Prof. Dr. rer. nat. habil. Martin O. Steinhauser

Frankfurt University of Applied Sciences, Germany

Faculty of Computer Science and Engineering

Short Lecture Course:

Introduction to Computational Science with Applications in Molecular Dynamics

Session 7: The Monte-Carlo Method / Random Numbers

Session 7: Overview

OUTLINE OF LECTURE

- What are Random Numbers?
- What is the Monte Carlo Method?

- **Handout 5-7:** Original (historical) Publications on the MC-Method
- **Handout 8:** The MC-Method (skript)
- **Lecture Skript**: JapanLecture_MonteCarloMethod_Skript.pdf

To download lecture material, please go to Github:

https://github.com/Kosmokrat/JapanLecture2024

Overview of this short course

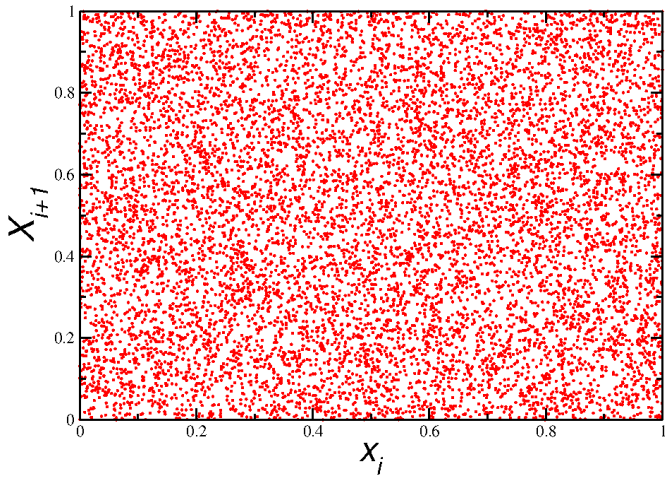
Topics Covered (subject to change)

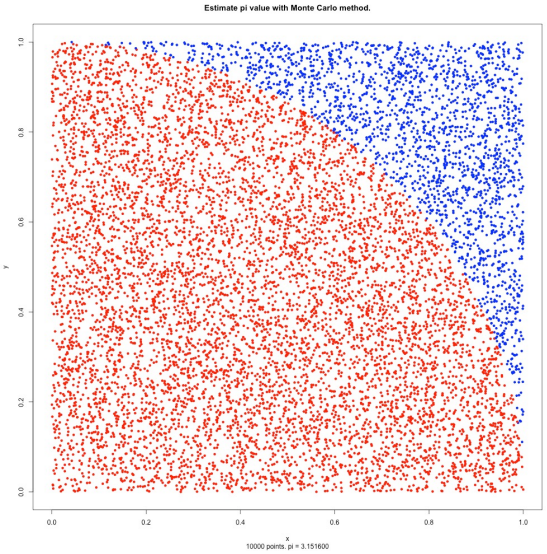
1st Session: Lec. 1-2		Introduction & Bits and Bytes
2nd Session: Lec 3	(2x)	Bits and Bytes continued
3rd Session: Lec 4-6		Molecular Dynamics
4th Session: Lec 7-8		MD continued / Algorithms
5th Session: Lec 9	(2x)	Algorithms/ Problem of Sorting
6th Session: Lec 10-11		Asymptotic Analysis of Algorithms
7th Session: Lec 12-13		Monte Carlo/Random Numbers

Overview of Presentation

Overview of this Lecture

1	The Monte Carlo Method: Random Numbers
2	The Monte Carlo Method: Introduction
3	The Monte Carlo Method: Simple and Importance Sampling





What is Monte Carlo? Some Remarks

- Named at Los Alamos in the 1940‘s
- MC is any Method which uses (Pseudo-)Random Numbers as an essential part of the algorithm

  → Stochastic, not Deterministic (like Molecular Dynamics)

- It is a Method for doing highly-dimensional integrals by sampling the integrand.
- MC is often a Markov Chain, called Metropolis MC

Historical Perspective: Handouts 5-7

COMPUTING PRACTICES

Edgar H. Sibley
Panel Editor

Practical and theoretical issues are presented concerning the design, implementation, and use of a good, minimal standard random number generator that will port to virtually all systems.

RANDOM NUMBER GENERATORS: GOOD ONES ARE HARD TO FIND

STEPHEN K. PARK AND KEITH W. MILLER

An important utility that digital computer systems should provide is the ability to generate random numbers. Certainly this is true in scientific computing where many years of experience has demonstrated the importance of access to a good random number generator. And in a wider sense, largely due to the encyclopedic efforts of Donald Knuth [18], there is now a realization that random number generation is a concept of fundamental importance in many different areas of computer science. Despite that, the widespread adoption of good, portable, *industry standard* software for random number generation has proven to be an elusive goal. Many generators have been written, most of them have demonstrably non-random characteristics, and some are embarrassingly bad. In fact, the current state of random number generation software is accurately described by Knuth [18, p. 176] who advises "... look at the subroutine library of each computer installation in your organization, and replace the random number generators by good ones. Try to avoid being too shocked at what you find."

Knuth's advice applies equally well to most recently published computer science textbooks, particularly those written for the undergraduate market. Indeed, during the preparation of this article we reviewed more than 50 computer science textbooks that contained software for at least one random number generator. Most of these generators are unsatisfactory.

This article was motivated by practical software considerations developed over a period of several years while teaching a graduate level course in simulation. Students taking this course work on a variety of systems and their choices typically run the gamut from personal computers to mainframes. With Knuth's advice in mind, one important objective of this course is for all students to write and use implementations of a good, minimal standard random number generator that will port to *all* systems. For reasons discussed later, this minimal standard is a multiplicative linear congruential generator [18, p. 10] with multiplier 16807 and prime modulus $2^{31} - 1$. As it turns out, porting this random number generator (or any other for that matter) to a wide variety of systems is not as easy as it may seem. The issues involved are discussed later in this article.

The body of this article is organized into four sections. In the first, we present the rationale for our choice of a minimal standard generator. We believe that this is the generator that should always be used—unless one has access to a random number generator *known* to be better. In the second section we demonstrate how to implement the minimal standard in a high-level language on a variety of systems. The third section presents theoretical considerations and implementation details in support of the discussion in the previous sections. Finally, in the last section, we present selected examples of unsatisfactory generators that have either appeared in recently published (post-1980) computer science textbooks or are currently supplied by popular programming environments.

PEOPLE

Metropolis, Monte Carlo, and the MANIAC

by Herbert L. Anderson

In 1942 Nick Metropolis was working with Edward Teller on the reactor project at the University of Chicago when J. Robert Oppenheimer invited the young physicist to continue his collaboration with Teller, but at Los Alamos. There Metropolis joined the Manhattan Project as a member of the Theoretical Division, having been encouraged by Teller to move from experimental to theoretical physics. His first assignment was to develop equations of state for materials at high temperatures, pressures, and densities.

Over the years Metropolis turned increasingly to mathematics and computer design, and by 1948 he was leader of a Los Alamos team that designed and built the MANIAC, one of the first electronic digital computers. Many of the country's foremost scientists were eager to try their experiments on the wonderful new machine and came to the Laboratory to work with Metropolis. A few years later, together with Teller, John von Neumann, Stanislaw Ulam, and Robert Richtmyer, Metropolis developed techniques and algorithms for using the Monte Carlo method (so named by Metropolis) on the new computers.

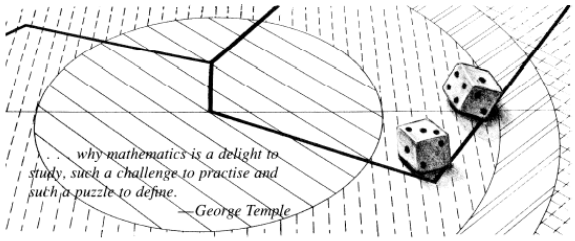
The Monte Carlo method is an application of the laws of probability and statistics to the natural sciences. The essence of the method is to use various distributions of random numbers, each distribution reflecting a particular process in a sequence of processes such as the diffusion of neutrons in various materials, to calculate samples that approximate the real diffusion history. Statistical sampling had been known for some time, but without computers the process of making the calculations was so laborious that the method was seldom used unless the need was compelling. The computer made the approach extremely useful for many physics problems.

Metropolis was also involved in the development of an importance-sampling scheme, called the Metropolis algorithm, that improves the effectiveness of the Monte Carlo method. In the past twenty years his work has included nonlinear problems and combinatorial theory as well as Monte Carlo calculation. He was named a Senior Fellow of the Laboratory in 1980.

In September 1985 more than one hundred researchers from around the world met in Los Alamos for a four-day conference, in honor of Nick Metropolis, on the frontiers of quantum Monte Carlo. One of the speakers was Herb Anderson, from the Laboratory Physics Division. Anderson's presentation was a fascinating reminiscence about the first modern calculating machines and the scientists who used them, about the intellectual ferment in the physics community that began early in this century and still continues, and about Nick Metropolis and the MANIAC. This article is adapted from Anderson's presentation.

THE BEGINNING *of the* MONTE CARLO METHOD

by N. Metropolis



. . . why mathematics is a delight to study, such a challenge to practise and such a puzzle to define.
—George Temple

The year was 1945. Two earth-shaking events took place: the successful test at Alamogordo and the building of the first electronic computer. Their combined impact was to modify qualitatively the nature of global interactions between Russia and the West. No less perturbative were the changes wrought in all of academic research and in applied science. On a less grand scale these events brought about a renascence of a mathematical technique known to the old guard as statistical sampling; in its new surroundings and owing to its nature, there was no denying its new name of the Monte Carlo method.

This essay attempts to describe the details that led to this renascence and the roles played by the various actors. It is appropriate that it appears in an issue dedicated to Stan Ulam.

Some Background

Most of us have grown so blase about computer developments and capabilities -even some that are spectacular—that it is difficult to believe or imagine there was a time when we suffered the noisy, painstakingly slow, electromechanical devices that chomped away on punched cards. Their saving grace was that they continued working around the clock, except for maintenance and occasional repair (such as removing a dust particle from a relay gap). But these machines helped enormously with the routine, relatively simple calculations that led to Hiroshima.

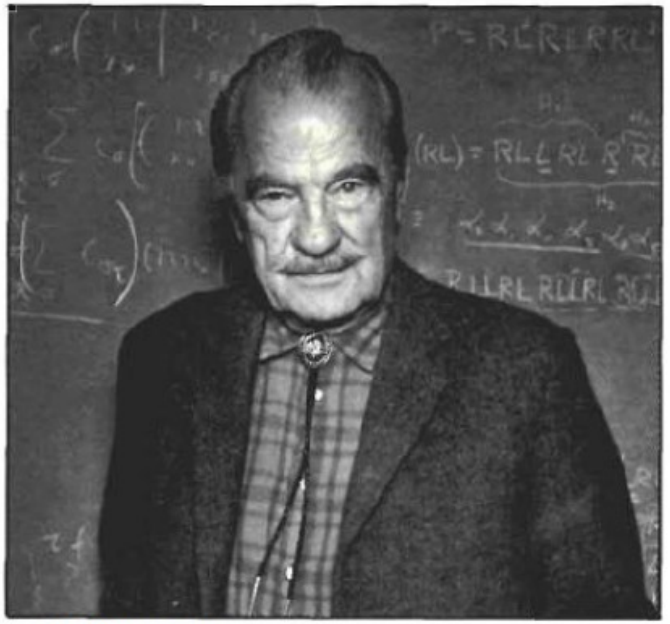
The ENIAC. During this wartime period, a team of scientists, engineers, and technicians was working furiously on the first electronic computer—the ENIAC—at the University of Pennsylvania in Philadelphia. Their mentors were Physicist First Class John Mauchly and Brilliant Engineer Presper Eckert. Mauchly, familiar with Geiger counters in physics laboratories, had realized that if electronic circuits could count, then they could do arithmetic and hence solve, *inter alia*, difference equations—at almost incredible speeds! When he'd seen a seemingly limitless array of women cranking out firing tables with desk calculators, he'd been inspired to propose to the Ballistics Research Laboratory at Aberdeen that an electronic computer be built to deal with these calculations.

John von Neumann, Professor of Mathematics at the Institute for Advanced Study, was a consultant to Aberdeen and to Los Alamos. For a whole host of

The Monte Carlo Method: Researchers involved in Development

Stanislaw Ulam

John von Neumann



N.Metropolis

Some Historical Remarks

- ✓ First paper that coined the term “Monte Carlo”
  “The Monte Carlo Method”, N. Metropolis and S. Ulam,
  *Journal of the American Statistical Association, **44**, 335-341 1949*

- ✓ Importance Sampling (Metropolis Algorithm) was introduced in 1953
  “Equation of State Calculations by Fast Computing Machines”,
  N. Metropolis, A. W. Rosenbluth, M.N. Rosenbluth and A. H. Teller,
  *J. Chem. Phys., **21**, 1087-1092 1953*

- ✓ Metropolis algorithm is the first among the top 10 algorithms of the 20th century
  “Top Ten Algorithms of the Century”, J. Dongarra, F. Sullivan,
  *Comput. Sci. Eng., **2**, 22-23 2000*

- ✓ 70th Anniversary of the Metropolis MC Algorithm in 2023 !

Original Publication: The Metropolis Algorithm

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed, only two-body forces are considered, and the potential field of a molecule is assumed spherically symmetric. These are the usual assumptions made in theories of liquids. Subject to the above assumptions, the method is not restricted to any range of temperature or density. This paper will also present results of a preliminary two-dimensional calculation for the rigid-sphere system. Work on the two-dimensional case with a Lennard-Jones potential is in progress and will be reported in a later paper. Also, the problem in three dimensions is being investigated.

* Now at the Radiation Laboratory of the University of California, Livermore, California.

II. THE GENERAL METHOD FOR AN ARBITRARY POTENTIAL BETWEEN THE PARTICLES

In order to reduce the problem to a feasible size for numerical work, we can, of course, consider only a finite number of particles. This number N may be as high as several hundred. Our system consists of a square† containing N particles. In order to minimize the surface effects we suppose the complete substance to be periodic, consisting of many such squares, each square containing N particles in the same configuration. Thus we define d_{AB}, the minimum distance between particles A and B, as the shortest distance between A and any of the particles B, of which there is one in each of the squares which comprise the complete substance. If we have a potential which falls off rapidly with distance, there will be at most one of the distances AB which can make a substantial contribution; hence we need consider only the minimum distance d_{AB}.

† We will use the two-dimensional nomenclature here since it is easier to visualize. The extension to three dimensions is obvious.

Historical Perspective

- Prior to computer simulations, liquids were modeled mechanically - large assemblies of macroscopic spheres or ball bearings (crude, and what about thermal motion?)

CHEMICAL PHYSICS

VOLUME 12, NUMBER 1 JANUARY, 1944

On the Statistical Mechanics of Liquids, and the Gas of Hard Elastic Spheres

O. K. RICE
University of North Carolina, Chapel Hill, North Carolina
(Received September 4, 1943)

Because of the extreme complications arising from a direct deductive approach, the theory of the liquid state requires the use of a model involving simplifying assumptions. In this paper an attempt is made to formulate general principles which any such model must follow. The first step is a general discussion of "communal" entropy, arising from the sharing of available space by all the atoms. Arguments are advanced to support the contention that for a gas of hard elastic spheres the communal entropy is fully excited in each direction of space, and amounts in all to $3R$ per mole. The communal entropy of assemblages of atoms exerting normal attractive and repulsive forces (in particular, the Debye solid) is considered. The geometry, the equation of state, and the partition function for an assemblage of hard elastic spheres are considered in detail. By extension of these ideas, allowing for the type of force actually exerted on each other by real atoms, a general form of partition function for a monatomic liquid is set up. This partition function involves a sum of two parts, one corresponding to a vibrational motion expressed in terms of the Debye characteristic temperature of the solid, and the other being a translational term, each part carrying with it its own communal entropy.

Some Historical Remarks

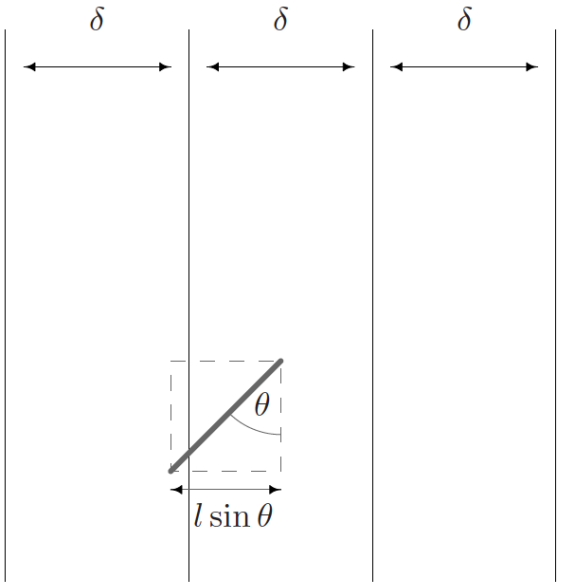
- ✓ The name derives from the association of statistical sampling methods with games of chance, such as those played in the casinos of Monte Carlo
- ✓ Although the name “Monte Carlo” was coined in 1949, statistical sampling methods have been around earlier than this. Sometime attributed to Fermi
- ✓ The full power of MC methods could be used only at the advent of mechanical calculating machines (first computers) at the late 1940s and in the 1950s.
- ✓ In the early 20th century, mechanical adding machines were (mis-)used for war purposes (firing tables for heavy artillery).
- ✓ During the Manhattan project in World War II, MC methods were used in the development of the hydrogen bomb on the world’s first ‘supercomputer’ ENIAC.
- ✓ With non-military use of supercomputers since the beginning 1950s, the MC techniques used on ENIAC and MANIAC were taken up by the scientific community.

Basic Idea of MC Integration: The Dartboard Method

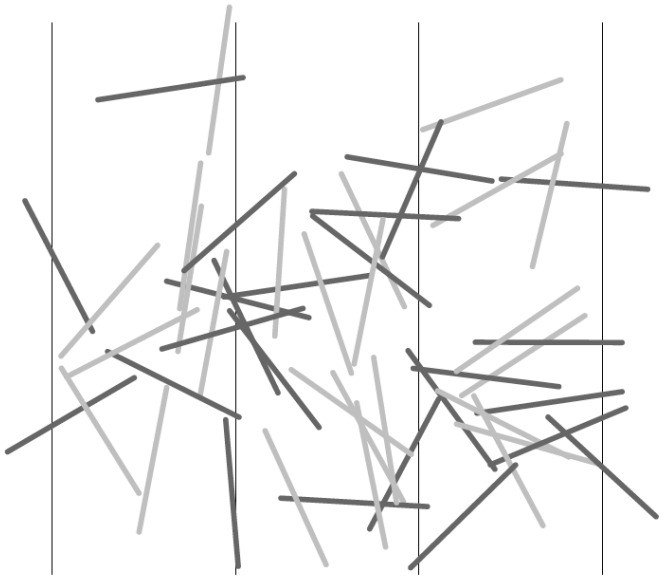
Whiteboard Notes

Early Example of MC Integration: Compte Buffon's Needle (1777)

Throw a needle of length l, on a grid of lines, distance d apart, with $l<d$

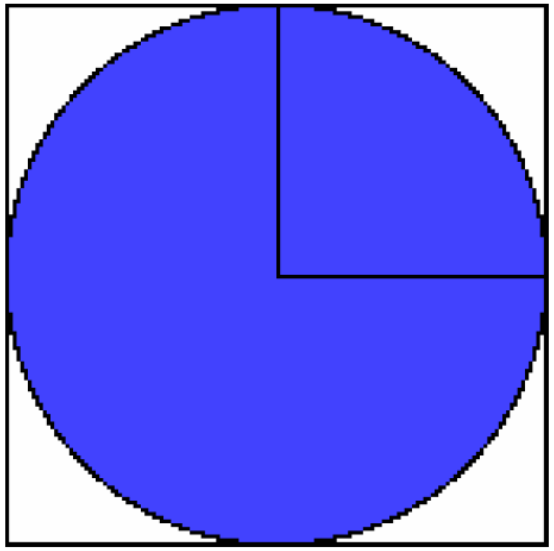


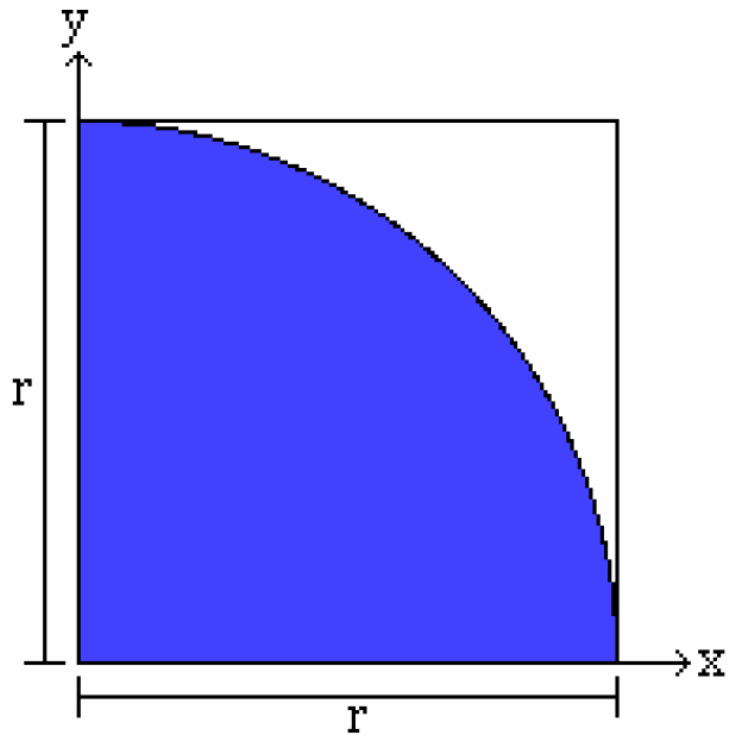
(a) Illustration of the geometry behind *Buffon's needle*



(b) Results of the *Buffon's needle* experiment using 50 needles. Dark needles intersect the thin vertical lines, light needles do not.

Hands-On Example: How about some Pi(e)





Hands-On Example: How about some Pi(e)

Task:

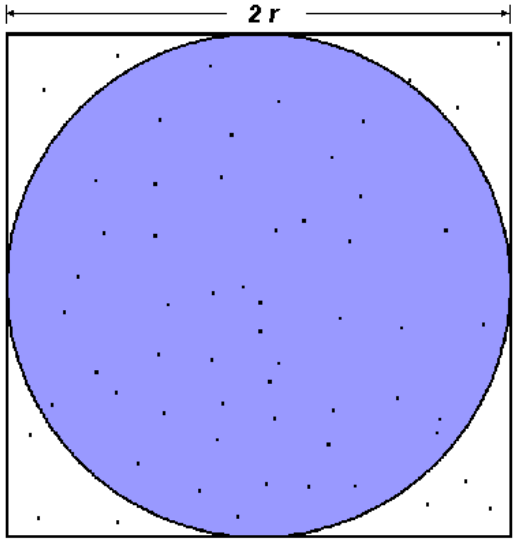
You have 5 Minutes to write a perfect, parallelized, beautiful program in C, which is error-free and bug-free and which calculates Pi according to the idea sketched on the previous slide to ten-digit precision in less than 1 second.

Hands-On Example: How about some Pi(e)

A Serial Program

ALGORITHM FOR CALCULATING π:

1. Inscribe a circle in a square.
2. Randomly generate points in the square.
3. Determine the number of points in the square that are also in the circle.
4. Let r be the number of points in the circle divided by the number of points in the square.
5. PI ~ 4 r
6. The more points generated, the better the approximation



$A_S = (2r)^2 = 4r^2$

$A_C = \pi r^2$

$\pi = 4 \times \frac{A_C}{A_S}$

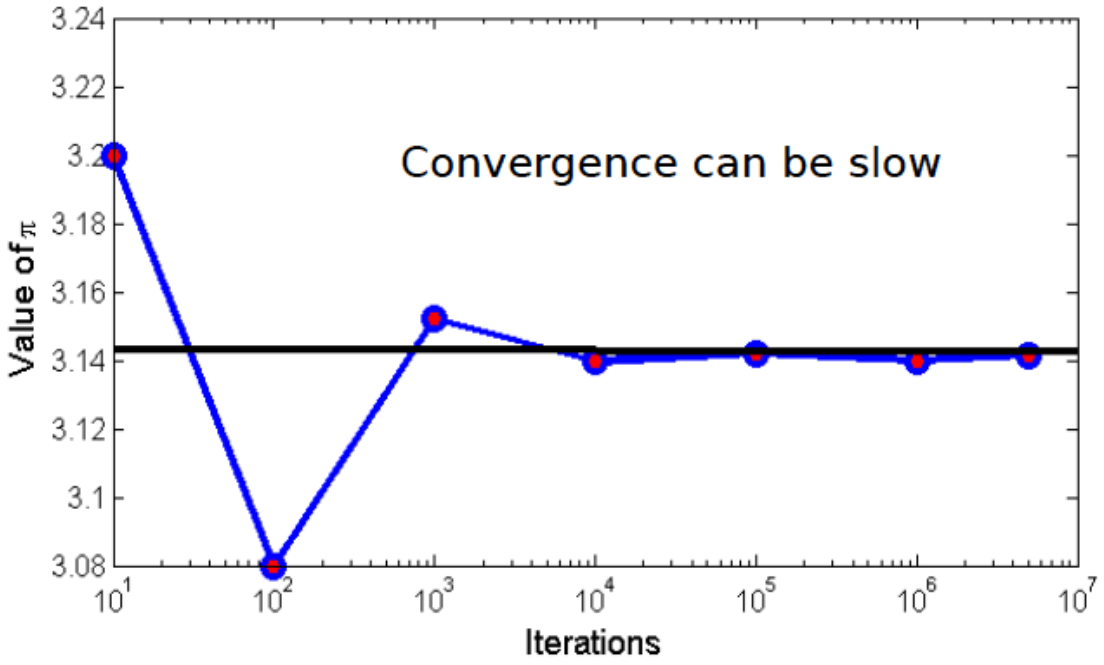
SERIAL PSEUDOCODE:

```
npoints = 10000
circle_count = 0

do j = 1,npoints
  generate 2 random numbers between 0 and 1
  xcoordinate = random1
  ycoordinate = random2
  if (xcoordinate, ycoordinate) inside circle
  then circle_count = circle_count + 1
end do

PI = 4.0*circle_count/npoints
```

EXAMPLE
Calculation of π: A Serial Program

ALGORITHM FOR CALCULATING π:

1. Inscribe a circle in a square.
2. Randomly generate points in the square.
3. Determine the number of points in the square that are also in the circle.
4. Let r be the number of points in the circle divided by the number of points in the square.
5. PI ~ 4 r
6. The more points generated, the better the approximation



Random Number Generation (RNG)

What is a random number?

Random Number Generation (RNG)

What is a random number?

- A single number is not random. Only an infinite sequence can be described as random
- Random means the absence of order. (Negative property).
- Can an intelligent gambler make money by betting on the next numbers that will turn up?
- All subsequences are equally distributed. This is the property that MC uses to do (high-dimensional) integrals.

Random Number Generation (RNG)

What is a random number?

- A single number is not random. Only an infinite sequence can be described as random
- Random means the absence of order. (Negative property).
- Can an intelligent gambler make money by betting on the next numbers that will turn up?
- All subsequences are equally distributed. This is the property that MC uses to do (high-dimensional) integrals.

„Random numbers should not be taken random.“ Knuth (1986)

Random Numbers on a Computer

- Truly random – the result of a physical process such as timing clocks, circuit noise, Geiger counts, bad memory
  - Too slow (we need 10^{10} / sec) and expensive
  - Low quality
  - Not reproducible

- Pseudo-random *prng* (pseudo means *fake*)
  - Deterministic sequence with a repeat period but with the appearance of randomness (if you don‘t know the algorithm)

- Quasi-random (quasi means *almost random*)
  - „half way“ between random and a uniform grid
  - Points being generated are always correlated

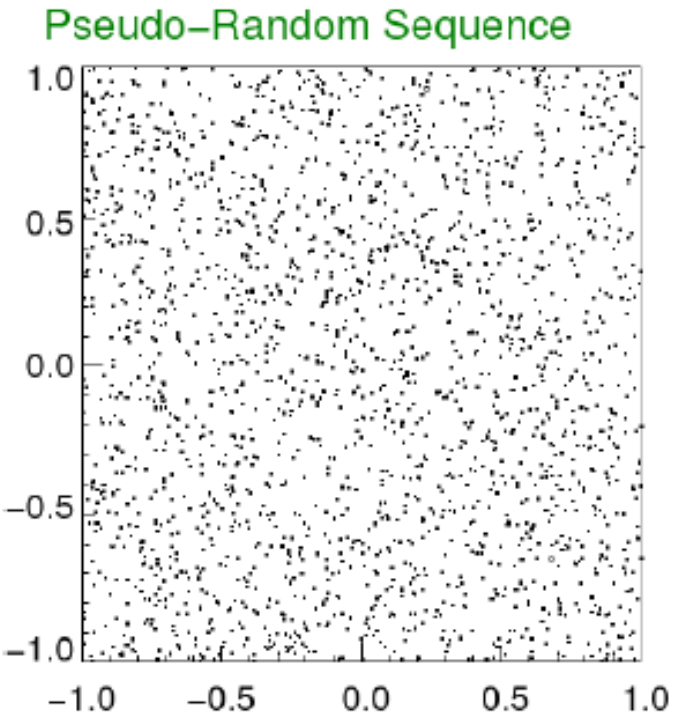
Pseudo- and Quasi-Random

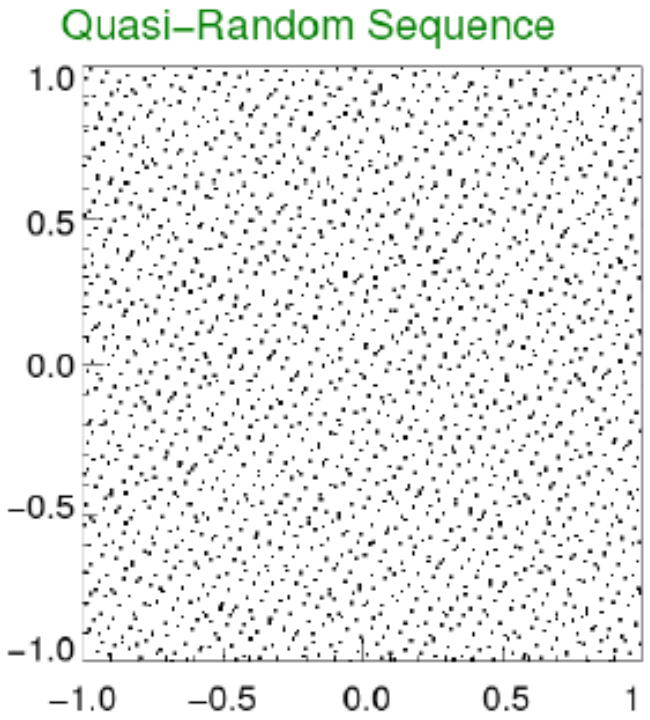
- LCG:

$$I_{i+1} = \left(I_i a + c \right) \bmod m$$

- Richtmyer: equally distributed point sets (p_k is the k-th prime)

$$I_n^k = \left(i \sqrt{p_k} \right) \bmod 1$$

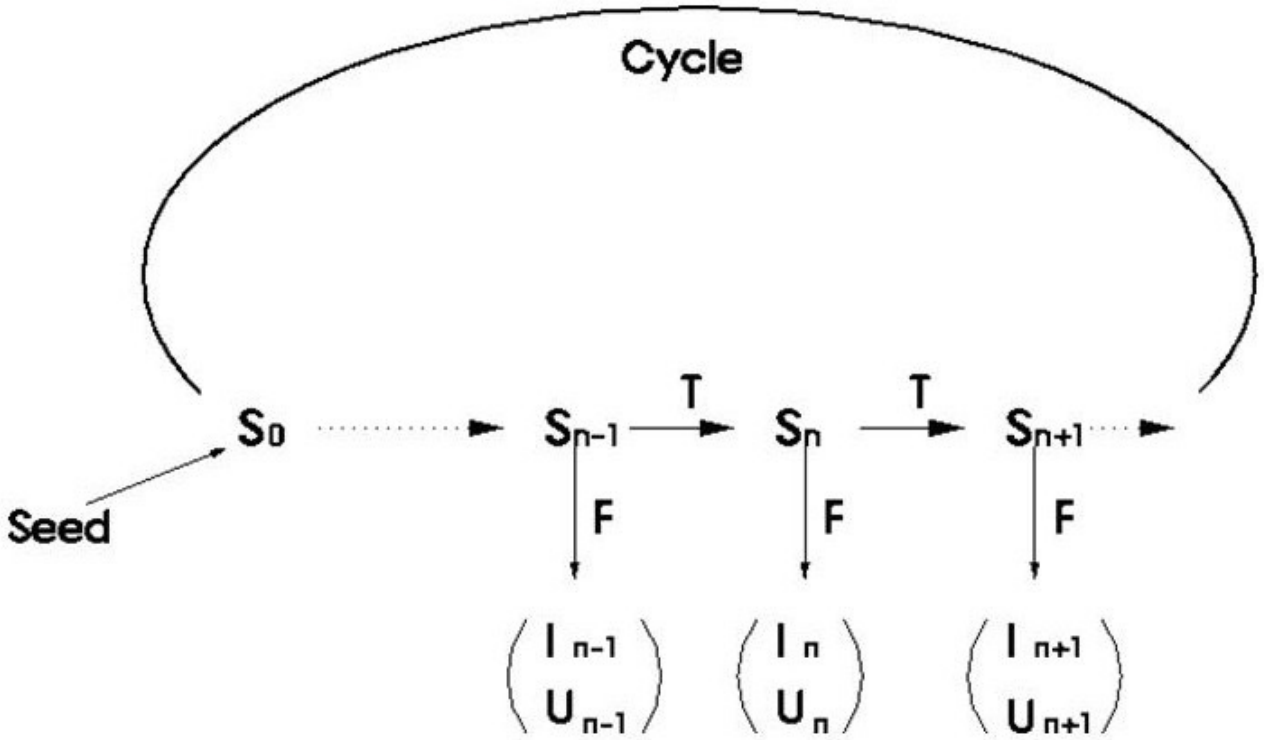




Desired Properties of RNG on a Computer

- Deterministic: easy to debug (repeatable simulations)
- Long Periods: cycle length is long
- Uniformity: RN generated in space evenly.
  - Go through complete cycle, and all integers occur once!
- Uncorrelated Sequences:
  $\langle f_1(x_{i+1}) \ldots f_k(x_{i+k}) \rangle = \langle f_1(x_{i+1}) \rangle \ldots \langle f_k(x_{i+k}) \rangle$
  - Monte Carlo can be very sensitive to such correlations, especially for large k
- Efficiency: Needs to be fast

Pseudo Random Sequence



S: State and initial seed.

T: Iteration process,

F: Mapping from state to integer RN (I) or real RN (U).

Common PRN Generators

Linear Congruential Generator (LCG) **modulo (mod) is remainder after division**

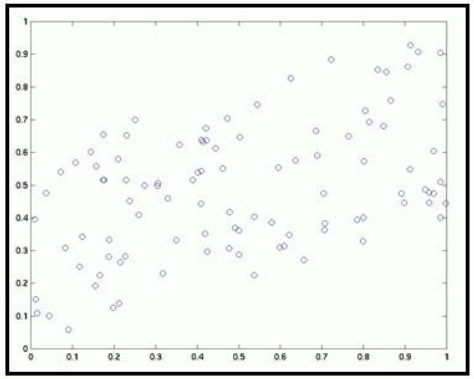
64-bit word (or 2 32-bit LCG) give a period of ~10^{18}.

	Recurrence	Parallelization
• Multiplicative Lagged Fibonacci • Modified Lagged Fibonacci	$z_n = z_{n-k} * z_{n-l}$ $z_n = z_{n-k} + z_{n-l}$ (modulo 2^m)	vary initialization
• 48 bit LCG • 64 bit LCG • Prime Modulus LCG	$z_n = a*z_{n-1} + p$ (modulo m)	vary p vary a
• Combined Multiple Recursive	$z_n = a_{n-1}*z_{n-1} + \ldots + a_{n-k}*z_{n-k}$ + LCG	vary LCG

Sequential RNG Problems

- Correlations ⇒ non-uniformity in higher dimensions

Uniform in 1-D but
non-uniform in 2-D



This is the important property to guarantee:

$$< f_1(x_{i+1}) f_2(x_{i+2}) > = < f_1(x_{i+1}) > < f_2(x_{i+2}) >$$

MC uses numbers many at a time--they need to be uniform.

e.g., 128 atom MC moving 3N coords. at random needs 4 PRNG (x,y,z, i) or

128 x 4=1024. So every 1024 moves may be correlated!

LCG Numbers Fall in Planes

Good LCG generators have the planes close together.

For a k-bit generator, do not use them more than k/2 together.

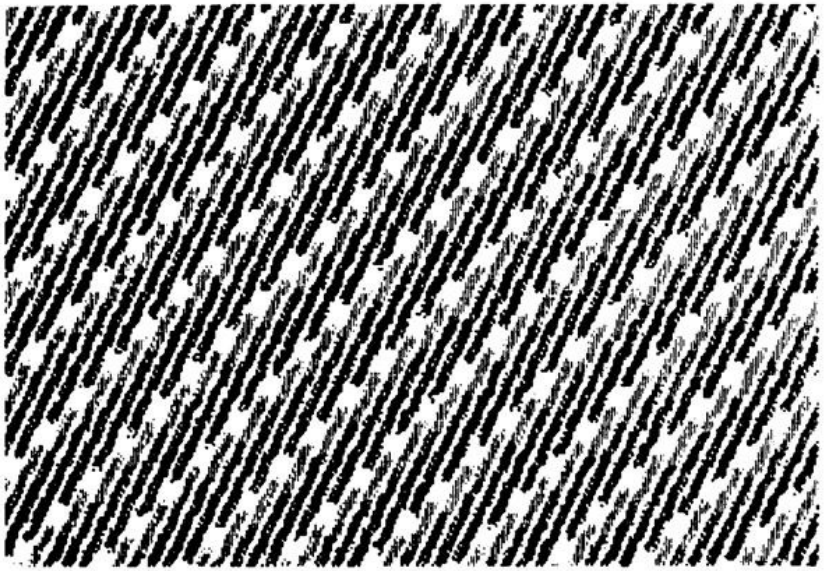


Fig. 8.2 Correlation between the triplets of points $\vec{r}_n$ $\{x_{3n}, x_{3n+1}, x_{3n+2}\}$ generated by a pseudorandom number generator. Tl value of the third coordinate is represented by the intensity of the point.

2/15/2013 13

Best: Use tested RNG from Libraries!

Some Notes

- **Linear Congruential Generator** (LCG); Period: ~10^{20}
- **IBM's** Park and Miller Generator; Period: $2^{31}-1$
- **UNIX's** drand48(); Period: $2^{48}=2.8 \times 10^{14}$
- **Mersenne Twistor** (uses bit shuffling and register shifts)
  Developed by Matsumoto and Nishimura (1997); a modification of a generalized shift register, uses 624 32-bit integers as internal storage. Uses bit shuffling and merging. Period is Huge: $2^{19937}-1$. Properties proven to be good in up to 623 dimensions. Also fairly fast.
- Combined Generators (e.g. Bays-Durham ran2() in Numerical Receipes in C)

Practical Aspects – Which Generator to Use ?

NEVER, NEVER, NEVER use a black box (for anything!) !
This is just bad for many reasons

Never try to „improve“ any generator by fiddling the parameters

Take care with initialization (seeding)

Use a well-known and tested generator
e.g. Park and Miller for uniform numbers
e.g. Bays-Durham for Gaussian distributions

ALWAYS test your results with (at least) 2 **different** generators !

Overview of Presentation

Statistical Mechanics and Thermodynamics

Sample with certain constraints

Fixed thermodynamic variables

Macroscopic conditions (constant volume, temperature, number of particles, ...) translate to the microscopic world as **boundary conditions**

Microscopic system is defined by the extensive variables that are constant in the macroscopic world, e.g. (E, V, N), (V, N),...

The probability distribution for the microscopic system and its Hamiltonian are related to the macroscopic free energy function.

Why Statistical Mechanics?

Relate **microscopic** phenomena and **macroscopic** properties:

Given a (macroscopic) thermodynamic state of a material, what are the probabilities of finding the system in the various possible microscopic states?

Given a series of microscopic states, what is the corresponding macroscopic state?

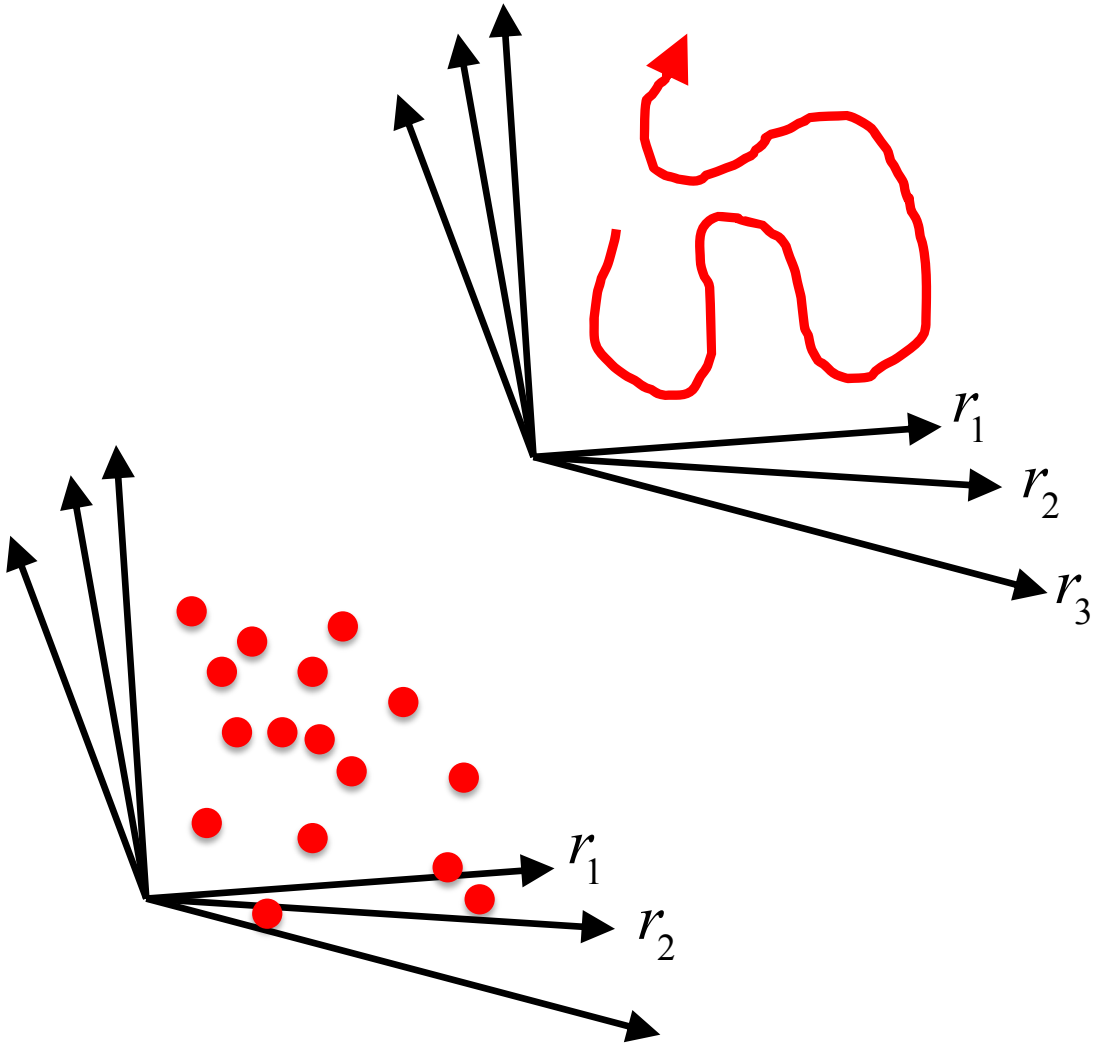
Summary: Important Ensembles and their Boundary Conditions

Microcanonical (NVE)	Canonical (NVT)	Isobaric/isothermal (NPT)
Probability distributions		
$P(\{r_i\},\{p_i\}) = \frac{1}{\Omega(E,V,N)}$	$P(\{r_i\},\{p_i\}) = \frac{e^{\frac{-H(\{r_i\},\{p_i\})}{kT}}}{Z(T,V,N)}$	$P(\{r_i\},\{p_i\},V) \frac{e^{\frac{-H(\{r_i\},\{p_i\})-PV}{kT}}}{Z_P(T,P,N)}$
$\Omega(E,V,N) = \sum_{micro} \delta\left(E - H(\{r_i\},\{p_i\})\right)$	$Z_P(T,V,N) = \sum_{micro} e^{\frac{-E}{kT}}$	$Z_P(T,P,N) = \sum_V \sum_{micro} e^{\frac{-E-PV}{kT}}$
Free energies (atomistic ↔ macroscopic thermodynamics)		
$S = k \log \Omega(E,V,N)$	$F(T,V,N) = -kT \log Z$	$G(T,P,N) = -kT \log Z_p$

Molecular Dynamics vs. Monte Carlo Simulation

Molecular Dynamics:
Solve Equations of Motion

Monte Carlo:
(Importance) Sampling



Canonical Ensembles: Averages and Ergodicity

Consider a quantity A that depends on the atomic positions and momenta:

$$A\left(\{r_i\},\{p_i\}\right)$$

In *equilibrium* the average values of A is:

$$\langle A\rangle = \sum_{\text{microstates}} AP_{\text{micro}} = \frac{\sum_{\text{microstates}} A\left(\{r_i\},\{p_i\}\right) e^{-\beta H\left(\{r_i\},\{p_i\}\right)}}{\sum_{\text{microstates}} e^{-\beta H\left(\{r_i\},\{p_i\}\right)}}$$

Monte Carlo

Ensemble average

When you measure the quantity A in an *experiment*:

$$\langle A\rangle = \frac{1}{\tau}\int_0^\tau A\left(\{r_i(t)\},\{p_i(t)\}\right)$$

Molecular Dynamics

Time series average

Under equilibrium conditions temporal and ensemble averages are equal !

So...This is the Problem We Want to Solve

Our goal is the evaluation of integrals such as:

$$\langle A \rangle = \frac{\int dr^N dp^N A(r^N, p^N) \exp\left[-E(r^N, p^N)/kT\right]}{\int dr^N dp^N \exp\left[-E(r^N, p^N)/kT\right]}$$

Standard Monte Carlo Integration

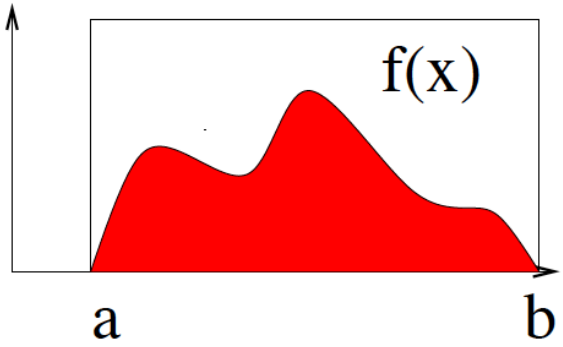
- Consider 1-dimensional definite integral:

$$I = \int_a^b dx f(x)$$

This can be approximated by pulling N random numbers x_i, $i = 1 \ldots N$, from a distribution which is constant in the interval $a \leq x \leq b$:

$$I \approx \frac{b-a}{N} \sum_{i=1}^{N} f(x_i).$$

The result becomes exact when $N \to \infty$.



- If the number of random numbers N is large enough, the error in the approximation is $\propto 1/\sqrt{N}$, due to *central limit theorem*.
- On the other hand, if we divide the a, b-interval into N steps and use some regular integration routine, the error will be proportional to $1/N^p$, where $p = 1$ even with the most naive midpoint integral rule. Thus, Monte Carlo is not competitive against regular quadratures in 1 dimensional integrals (except in some pathological cases).

Standard Monte Carlo Integration

- What now happens in higher dimensional integrals? Let us consider d dimensions:

$$I = \int_V d^d x f(x)$$

For simplicity, let V be a d-dim hypercube, with $0 \leq x_\mu \leq 1$. Now the Monte Carlo integration proceeds as follows:

- Generate N random vectors x_i from flat distribution ($0 \leq (x_i)_\mu \leq 1$).
- As $N \to \infty$,

$$\frac{V}{N} \sum_{i=1}^{N} f(x_i) \to I \,.$$

- Error: $\propto 1/\sqrt{N}$ independent of d! (Central limit)

Appoximating the Integral with a distribution leads to a constant error of $1/\sqrt{N}$, even in d dimensions.

Standard Monte Carlo Integration

- "Normal" numerical integration methods (see e.g. in Numerical Recipes): Divide each axis in n evenly spaced intervals
  - Total # of points $N \equiv n^d$
  - Error:
    - $\propto 1/n$ (Midpoint rule)
    - $\propto 1/n^2$ (Trapezoidal rule)
    - $\propto 1/n^4$ (Simpson)

If d is small, Monte Carlo integration has much larger errors than standard methods

Standard Monte Carlo Integration

- When is MC as good as Simpson? Let's follow the error

$$1/n^4 = 1/N^{4/d} = 1/\sqrt{N} \qquad \rightarrow \qquad d = 8.$$

  In practice, MC integration becomes better when $d \gtrsim 6$–8.

- In lattice simulations $d = 10^6 - 10^8$!

- In practice, N becomes just too large very quickly for the standard methods: already at $d = 10$, if $n = 10$ (pretty small), $N = 10^{10}$. This implies simply too many evaluations of the function!

Monte Carlo: Simple or Important?

The modern form of simple/important sampling originated with Ulam and Segré in Los Alamos and the ENIAC computer (but really goes back to Fermi)

Before that, „sampling“ was used as a method for integration of functions (Comte de Buffon (1777)

Suggestion for Simple Sampling of Materials:

Pick M states randomly from the ensemble and calculate the average property as: $\langle A \rangle = \Sigma_{v=1}^{M} p_\nu A_\nu$

$$P_\nu = \frac{\exp(-\beta H_\nu)}{\Sigma_{\nu=1}^{M} \exp(-\beta H_\nu)}$$

Simple Sampling does not work here, as one picks mainly states with low weight in the true partition function, i.e. states with high energy.

Picking States with a Biased Probability: Importance Sampling

Can we pick states from the ensemble with a probability proportional to $\exp(-\beta E)$ rather than picking random and later weighing them by a probability ?

$$\langle A \rangle = \Sigma_{\nu=1}^{M} \frac{\exp(-\beta H_\nu)}{\Sigma_{\nu=1}^{M} \exp(-\beta H_\nu)} A_\nu \quad \Rightarrow \quad \langle A \rangle = \Sigma_{\upsilon=1}^{M} A_\nu$$

Random Sample

Probability-Weighted Sample

How to construct probability-weighted samples ?

Metropolis Algorithm

„walks“ through phase space (Markov Chain of States), visiting each state with proper probability (in the infinite limit).

1. Random starting state
2. Pick trial state j from i with some rate $W^0_{i\text{-}j}$
3. Accept j with some probability $P_{i\text{-}j}$

Conditions for Generating Proper Probability Distribution

Equal a-priori probabilities: $W^0{}_{i\to j} = W^0{}_{j\to i}$

Detailed Balance: $P_i W_{i\to j} = P_j W_{j\to i}$

When $W_{i\to j}$ and $P_{i\to j}$ satisfy the above criteria, the Metropolis Algorithm will produce an equilibrium distribution

Proof:

Consider an ensemble of systems. To have stable (equilibrium) proportion of number of systems in each state, you need:

$$\Sigma_j \, P_i P_{i\to j} = \Sigma_j \, P_j P_{j\to i}$$

A typical Metropolis Algorithm (but not at all the only one!)

$P_{i \to j} = 1$ when $E_j < E_i$ Downhill moves are always accepted

$P_{i \to j} = \exp(-\beta(E_j - E_i))$ when $E_j > E_i$ Uphill moves are accepted with some thermal-driven probability

Put it all together: A Monte Carlo Algorithm

1. Start with some configuration
2. Choose perturbation of the system
3. Compute energy for that perturbation

4. If $\Delta E < 0$ –> accept perturbation
   If $\Delta E \geq 0$ –> accept perturbation, accept perturbation with probability $\exp\left[\frac{-\Delta E}{k\mathrm{T}}\right]$
5. Choose next perturbation

Property will be average over these states

It is Your Move !

"Dynamics" in Monte Carlo is NOT real, hence you can pick any „perturbations" that satistfy the criterion of detailed balance and a priori probabilities.

For example: Mixing of A and B atoms on a lattice

You could pick nearest neighbor A-B interchange („like" diffusion)

→ Glauber Dynamics

You could pick nearest neighbor A-B interchange („like" diffusion)

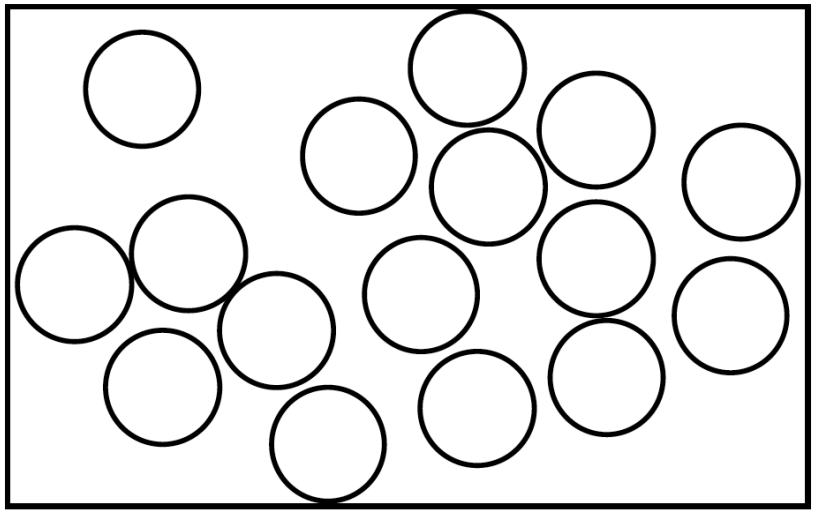
→ Kawasaki Dynamics exchanging the identity is **much** faster !

The really cool thing: One can do Monte Carlo on any Hamiltonian

Example for Off-Lattice Monte Carlo

For example: A liquid

Which perturbation to pick?



Anything that is consistent with the degrees of freedom of the System

1. randomly pick an atom
2. Displace by some random amount a between two limits
3. Comupte $\Delta E = E_{new} - E_{old}$
4. If $\Delta E < 0$ just accept the perturbation
   If $\Delta E > 0$ accept perturbation with probability $P \sim \exp(-\frac{\Delta E}{k_B T})$

Improving the Metropolis Algorithm

- ✓ Achieving equilibrium of a system using the Metropolis algorithm can be greatly accelerated by introducing additional 'MC moves' into the system.

- ✓ The great advantage of the MC method is, that *any* MC trial moves that are thermodynamically permissible, are allowed – they *don't have to be physically realistic*, they only need to obey the acceptance rules which generate a Boltzmann-like distribution of microstates. However, one always has to make sure, that the moves are *still ergodic*.

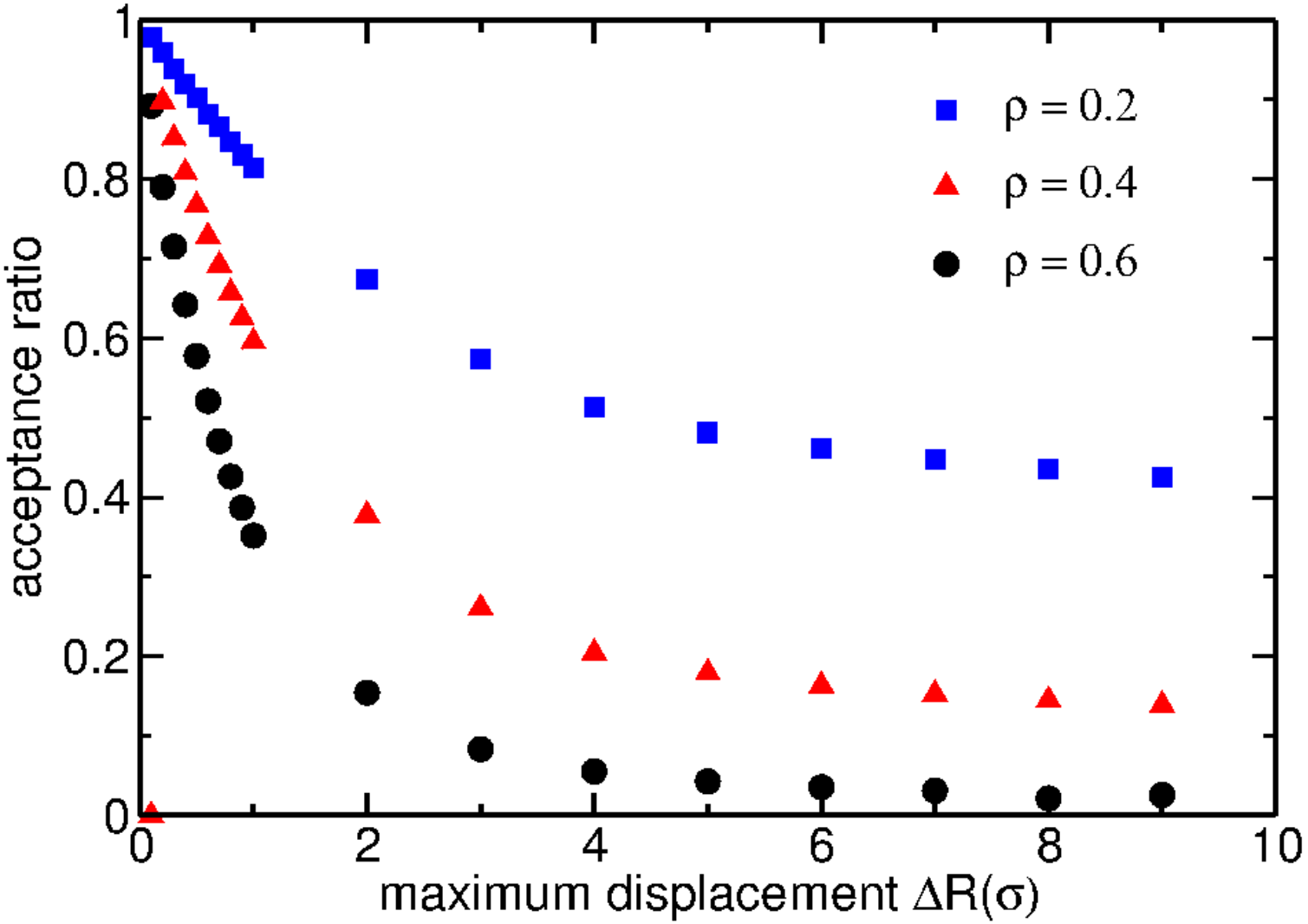
For example:

- ✓ In a monoatomic fluid one can simply construct trial moves by randomly displacing particles.

- ✓ For flexible macromolecules (polymers), one must also consider internal degrees of freedom (stiffness, bond length, bond angles, eigenvolume, etc.) by using *orientational* MC trial moves.

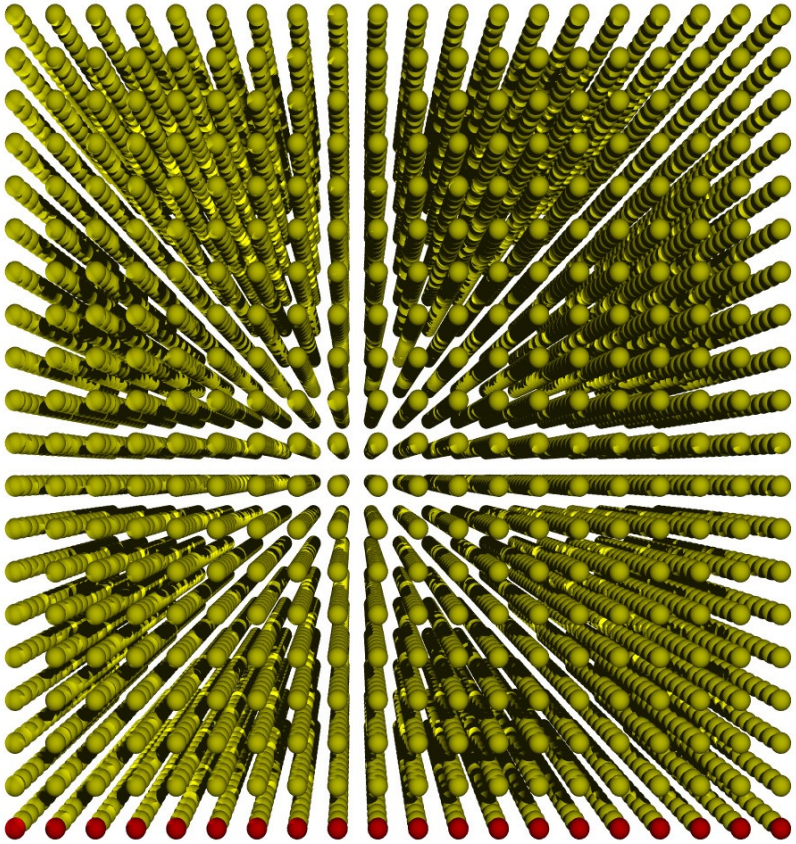
Example: Acceptance Rate of a MC-NVT Ensemble of Hard Disks

2D, N= 2000

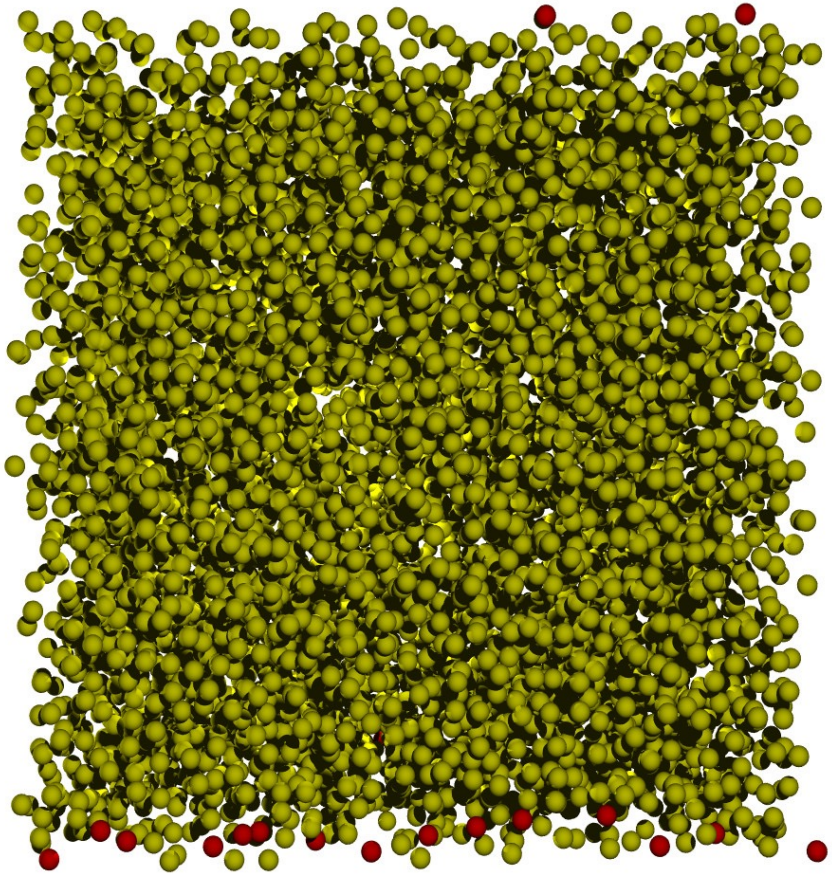


Example: MC-NVT Ensemble of Hard Spheres (3D)

N = 400



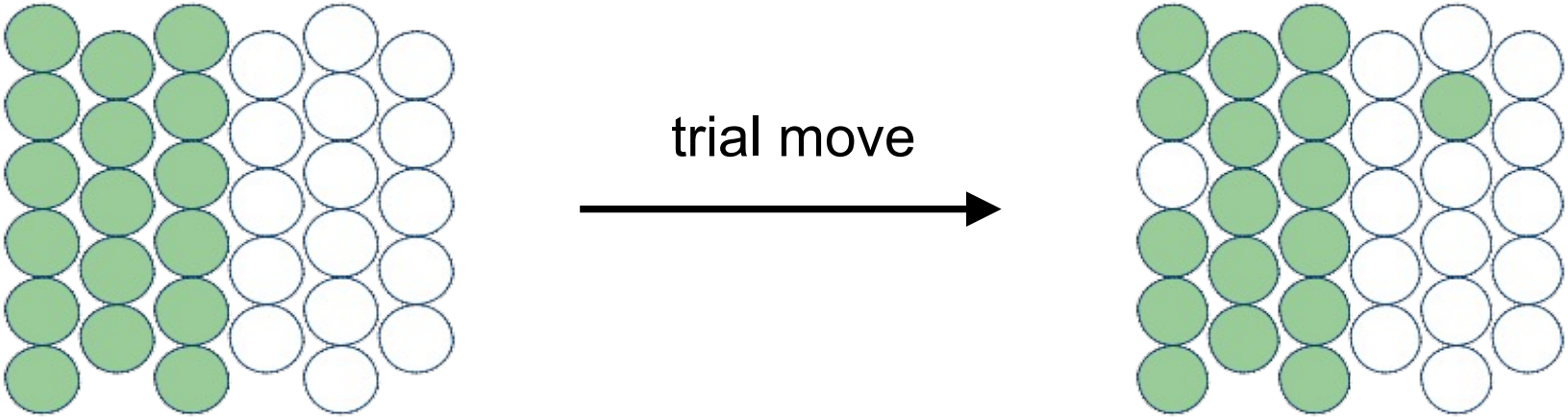
Initial cubic setup



After 10^4 MC steps

Improving the Metropolis Algorithm
Accelerating Equilibration in a Binary Mixture

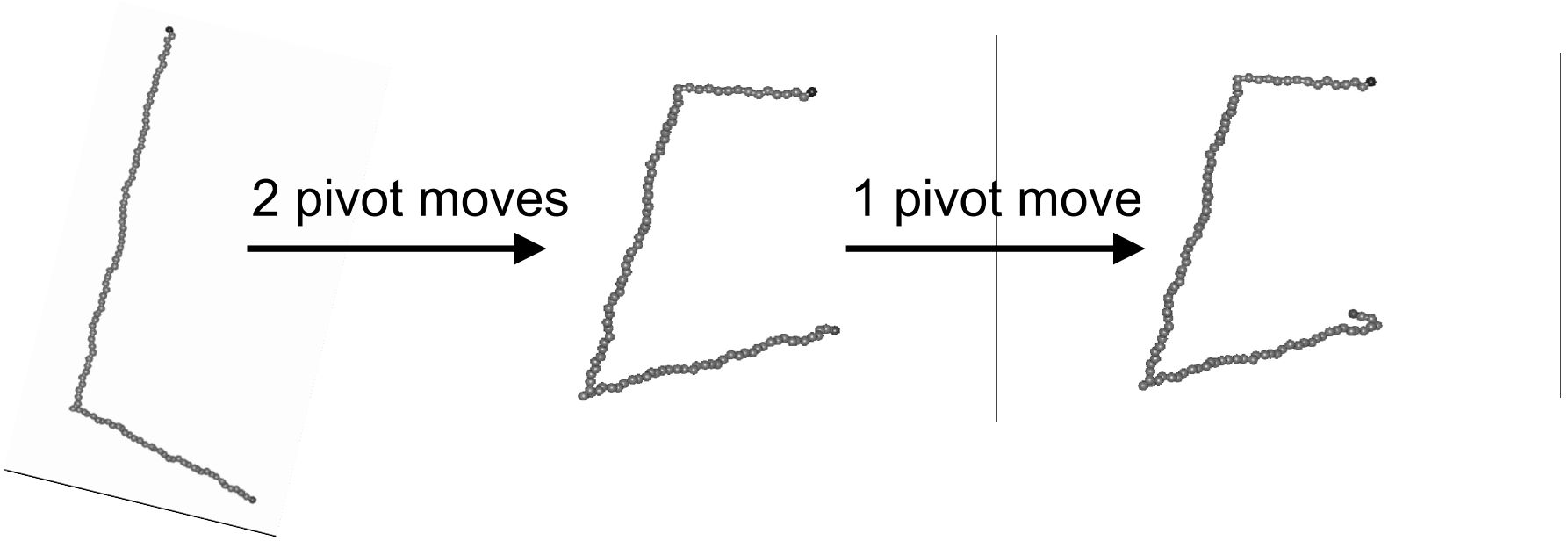
- ✓ An exchange of particle identity as a trial move, even though they might be separated by a large distance and other particles can greatly accelerate the simulation of mixing of species in a dense system.



- ✓ The price one pays for unphysical moves: Loss of information about the dynamics
- ✓ In practice: When using a reasonably physical set of trial moves, MC can also yield information about the equilibrium *dynamics* of a system. Such MC methods are known as *kinetic Monte Carlo (KMC)* techniques.

Improving the Metropolis Algorithm
Accelerating Equilibration in a Binary Mixture

- ✓ Pivot moves which move part of a macromolecule about a randomly chosen monomer of the molecule can accelerate the large-scale equilibration of a polymer.

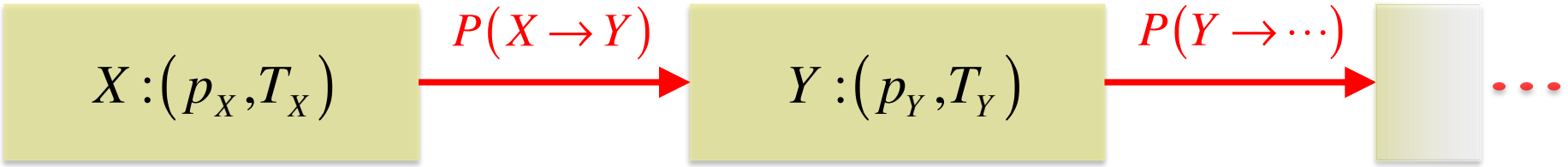


The algorithm is explained in: *"A molecular dynamics study on universal properties of polymer chains in different solvent qualities. Part I. A review of linear chain properties", M. O. Steinhauser, J. Chem. Phys. 122, 1-13 2005*

MC as a Markov Process

- ✓ So far we haven't said anything about *how* we have to choose each state so it appears with its correct Boltzmann probability. One way to answer to this question is to consider MC as a Markov process.

- ✓ A Markov process is a mechanism by which an initial state X is transformed into a new state Y in a stochastic fashion using a set of transition probabilities $P(X \to Y)$ which satisfy certain conditions.



MC as a Markov Process

- ✓ Transition probabilities of a true Markov process should satisfy the following conditions:
  - They should vary with time,
  - They should neither depend on the properties of the initial or final state, nor on any other state through which the system passes,
  - For a given initial state, the sum of transition probabilities over all final states must be equal to one.

- ✓ In a MC simulation we repeatedly apply a Markov process to generate a Markov chain of states.

- ✓ The Markov process is chosen such that it produces a succession of states which appear with their Boltzmann probabilities. This is known as equilibrium of the system.

MC as a Markov Process

- ✓ In order to reach equilibrium, the Markov process has to be ergodic, i.e. it should be possible to reach any state of the system from any other state.

- ✓ The Markov process should satisfy the principle of detailed balance, i. e. the rates at which the system makes transitions into and out of any microstate should be the same:

$$p_X P(X \to Y) = p_Y P(Y \to X)$$

- ✓ The condition of detailed balance imposes a time-reversal symmetry on the system and provides a sufficient (but not necessary) condition for the ratio of the transition probabilities to produce a Boltzmann equilibrium distribution of the occupancy of microstates.

MC as a Markov Process

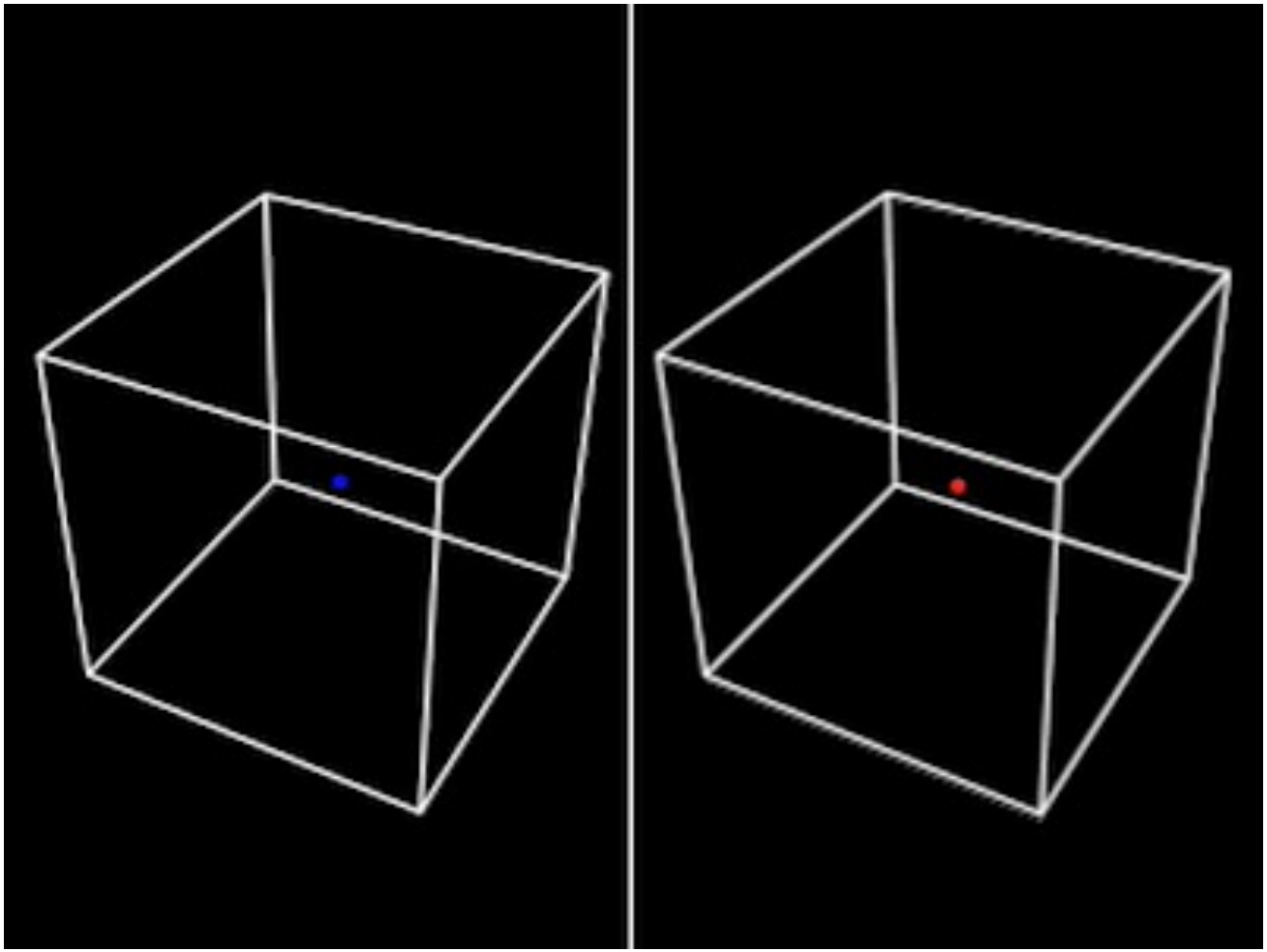
- For the equilibrium distribution to be Boltzmann-like, the ration of the transition probabilities should be

$$\frac{p_X}{p_Y} = \frac{P(Y \to X)}{P(X \to Y)} = \exp\left[-\beta\left(E_Y - E_X\right)\right]$$

- In practice, this is done by tuning the actual transition probabilities such that the **acceptance ratio**, i.e. the probability of a randomly generated transition is as large as possible.

- Only Constraints: 1. The above equation
  2. The sum over all transition probabilities is equal to one.

Last Example: MC Simulation of Proteins
Simple Random Walk (Brownian Motion)



The Metropolis Algorithm

- ✓ The Metropolis algorithm is a general MC scheme for simulating in the **NVT canonical ensemble**, and it works reasonably efficiently in a wide range of situations.
- ✓ A notable exception, where MC does NOT work well, is close to phase transitions where large fluctuations occur.
- ✓ The Metropolis algorithm requires a large number of **good quality random numbers** for the decision of whether a 'MC trial move' which raises the total energy, should be accepted or not.

The Metropolis Algorithm

✓ The Metropolis algorithm implements a Markov Chains and it is a repetition of three steps:

Algorithm The Metropolis algorithm

▷ Step 1

1. Start with a system in an arbitrarily chosen state X and calculate the energy E_X of the system.

▷ Step 2

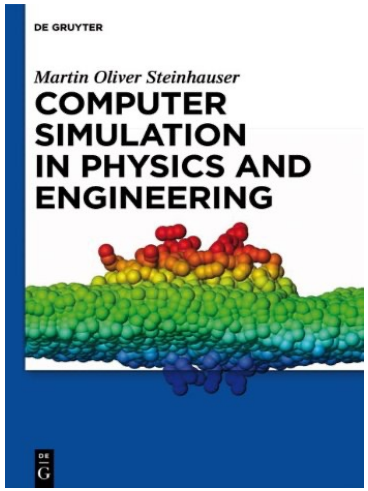
2. Generate a new state Y by a small *ergodic* perturbation to X and calculate the energy E_Y of the system.

▷ Step 3

3.

if $(E_Y - E_X) \leq 0$ **then**
    accept the new state Y,
else
    if $(E_Y - E_X) > 0$ **then**
        accept the new state with probability $exp[-\beta(E_Y - E_X)]$;
     return to Step 2 and repeat until equilibrium is reached.
    end if
end if

Books you might want to check out
Scientific Computing on Different Length- and Time Scales



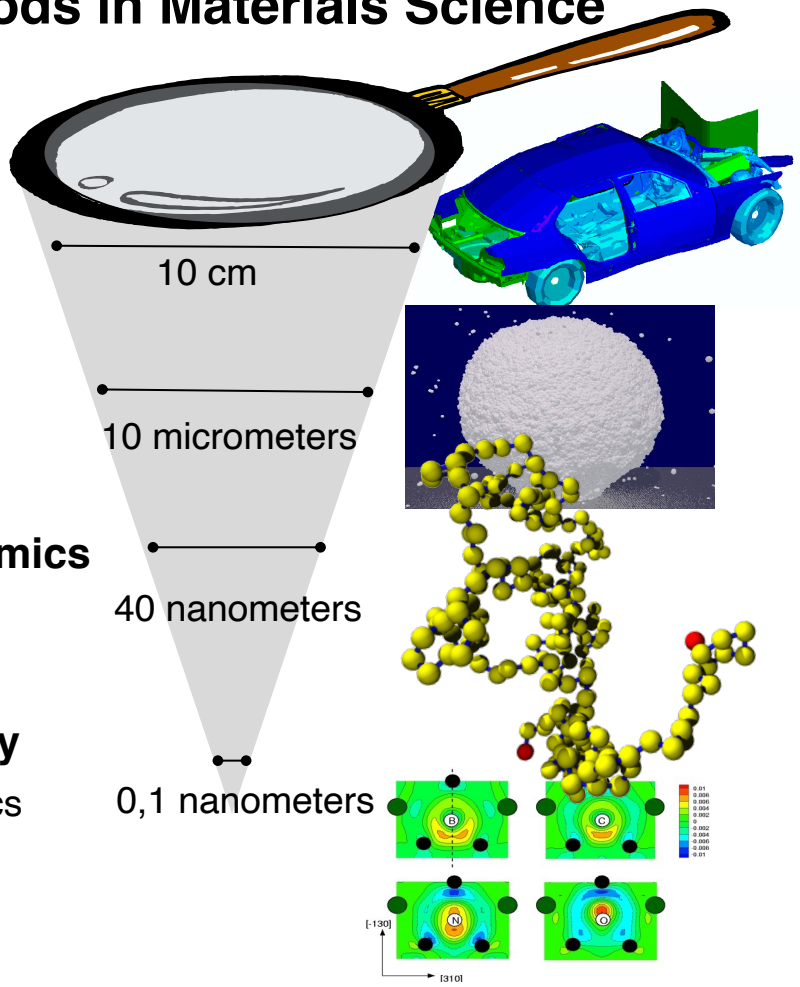
Computational Methods in Materials Science generating Big Data

Finite Elements
Macroscale:
Continuum Mechanics

Meso Particle Dynamics
Mesoscale: Particle Dynamics

Classical Molecular Dynamics
Microscale: Atomistics

Density Functional Theory
Nanoscale: Quantum Mechanics



My University Research Page: https://www.frankfurt-university.de/steinhauser

Contact Me: martin.steinhauser@fb2.fra-uas.de

Research Gate: https://www.researchgate.net/profile/Martin-Steinhauser