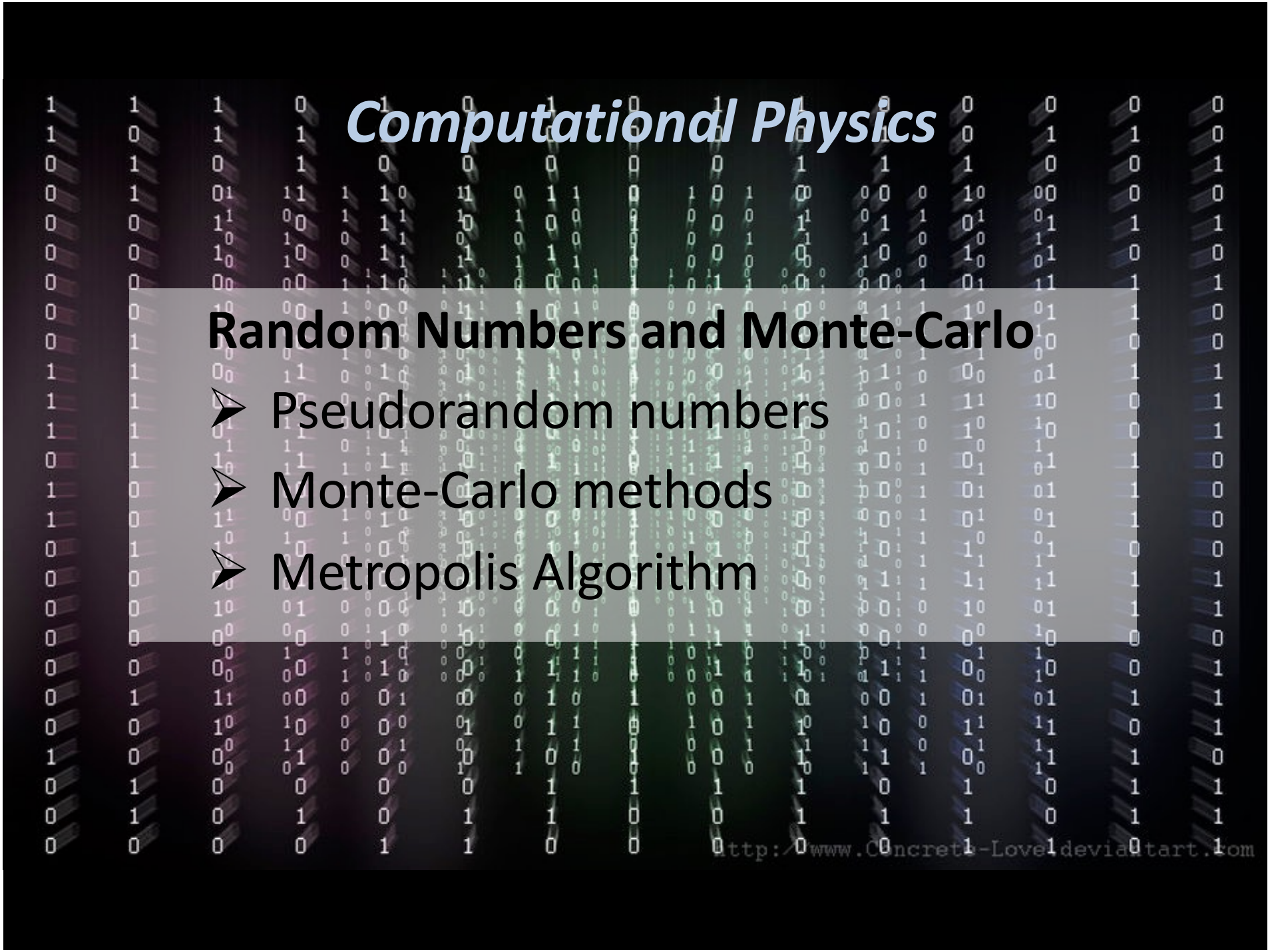


Computational Physics

Random Numbers and Monte-Carlo

- Pseudorandom numbers
- Monte-Carlo methods
- Metropolis Algorithm

Introduction

- Stochastic methods in Computational Physics are based on **random numbers** and on the concepts of probability theory.
- example of randomness in physical systems is certainly the outcome of a dice-throw or the drawing of lotto numbers.
- random numbers generated by the above methods are effectively unpredictable
- Another example is Brownian motion or diffusion which describes the random motion of particles
- In stochastic Computational Physics methods, one typically needs a lot of random numbers, which must be generated fast
- true random numbers are usually slowly generated → we need to use algorithms.

Random numbers

- Utopia
  - True random generators (TRNG): exhibiting "true" randomness, such as the time between "tics" from a Geiger counter exposed to a radioactive element
    - Hard to find
    - Hard to proof
    - Complex implementation
- Reality
  - Pseudo random number generators (PRNG)
    - Sequences having the appearance of randomness, but nevertheless exhibiting a specific, repeatable pattern.
    - numbers calculated by a computer through a deterministic process, cannot, by definition, be random

"*Any one who consider arithmetical methods of producing random digits is, of course, in a state of sin.*"

John von Neumann [1951]

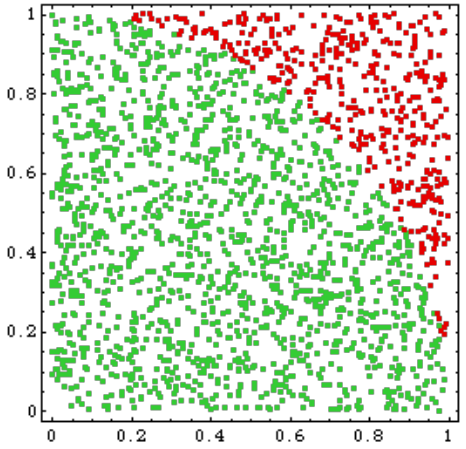
Physical (True?) RNG

- Radioactive decay
- Air Turbulence in disk drives
- Lava lamp
  e.g., http://www.lavarnd.org (not original anymore)
- http://www.random.org
- hardware random number generator (HRNG): Intel 8xx chipset, now available on most CPUs

- Timing of keystrokes when a user enters a password.
- Measurement of timing skew between two systems timers:
  - A hardware timer
  - A software timer

(Pseudo) Random number generators ([P]RNG)

- **Desirable Attributes:**
  - **Uniformity**
  - **Independence**
  - **Efficiency & Parallelizability**
  - **Replicability**
  - **Long Cycle Length**
- Needed for:
  - Numerical Algorithms
  - Simulations
  - "Monte-Carlo" Methods
  - encryption
- Each random number x_i is an independent sample drawn from a continuous uniform distribution between 0 and 1



Example: calculation of π using MC

$$\text{pdf:} \quad f(x) = \begin{cases} 1\,, & 0 \le x \le 1 \\ 0\,, & \text{else} \end{cases}$$

PRNG algorithms

Remember:

- Given knowledge of the algorithm used to create the numbers and its internal state (i.e. seed), you can predict all the numbers returned by subsequent calls to the algorithm, whereas with genuinely random numbers, knowledge of one number or an arbitrarily long sequence of numbers is of no use whatsoever in predicting the next number to be generated.
- Computer-generated "random" numbers are more properly referred to as *pseudorandom numbers*, and *pseudorandom sequences* of such numbers.

Example: Linear congruential generators (LCG)

- Linear Congruential Method:
  - Basic generator
    $X_{n+1} = (a\, X_n + c) \pmod m$,
  - With modulus $m \geq 0$, multiplier $m > a > 0$, increment $0 \leq c < m$
  - Most natural choice for m is one that equals to the capacity of a computer integer type used.
  - $m = 2^b$ (binary machine), where b is the number of bits in the integer type.
  - $m = 10^d$ (decimal machine), where d is the number of digits in the integer type.
  - X_0 is called the seed

...

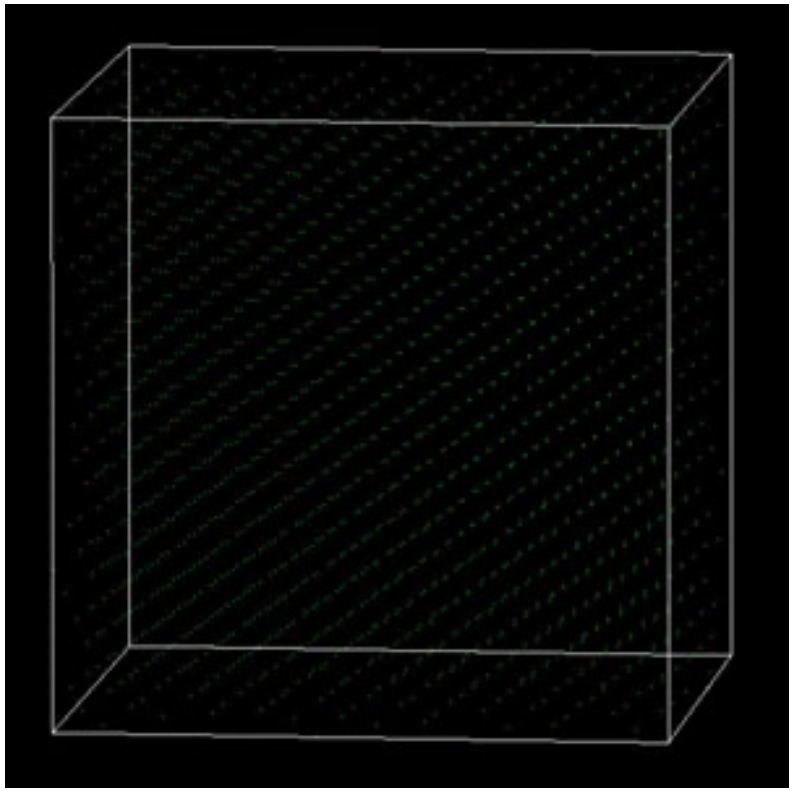
- The appearance of randomness is provided by performing modulo arithmetic or remaindering
- With X_n determined, we generate a corresponding real number as follows:
  $R_n=X_n/\text{float}(m)$ or $R_n=X_n/\text{float}(m+1)$
- When dividing by m, the values, R_n, are then distributed on [0,1).
- We desire uniformity, where any particular R_n is just as likely to appear as any other R_n, and the average of the R_n is very close to 0.5.
- Again: the next result depends upon only the previous integer – This is a characteristic of linear, congruential generators which minimizes storage requirements, but at the same time, imposes restrictions on the period.

Advantages/Disadvantages

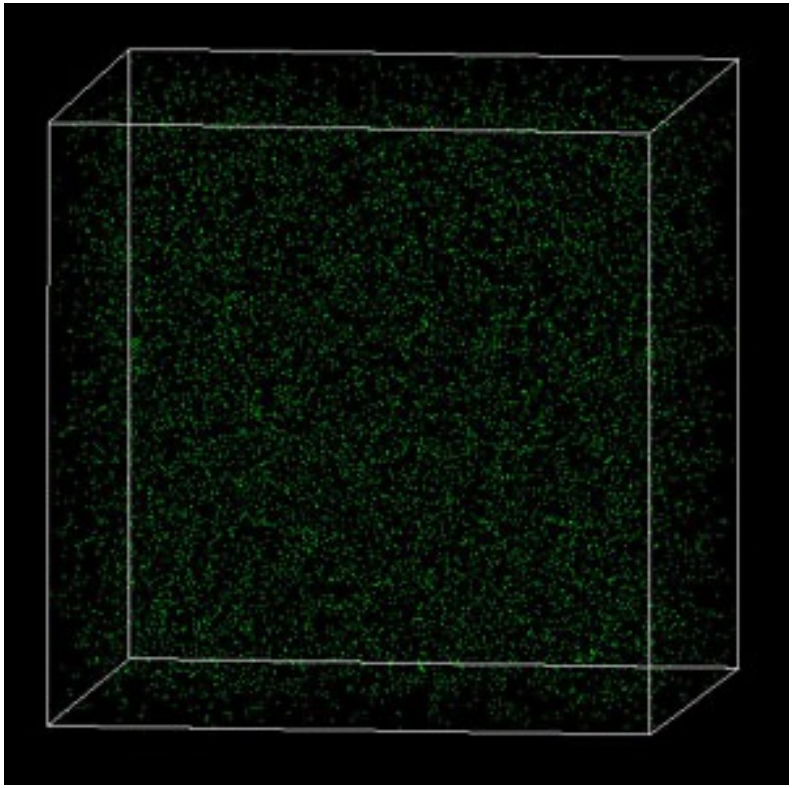
LCGs:

- fast and require minimal memory (typically 32 or 64 bits) to retain state
- valuable for simulating multiple independent streams
- should not be used for applications where high-quality randomness is critical
- not suitable for a Monte Carlo simulation because of the serial correlation (among other things)
- should also not be used for cryptographic applications
- LCGs tend to exhibit some severe defects:
  - For instance, if an LCG is used to choose points in an n-dimensional space, the points will lie on, at most, $m^{1/n}$ hyperplanes (Marsaglia's Theorem, developed by George Marsaglia). This is due to serial correlation between successive values of the sequence x_i. The spectral test, which is a simple test of an LCG's quality, is based on this fact.

LCG problem



Points fall on planes



Ideal random points

concrete LCGs and other examples

- **LCG**: PARK- MILLER parameters: $a=5^7$, $c=0$, $m=2^{31}-1$
- **LCG + shuffling**: generate a random array and access elements randomly
- Use more previous numbers (instead of one in LCG):

$$x_{n+1} = \left(\sum_{k=0}^{\ell} a_k x_{n-k} \right) \bmod m$$

- e.g., **Fibonacci generator**: $x_{n+1} = (x_n + x_{n-1}) \bmod m$
- **lagged Fibonacci generator**: replace "+" by any binary operation, such as addition, subtraction, multiplication or some logical operation
  - *shift register generator*: typical cycle 10^{75}
  - *MARSAGLIA-ZAMAN generator*: typical cycle 10^{171}

- **Mersenne Twister** – fast, negligible serial correlation, good for MC, cycle = $2^{19937}-1$ (used in python)

Read chapter 12 for more details and information

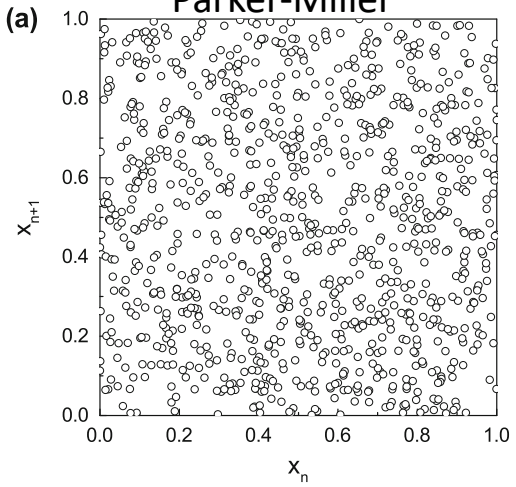
Testing PRNGs

- statistical tests:
  - calculate moments: $\langle X^k \rangle = \int \mathrm{d}x\, x^k p(x) = \int_0^1 \mathrm{d}x\, x^k = \frac{1}{k+1}$
  - calculate correlations: $\langle X_n X_{n+k} \rangle = \langle X_n \rangle^2 = \frac{1}{4}$
  - calculate pdf directly by histograms →
  - χ^2 test: check if sum of ν squared random numbers follow χ^2-distribution (see book)

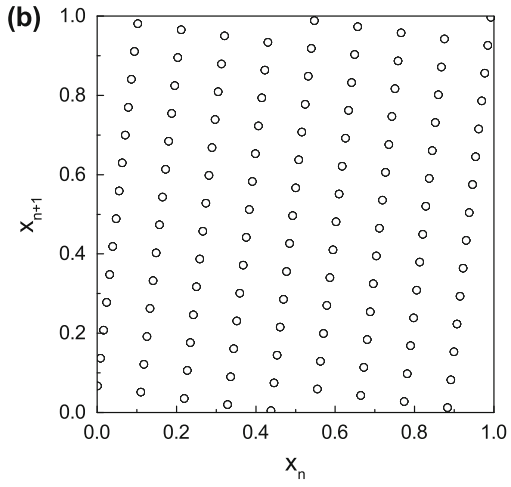
$$p(x;\nu) = \frac{x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)}, \quad x \geq 0$$

- graphical test (plot e.g. pairs of random numbers as points) ↓

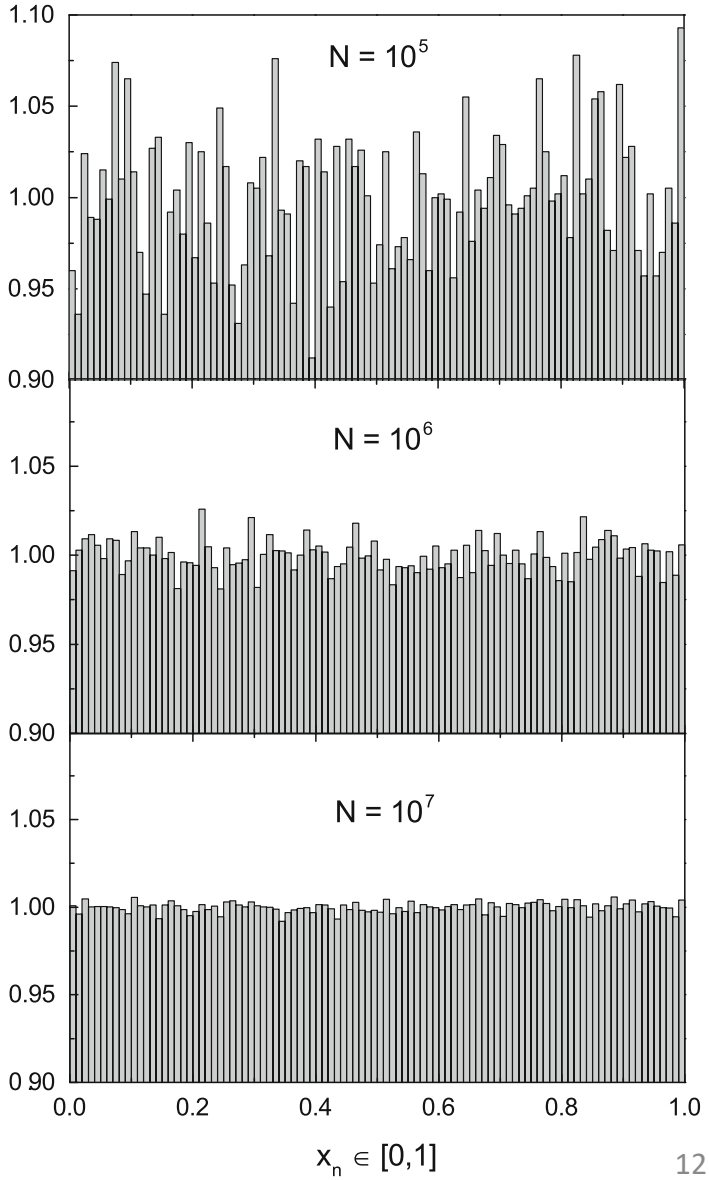
Parker-Miller



a=137, c=0, m=2^{11}



Parker-Miller



Monte-Carlo methods

- Monte Carlo (MC) methods have been used for centuries *(statistical sampling)*.
- However, during World War II, this method was used to simulate the probabilistic issues with neutron diffusion (first real use).
- modern version of the Monte Carlo method invented in the late 1940s by Stanislaw Ulam, while working on nuclear weapons projects at the Los Alamos National Laboratory
- named by Nicholas Metropolis, after the Monte Carlo Casino, where Ulam's uncle often gambled

What is a MC method?

- *Non-Monte Carlo* methods typically involve ODE/PDE equations that describe the system.
- Monte Carlo method refers to any method that makes use of random numbers
  - Simulation of natural phenomena
  - Simulation of experimental apparatus
  - Numerical analysis
- Monte Carlo methods are stochastic techniques.
- It is based on the use of random numbers and probability statistics to simulate problems.

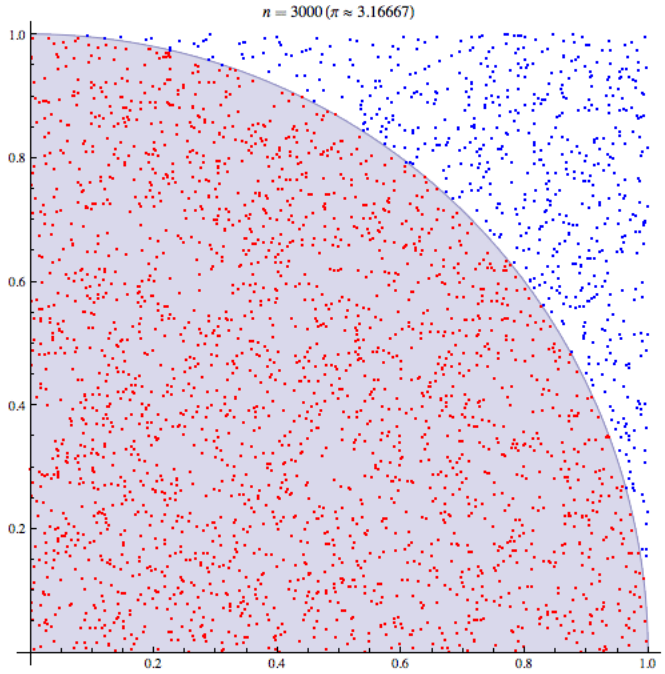
Why is MC used?

- It allows us to examine complex systems. And is usually easy to formulate (independent of the problem).
- For example, solving equations which describe two atom interactions. This would be doable without using Monte Carlo method. But solving the interactions for thousands of atoms using the same equations is impossible.
- However, the solutions are imprecise and it can be very slow if higher precision is desired.

Simple Example 1: π

Consider a circle inscribed in a unit square: the circle and the square have a ratio of areas that is π/4 → the value of π can be approximated using a Monte Carlo method:

- Draw a square, then inscribe a circle within it
- Uniformly scatter some objects of uniform size (grains of rice or sand) over the square.
- Count the number of objects inside the circle and the total number of objects.
- The ratio of the two counts is an estimate of the ratio of the two areas, which is π/4. Multiply the result by 4 to estimate π.



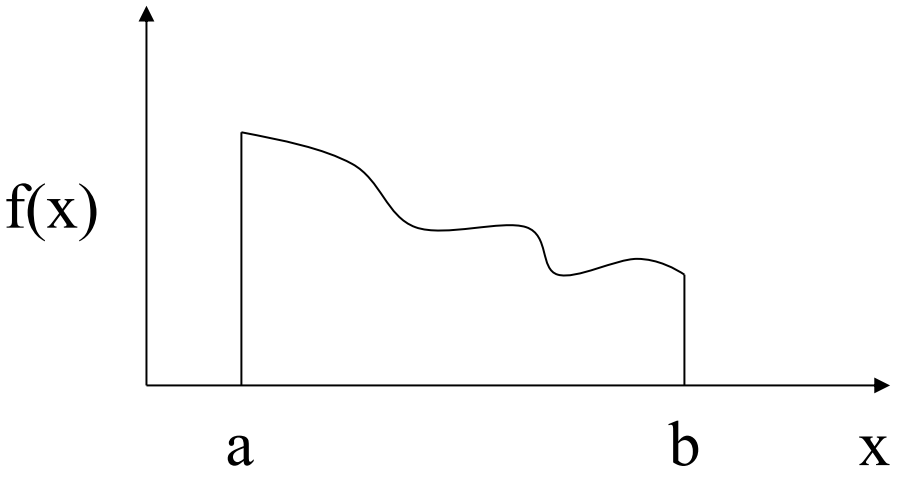
Simple Example 2: dice

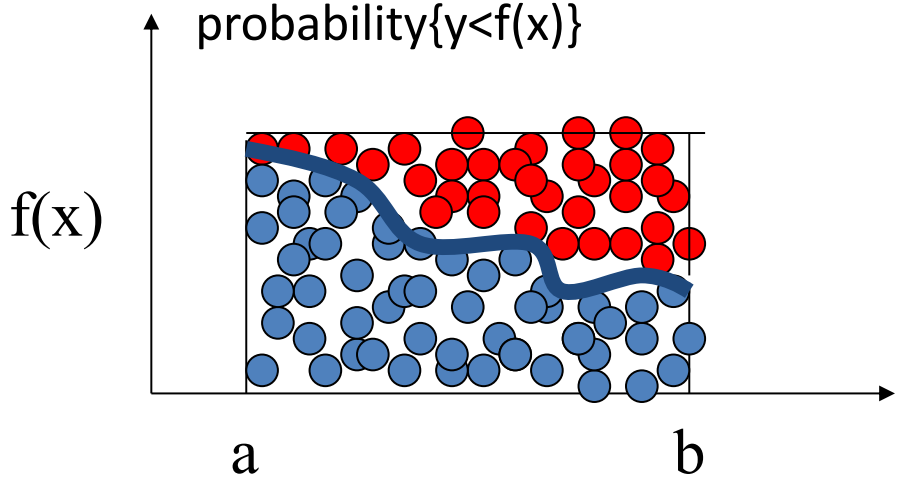
- **Problem**: What is the probability that 10 dice throws add up exactly to 32?
- **Exact Way.** Calculate this exactly by counting all possible ways of making 32 from 10 dice.
- **Approximate (Lazy) Way.** Simulate throwing the dice (say 500 times), count the number of times the results add up to 32, and divide this by 500.
- **Lazy Way can get quite close to the correct answer quite quickly.**

Simple Example 3: integration

- Method 1: Analytical Integration
- Method 2: Numerical, e.g., Quadratures
- Method 3: MC – random sampling the area enclosed by a<x<b and 0<y<max (f(x)): *hit and miss integration*

$$\int_a^b f(x)\,dx \approx \max(f(x))(b-a)\left(\frac{\#\bullet}{\#\bullet+\#\bullet}\right)$$





...

General case: integrate function over complicated region G

- Pick a simple (e.g. rectangular) region G'
- Sample N' random points over G'
- Count points in G: N

$$\frac{\mathrm{vol}(G)}{\mathrm{vol}(G')} = \frac{N}{N'}$$

Mean-value integration

Other way of MC integration is based on the mean-value theorem for a continuous function f(x) in x ∈[a,b], which states that a z∈[a,b] exists that:

$$\int_a^b \mathrm{d}x f(x) = f(z)(b-a)$$

where $f(z) \equiv \langle f \rangle$ is the mean or expectation value of $f(x)$.
Therefore:

$$\frac{1}{b-a}\int_a^b \mathrm{d}x' f(x') \simeq \bar{f} \pm \sqrt{\frac{\overline{f^2} - \bar{f}^2}{N}}$$

error, vanishes for N→∞

where we calculate from uniform random $x_i \in [a,b]$:

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \qquad \overline{f^2} = \frac{1}{N}\sum_{i=1}^{N} f^2(x_i)$$

… more general

The expectation value is in general defined as

$$\langle f \rangle = \int \mathrm{d}x f(x) p(x)$$

where $x \in \mathbb{R}^d$ and p(x) is a pdf
A typical example is the calculation of the thermal expectation value in **statistical physics** where the pdf p(x) is given by the normalized BOLTZMANN distribution:

$$p(x) = \frac{1}{Z} \exp\left[-\frac{E(x)}{k_\mathrm{B} T}\right]$$

where E(x) is the energy, k_B the Boltzmann constant, T the temperature, and Z the canonical partition function (normalization)

Metropolis algorithm

The Metropolis algorithm is a more sophisticated method to produce random numbers from given distributions and is particularly useful to treat problems in statistical physics where thermodynamic expectation values of some observable O are of interest

$$\langle O \rangle = \int \mathrm{d}x O(x) q(x)$$

x is a set of parameters (e.g. coordinates and momenta of particle) and q(x) the Boltzmann distribution:

$$q(x) = \frac{p(x)}{Z} \qquad Z = \int \mathrm{d}x\, p(x)$$

The Metropolis algorithm now aims at generating sampling points x_i according to the pdf q(x), such that the expectation value can be evaluated.

rejection method

Suppose we already have n parameter sets: x_0, x_1,…, $x_n=\{x_n\}$ which follow q(x), and we need to decide if a new value x_t should be added:
The Metropolis method introduces the following acceptance probability for x_t:

$$\Pr(A|x_t,x_n)=\begin{cases} 1 & \text{if } \dfrac{q(x_t)}{q(x_n)}\geq 1 \\ \dfrac{q(x_t)}{q(x_n)} & \text{otherwise.} \end{cases}$$

(see also chapter 13)

Hence, if $\Pr(A|x_t,x_n)=1$, we set $x_{n+1}=x_t$, and if $\Pr(A|x_t,x_n)<1$, we draw a random number $r\in[0,1]$ and accept x_t if $r<\Pr(A|x_t,x_n)$ and reject x_t otherwise.

or more compact: $$\Pr(A|x_t,x_n)=\min\left(\frac{p(x_t)}{p(x_n)},1\right)=p(x_t|x_n)$$

The underlying reason for this to work requires knowledge of stochastics in general and of MARKOV-chains in particular.

(see also chapter 16)

We apply this to the Ising model, which illustrates this at a concrete example

Example of a simulation process
(e.g. Ising model)

- Initialize the system
  - Put the system in a random state
- Make a trial move
  - Randomly make a trial move
- Calculate the energy change
  - Reevaluate the interactions of the moved particles with its neighbors and calculate the energy change
- Accept the trial move with the Metropolis scheme

$$P = \begin{cases} \exp\left(-\dfrac{\Delta E}{k_B T}\right) & \Delta E > 0 \\ 1 & \Delta E < 0 \end{cases}$$

- Keep trying the moves until system approach equilibrium
  - Either monitor the total energy change, or monitor the structure formed in the simulation box
- Sampling
  - Sample a certain property over a certain number of configurations

Homework/lab task

- Implement the iterative Poisson solver for arbitrary charge density $\rho(x,y)$:
  - Reproduce the results for mono, di-, and quadropole
  - Solve for two separated oppositely charged monopoles.
  - Calculate the electric field E(x,y) by calculating the negative gradient of $\varphi(x,y)$ [central difference]
- Implement a time-dependent heat equation solver and solve for fixed end temperatures and initial $T_i=0$ to evolve the temperature profile in time using the same conditions as for the stationary case.
- **Implement MC π calculation**