



the  
**abdus salam**  
international centre for theoretical physics

ICTP 40th Anniversary

SMR 1595 - 1

---

Joint DEMOCRITOS - ICTP School on  
CONTINUUM QUANTUM MONTE CARLO METHODS  
12 - 23 January 2004

---

**INTRODUCTION TO MONTE CARLO**  
**"Monte Carlo and Random Walks"**

**David M. CEPERLEY**  
Beckman Institute for Advanced Studies & Technology  
University of Illinois at Urbana-Champaign - N.C.S.A.  
II-61801 Urbana, U.S.A.

---

*These are preliminary lecture notes, intended only for distribution to participants.*



# Monte Carlo and Random Walks

Today we discuss basic Monte Carlo techniques.

- What is Monte Carlo?
  - Any computational method which uses random numbers as an essential part of the algorithm
  - Equivalent to performing integrals by sampling the integrand
  - Often a Markov chain, in particular Metropolis MC
- References
  - *Allen&Tildesley "Computer Simulation of Liquids"*
  - *Frenkel&Smit*
  - *Thijssen, "Computational Physics"*
  - *Kalos&Whitlock, "Monte Carlo Methods"*
  - *"Numerical Recipes"*

# MC is advantageous for high dimensional integrals

Consider an integral in the unit hypercube:

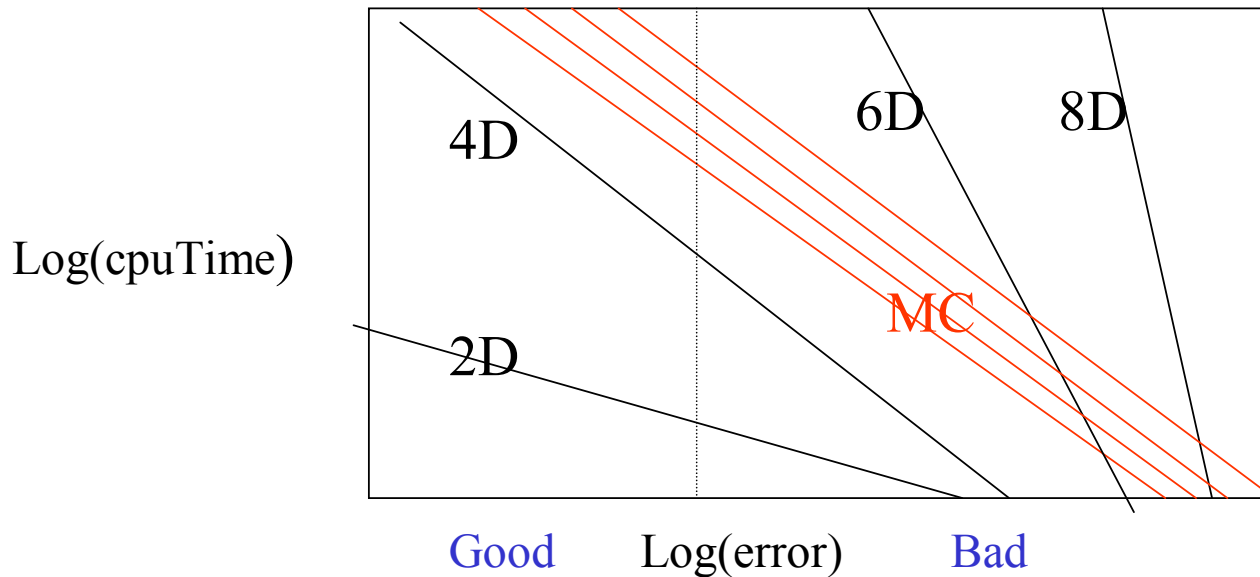
$$I = \int_0^1 dx_1 \dots dx_D f(x_1, \dots, x_D)$$

## By conventional deterministic methods:

- Lay out a grid with L points in each direction with  $h=1/L$
- Number of points is  $N=L^D \propto$  CPU time.

## HOW DOES ERROR GO WITH CPU TIME and DIMENSIONALITY?

- Error in trapezoidal rule goes as  $\varepsilon=f''(x) h^2$ .
- The CPU time  $\propto \varepsilon^{-D/2}$ .
- But by sampling we find  $\varepsilon^{-2}$ . To get another decimal place takes 100 times longer!



Other reasons to do Monte Carlo:

- Conceptually and practically simple.
- Comes with built in error bars.

*Many methods of integration have been tried, and will be tried in this world of sin and woe. No one pretends that Monte Carlo is perfect or all-wise. Indeed, it has been said that Monte Carlo is the worst method except all those other methods that have been tried from time to time. Churchill 1947*

# Probability Distributions

- $P(x)dx =$  probability of observing a value in  $(x, x+dx)$  is a **probability distribution function** (p.d.f.)

$$\int dx P(x) = 1 \quad P(x) \geq 0$$

- $x$  can be either a continuous or discrete variable.
- **Cumulative distribution:**

Probability of  $x < y$ .

Useful for sampling

$$c(y) = \int_{-\infty}^y dx P(x) \quad 0 \leq c(y) \leq 1.$$

- Average or expectation

- Moments:

– Zeroth moment  $I_0 = 1$

– Mean  $\langle x \rangle = I_1$

– Variance  $\langle (x - \langle x \rangle)^2 \rangle = I_2 - (I_1)^2$

$$\langle g(x) \rangle = \bar{g} = \int dx P(x) g(x)$$

$$I_n = \langle x^n \rangle = \int dx p(x) x^n$$

# Mappings of random variables

Let  $P_x(x)dx$  be a probability distribution

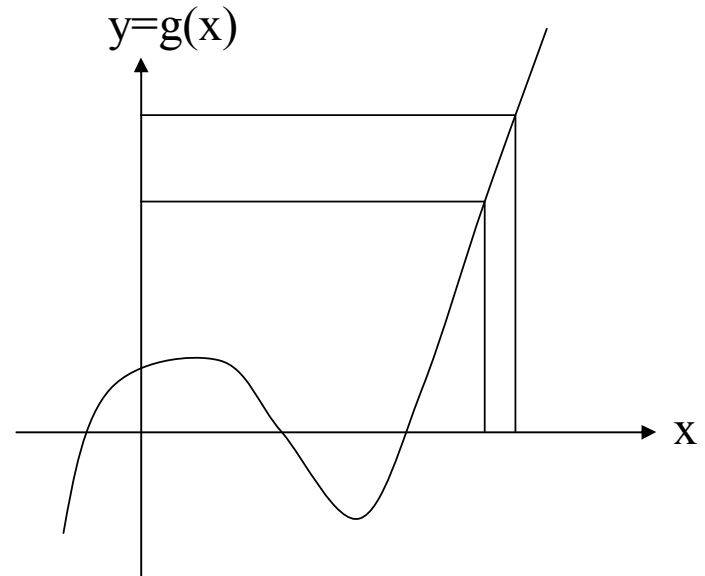
Let  $y=g(x)$  be a new variable

What is the pdf of  $y$ ?

$$P_y(y)dy = P_x(x)dx$$

$$P_y(y) = \frac{P_x(x)}{\left| \frac{dg}{dx} \right|_x}$$

What happens when  $g$  is not monotonic?



# Central Limit Theorem (Gauss)

Sample  $N$  values from  $p(x)dx$ . ( $X_1, X_2, X_3, \dots, X_N$ )

Estimate mean from

What is the pdf of mean?

$$y = \frac{1}{N} \sum_{i=1}^N x_i$$

*Solve by fourier transforms.*

If you add together two random variables, you multiply together their **characteristic functions**:

$$c_x(k) = \langle e^{ikx} \rangle = \int dx p(x) e^{ikx}$$

$$c_{x+y}(k) = c_x(k) c_y(k)$$

Then

$$c_{x_1+\dots+x_N}(k) = c_x(k)^N$$

Taylor expand

$$c_y(k) = c_x(k/N)^N$$

cumulants

$$\ln(c_x(k)) = \sum_{n=1}^{\infty} \kappa_n \frac{(ik)^n}{n!}$$





cumulants  $\kappa_n$

mean= $\kappa_1$

variance= $\kappa_2$

skewness = $\kappa_3$

kurtosis= $\kappa_4$

What happens to the reduced moments?

$$\boxed{\kappa_n} = \kappa_n N^{1-n}$$

Hence the n=1 moment remains invariant.

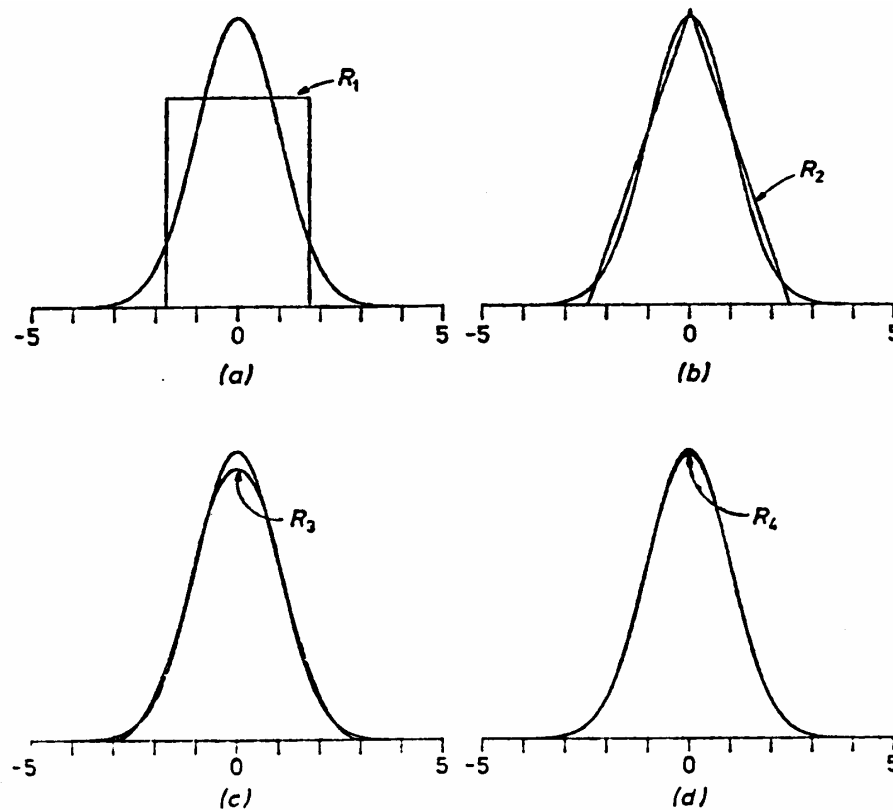
The rest get reduced by higher and higher powers of N.

$$\lim_{N \rightarrow \infty} c_y(k) = e^{ik\kappa_1 - k^2\kappa_2/2N - ik^3\kappa_3/6N^2 \dots}$$

$$p(y) = \left( N / 2\pi\kappa_2 \right)^{1/2} e^{-\frac{N(y-\kappa_1)^2}{2\kappa_2}}$$

Given enough averaging almost anything becomes a Gaussian distribution.

# Approach to normality



**Figure 1.** Distributions of sums of uniform random numbers, each compared with the normal distribution. (a)  $R_1$ , the uniform distribution. (b)  $R_2$ , the sum of two uniformly distributed numbers. (c)  $R_3$ , the sum of three uniformly distributed numbers. (d)  $R_{12}$ , the sum of twelve uniformly distributed numbers.

# Conditions on Central Limit Theorem

$$I_n = \langle x^n \rangle = \int dx p(x) x^n$$

- We need the first three moments to exist.
  - If  $I_0$  is not defined  $\Rightarrow$  not a pdf
  - If  $I_1$  does not exist  $\Rightarrow$  not mathematically well-posed.
  - If  $I_2$  does not exist  $\Rightarrow$  infinite variance. **Important to know if variance is finite for Monte Carlo.**
- Divergence could happen because of tails of distribution

$$I_2 = \int_{-\infty}^{\infty} dx p(x) x^2 \quad \lim_{x \rightarrow \pm\infty} x^3 p(x) \rightarrow 0$$

- Or because of singular points, e.g. at  $x=0$

$$\lim_{x \rightarrow \pm 0} x^3 p(x) \rightarrow \text{finite}$$

# Random Number Generation

*Also read "Numerical Recipes".*

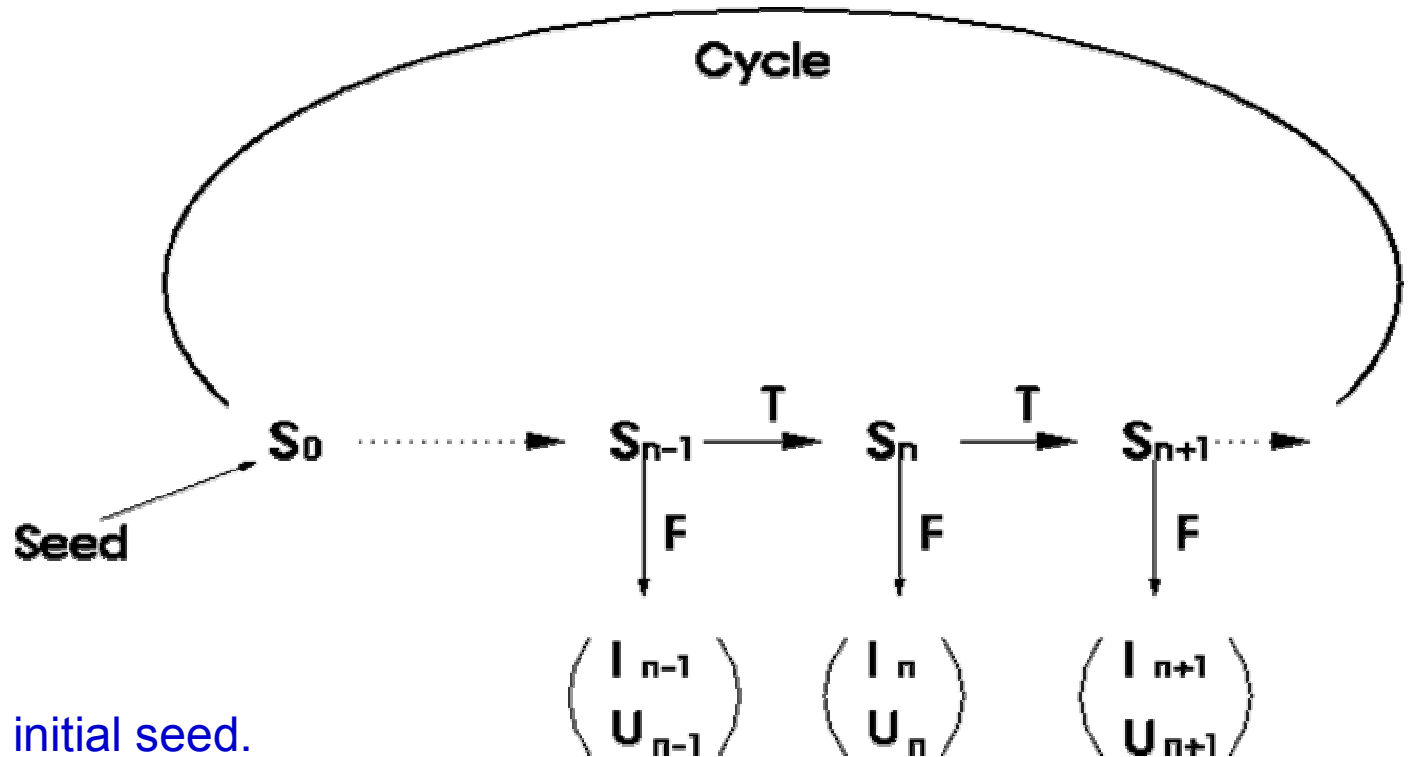
What is a random number?

- A single number is not random. Only an infinite sequence can be described as random.
- Random means the absence of order (a 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 integrals and that you will test for in the homework.

# Random numbers on a computer

- Truly random--the result of a physical process such as timing clocks, circuit noise, bad memory
  - Too slow (we need  $10^{10}$ /sec)
  - Too expensive
  - Low quality
  - Not reproducible
- Pseudo-random. prng (pseudo means fake)
  - Deterministic sequence
  - But if you don't know the algorithm they appear to be random
- Quasi-random (quasi means almost random)
  - "half way" between random and a uniform grid

# Pseudo Random Sequence



S: State and initial seed.

T: Iteration process,

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

# Cycle length

- If internal state has  $M$  bits, total state space is  $2^M$  values.
- If mapping is 1-1, then it will divide up space into a finite of cycles.
- Best case is a single cycle of length  $2^M$ .
- Entire period of the RNG is exhausted in:

- rand (1 processor)  $\sim 100$  second
  - drand48 (1 processor)  $\sim 1$  year
  - drand48 (100 processor)  $\sim 3$  days
  - SPRNG LFG ( $10^5$  procs)  $\sim 10^{375}$  years
- Assuming  $10^7$  RN/sec per processor

- It is easy to achieve very long cycle length but 32 or 24 bit generators are no longer adequate!

# Common PRNG Generators

<ul style="list-style-type: none"> <li>• Multiplicative Lagged Fibonacci</li> <li>• Modified Lagged Fibonacci</li> </ul>	$z_n = z_{n-k} * z_{n-l}$ $z_n = z_{n-k} + z_{n-l}$ <p>(modulo <math>2^m</math>)</p>	<p>vary initialization</p>
<ul style="list-style-type: none"> <li>• 48 bit LCG</li> <li>• 64 bit LCG</li> <li>• Prime Modulus LCG</li> </ul>	$z_n = a * z_{n-1} + p$ <p>(modulo <math>m</math>)</p>	<p>vary <math>p</math></p>
		<p>vary <math>a</math></p>
<ul style="list-style-type: none"> <li>• Combined Multiple Recursive</li> </ul>	$z_n = a_{n-1} * z_{n-1} + \dots + a_{n-k} * z_{n-k} + \text{LCG}$	<p>vary LCG</p>

Recurrence

Parallelization



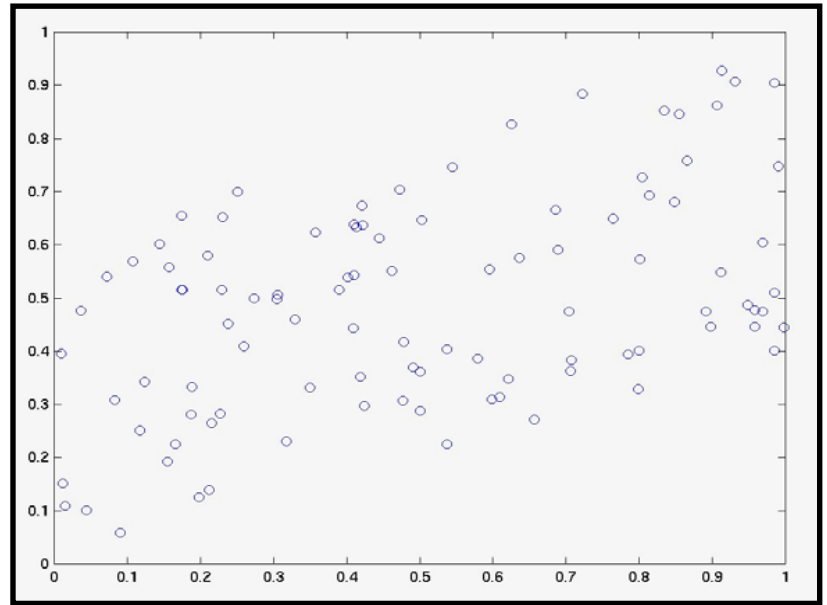
# Uniformity

- Output consists of taking  $N$  bits from state and making an integer in  $(0, 2^N - 1)$  “ $I_k$ ”
- One gets a uniform real “ $U_k$ ” by multiplying by  $2^{-N}$ .
- We will discuss how to get other distributions.
- If there is a single cycle, then integers must be uniform in the range  $(0, 2^N - 1)$
- Uniformity of numbers taken 1 at a time is usually easy to guarantee.
- But what we need is higher dimensional uniformity

# Sequential RNG Problems

- Correlations  $\Rightarrow$  non-uniformity in higher dimensions

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



This is the important property to guarantee:

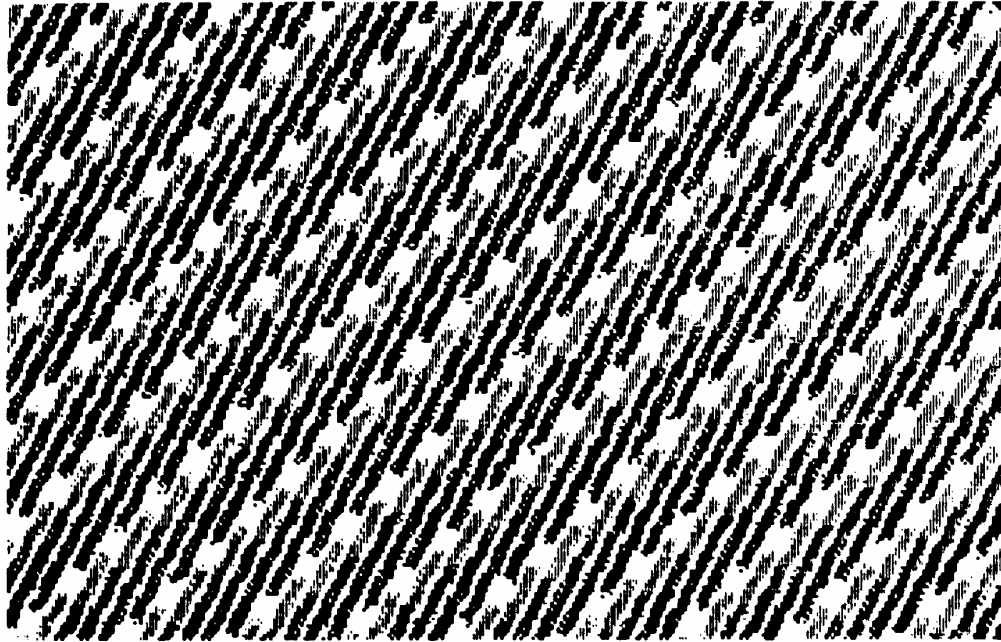
$$\langle f(x_{i+1})g(x_{i+2}) \rangle = \langle f(x_i) \rangle \langle g(x_i) \rangle$$

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

# 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. The value of the third coordinate is represented by the intensity of the point.

# SPRNG

- A library of *many* well tested *excellent parallel* PRNGs
- Callable from FORTRAN.C, C++, and JAVA
- Ported to popular parallel and serial platforms

## SPRNG Functions

- int ***init\_sprng***(int streamnum, int nstreams, int seed, int param)
- double ***sprng***(int \*stream)
- int ***isprng***(int \*stream)
- int ***print\_sprng***(int \*stream)
- int ***make\_sprng\_seed***()
- int ***pack\_sprng***(int \*stream, char \*\*buffer)
- int ***unpack\_sprng***(char \*buffer)
- int ***free\_sprng***(int \*stream)
- int ***spawn\_sprng***(int \*stream, int nspawned, int \*\*\*newstreams)

# Chi-squared test of randomness

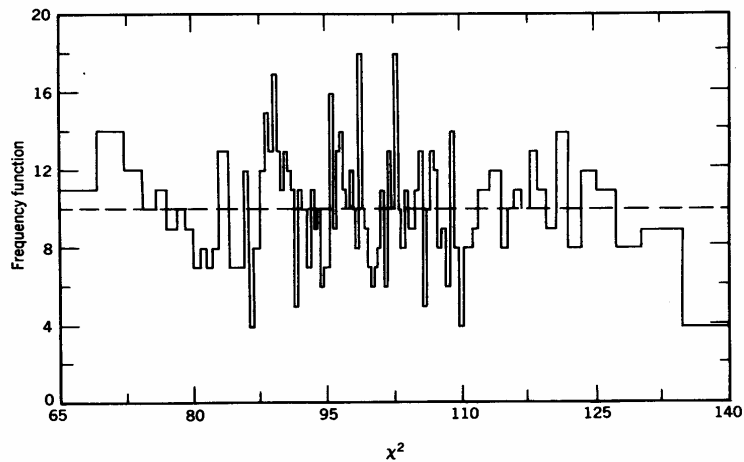
- HW exercise: do test of several generators
- Divide up "cube" into "N" bins.
- Sample many "P" triplets.
- Number/bin should be  $n = P/N \pm (P/N)^{1/2}$

N=100

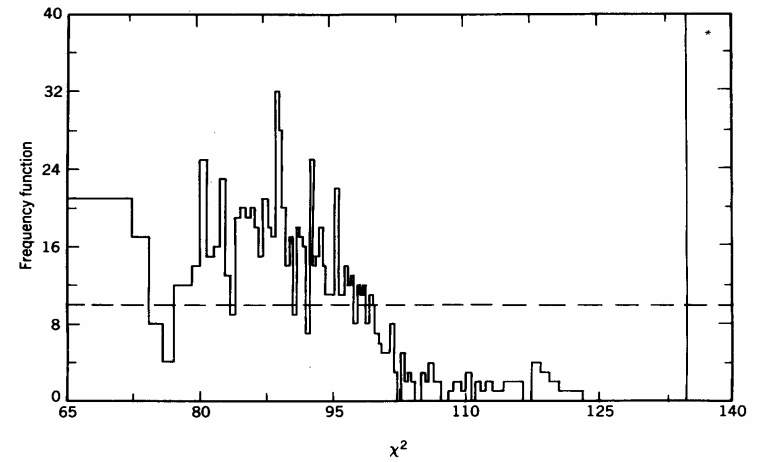
P=1000

GOOD

BAD



**Figure A.1.** The frequency function of observed values of  $\chi^2$  after 1000 samples. The bins are equally probable intervals where the expected number of  $\chi^2$  in each bin is 10. The pseudorandom number generator was Eq. (A.13).



**Figure A.2.** The frequency function of observed values of  $\chi^2$  after 1000 samples using prn generator Eq. (A.14). The expected number of  $\chi^2$  in each bin is 10. The value of the bin indicated by (\*) is offscale.

- Chi-squared statistic is 
$$\chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i}$$
- Where  $n_i$  is the expected number in bin  $i$ .
- The probability of a given chi-squared is  $Q(\chi^2 | N-1)$   
( $N-1$ ) because of “sum rule”.

For more details see “Numerical Recipes”

## Recommendation: For careful work use several generators!

- For most real-life MC simulations, passing existing statistical tests is necessary but not sufficient.
- Random number generators are still a black art.
- Rerun with different generators, since algorithm may be sensitive to different RNG correlations.
- Computational effort is not wasted, since results can be combined to lower error bars.
- In SPRNG, relinking is sufficient to change RNGs

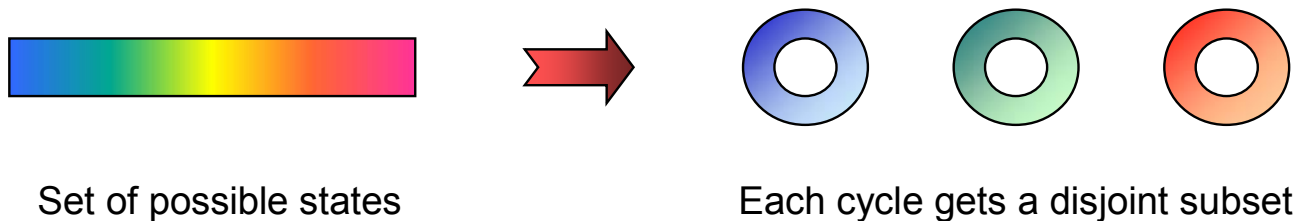
# Parallel Simulations

- Parallel Monte Carlo is easy? **Or is it?**
- Two methods for easy parallel MC:
  - **Cloning**: same serial run, different random numbers.
  - **Scanning**: different physical parameters (density,...).
- Any parallel method (MPI, ..) can be used.
- Problems:
  - Big systems require excessive wall clock time.
  - Excessive amounts of output data generated.
  - Random number correlation?



# Parallelization of RNGs

- **Cycle Division: Leapfrog or Partition.** Correlation problems
- **Cycle Parameterization:** If PRNG has several cycles, assign different **cycles** to each stream



- **Iteration Parameterization:** Assign a different iteration function to each stream

# Examples of Parallel MC codes and Problems

- Embarrassingly parallel simulations. Each processor has its own simulation. **Scanning** and **cloning**. *Unlikely to lead to problems unless they stay in phase.*
- Lots of integrals in parallel (e.g. thousands of Feynman diagrams each to an accuracy of  $10^{-6}$ ). *Problem if cycle length is exhausted.*
- Particle splitting with new particles forking off new processes. Need lots of generators. *Problem if generators are correlated initially.*
- Space-time partitioning. Give each local region a processor and a generator. *Problem if generators are correlated.*

# Sampling Distributions

*"Numerical Recipes" on random numbers*

How to generate non-uniform probability distributions.

These are used in importance sampling to reduce the variance of a Monte Carlo evaluation or to simulate various physical processes.

We start by assuming that there is software to generate udrn's in the range  $(0,1)$ .

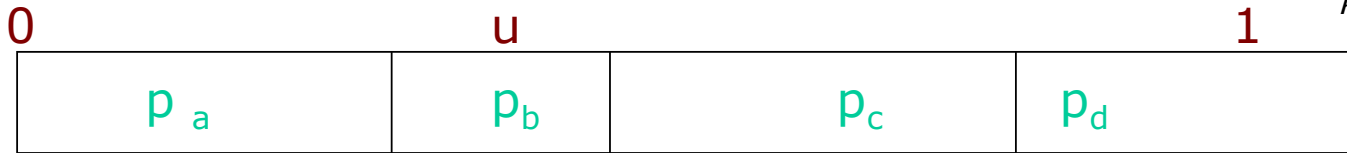
How do we sample an arbitrary  $p(x)dx$  ?

There are lots of tricks.

# Discrete Distributions

Any discrete distribution  $p_k$  can be sampled by constructing the cumulant.

$$c_k = \sum_{i=1}^k p_i$$



- Sample  $0 < u < 1$ .
  - Find which region it is in. i.e. find  $k$ :  $c_{k-1} < u < c_k$
  - Return label "k".
- 
- The search operation can be done by bisection in  $\log_2(N)$  steps.
  - For simple distributions, it might be even easier.
  - There is a faster  $O(1)$  method, using a precomputed table.

# Continuous Distributions

Generalize to a continuous function.

This is the mapping method.

- Construct the cumulant:  $c(y) = \int_{-\infty}^y dx p(x)$
- Sample  $u$  in  $(0,1)$
- Find  $x=c^{-1}(u)$  (Can always perform with a table lookup.)

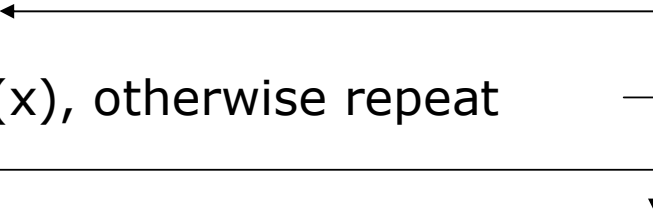
Some analytic examples:

- $P(x)=a e^{-ax} \quad 0 < x$  then  $x = -\ln(u)/a$

- $P(x)=(a+1)x^a \quad 0 < x < 1$  then  $x = u^{1/(a+1)}$

Problem occurs if inverse mapping is difficult.

# Rejection technique

- Sample  $x$  from  $q(x)dx$
  - Accept  $x$  with probability  $c(x)$ , otherwise repeat
- 

What is distribution of accepted  $x$ 's?

$$p(x)dx = c(x)q(x)dx / \text{normalization}$$

Hence choose

$$c(x) = a p(x) / q(x)$$

Where  $a$  is set so that  $c(x) \leq 1 \Rightarrow a \leq \min(q(x) / p(x))$

$1/a$  is the acceptance probability,  $a$  is the *efficiency*.

If inefficient will use lots of prns/step.

Do not undersample, or else efficiency will be low.

# Composition method

Combine several random numbers

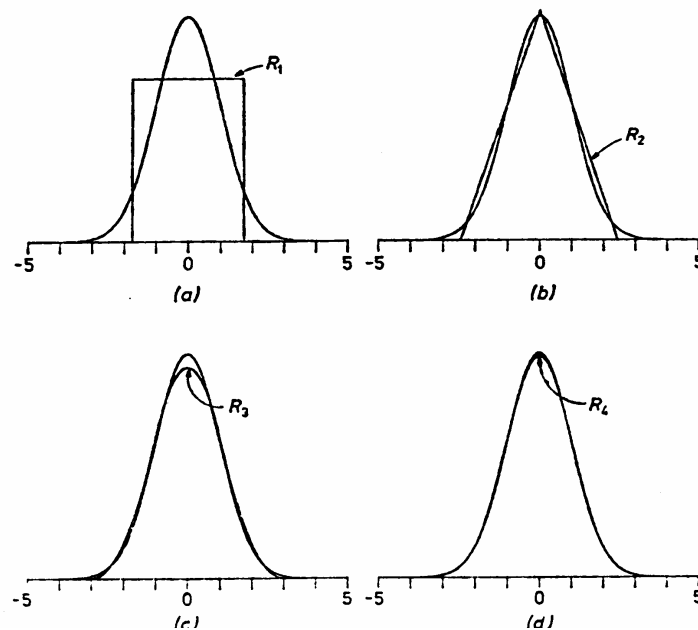
- Add several udrn's.  
Remember characteristic function  
Limit for large  $k$  is a Gaussian

Example: add two integers in  $(1,6)$

- Multiply or divide 2 udrns?

- Take maximum of 'k' udrns.

$$x = \max(u_1, \dots, u_k) \quad \text{Prove that} \quad P(x) = k x^{k-1}$$



# Normal distribution

- Inverse mapping is a little slow, also of infinite range.
- Trick: generate 2 at a time:  $r=(x,y)_2$

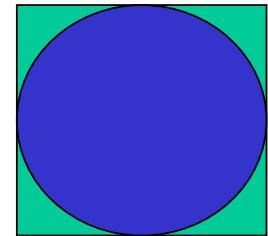
$$p(x, y)dxdy = (2\pi)^{-1} \exp\left(-\frac{r^2}{2}\right) = p(r)rdrd\theta$$

$$p(v)dv = \frac{1}{2}e^{-v/2} \text{ with } v = r^2$$

$$x = \sqrt{-2\ln(u_1)} \cos(2\pi u_2)$$

$$y = \sqrt{-2\ln(u_1)} \sin(2\pi u_2)$$

- Or sample angle using rejection technique:
  - Sample  $(x,y)$  in square
  - Accept if  $x^2+y^2 < 1$
  - Normalize to get the correct  $r$ .





# Code to sample normal distribution

Normal distribution  $\langle x \rangle = x_0$  and  $\langle (x-x_0)^2 \rangle = \sigma^2$

```
1      x=sprng()-0.5
      y=sprng()-0.5
      r2=x*x+y*y
      if (r2>0.25) go to 1
      radius= sigma*sqrt (-2*ln(sprng())/r2)
      xnormal=x0+x*radius
      ynormal=y0+y*radius
```

- No trig functions, 1 log, 1 sqrt, 1 divide
- Mixes up regularity of random numbers
- Efficiency of angle generation is  $4/\pi$ .
- Can get 2 ndrn's each time.

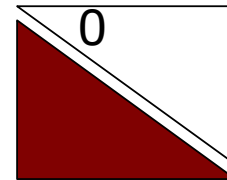
# Multivariate normal distributions

How to sample a correlated Gaussian? (say with  $D$  components)

- Assume we want  $\langle x_i x_j \rangle = T_{ij}$
- Make Choleski decomposition of  $T$ , (take square root).  
(see Numerical Recipes or notes)

$$SS^t = T$$

- We can assume  $S$  is a triangular matrix



- Generate  $D$  normally distributed numbers  $y$ .
- Transform to correlated random distribution

$$X = Sy$$

# Bias vs statistical error

- Bias is a *systematic error* caused by using a quantity with fluctuations in a non-linear expression.
- You will get a result systematically too high or low.
- Suppose  $Z+\delta Z$  is the result of a MC evaluation of  $\langle Z \rangle$ .
- But we quote  $F(Z)$ . *Example:*  $F=-kT \ln(Z)$ .
- What is the statistical error and bias on  $F$ ?
- Expand  $Z$  in power series about  $\langle Z \rangle$

$$F(Z) = F(\langle Z \rangle) + \frac{dF}{dZ} \delta Z + \frac{1}{2} \frac{d^2 F}{dZ^2} \delta Z^2$$

$$\text{bias}(F) = \langle F(Z) \rangle - F(\langle Z \rangle) = \frac{1}{2} \frac{d^2 F}{dZ^2} \langle \delta Z^2 \rangle \quad \text{order } \frac{1}{N}$$

$$\text{error}(F) = \left\langle \left( F(Z) - F(\langle Z \rangle) \right)^2 \right\rangle^{1/2} = \left| \frac{dF}{dZ} \right| \langle \delta Z^2 \rangle^{1/2} \quad \text{order } \frac{1}{N^{1/2}}$$

# Efficiency of MC

statistical error  $\sim \sqrt{\text{variance}/\text{computer time}}$ .

DEFINE:

$$\text{efficiency} = \zeta = \frac{1}{\nu T}$$

$\nu = \text{error}^2$  of mean  
 $T = \text{total computer time}$

One can either:

- write faster code,
- get a faster computer or
- work on reducing the variance/step.

# Importance Sampling

Given the integral

$$I = \int dx f(x)$$

How should we sample  $x$  to maximize the efficiency?

Estimator

Transform the integral to:

$$I = \int dx p(x) \left[ \frac{f(x)}{p(x)} \right] = \left\langle \left[ \frac{f(x)}{p(x)} \right] \right\rangle_p$$

The variance is:

$$v = \left\langle \left[ \frac{f(x)}{p(x)} - I \right]^2 \right\rangle_p = \int dx \frac{f(x)^2}{p(x)} - I^2$$

Optimal sampling:

$$\frac{\delta v}{\delta p(x)} = 0 \text{ with constraints}$$

Parameterize as:

$$p(x) = \frac{q^2(x)}{\int dx q^2(x)}$$

Solution:

$$p^*(x) = \frac{|f(x)|}{\int dx |f(x)|}$$

Estimator:

$$f(x) / p^*(x) = \frac{\text{sign}(f(x))}{\int dx |f(x)|}$$

If  $f(x)$  is entirely positive or negative, estimator is constant.  
**variance principle.**

“zero

We can't sample  $p^*(x)$ , but its form can guide us.

Importance sampling is a general technique: it works in many dimensions.

# Example of important sampling.

$$f(x) = \frac{e^{-x^2}}{1+x^2}$$

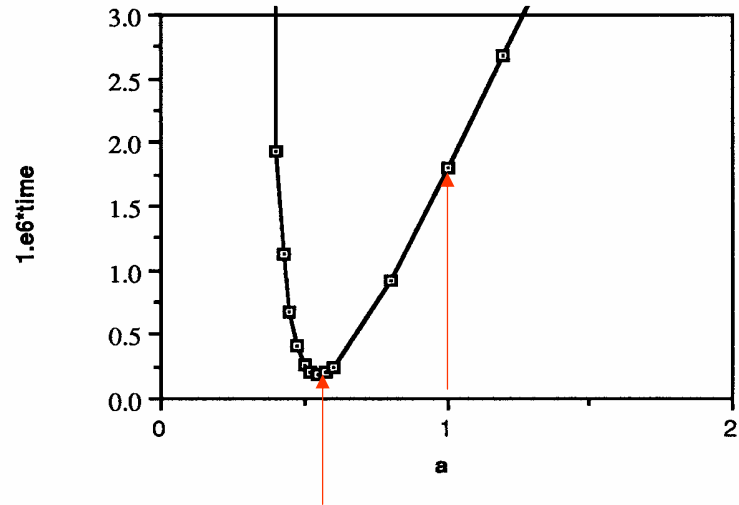
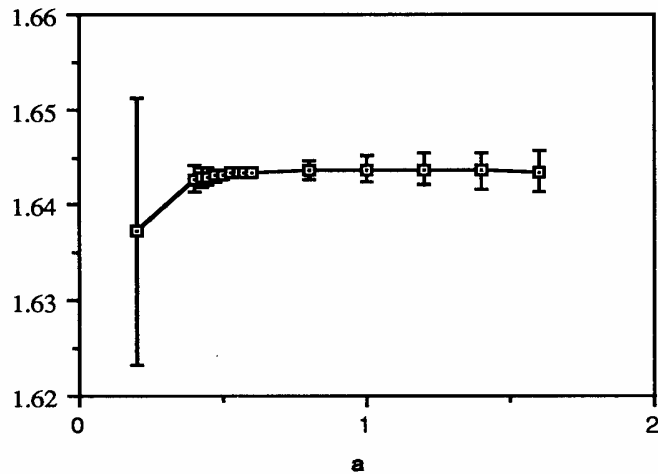
Optimize “a”

$$p(x) = (2\pi a)^{-1/2} e^{-\frac{x^2}{2a}}$$

Mean value is independent of a.

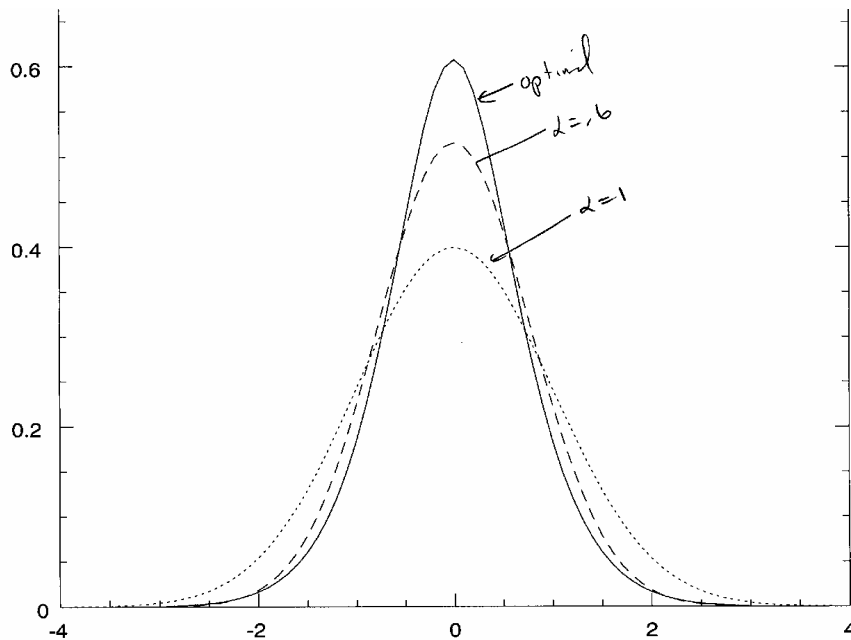
CPU time is not

relative cpu time

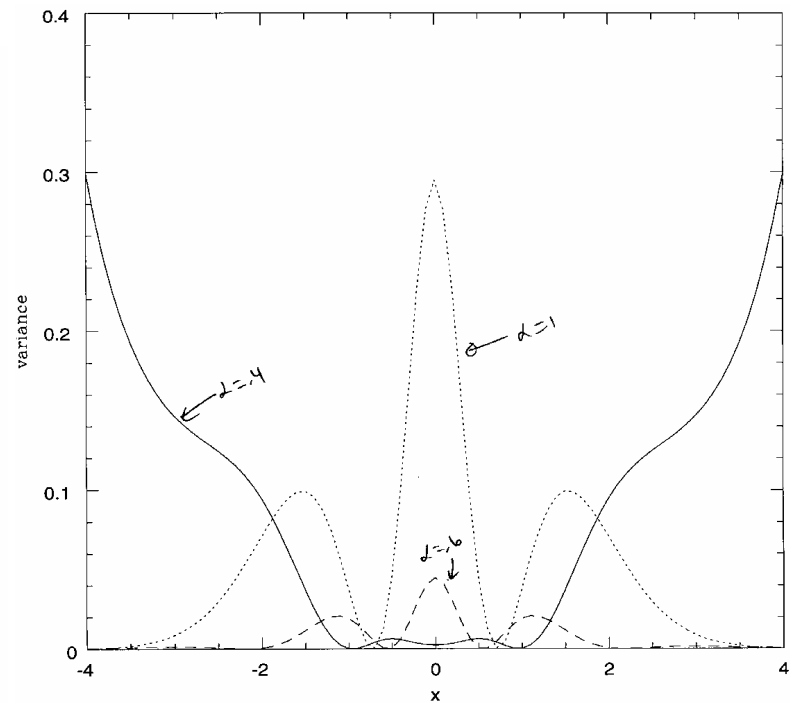


$$v = \int_{-\infty}^{\infty} dx p(x) \left( f(x) / p(x) - I \right)^2 = \int dx c \frac{e^{-x^2(2 - \frac{1}{2a})}}{(1 + x^2)^2} - I^2$$

## Importance sampling functions



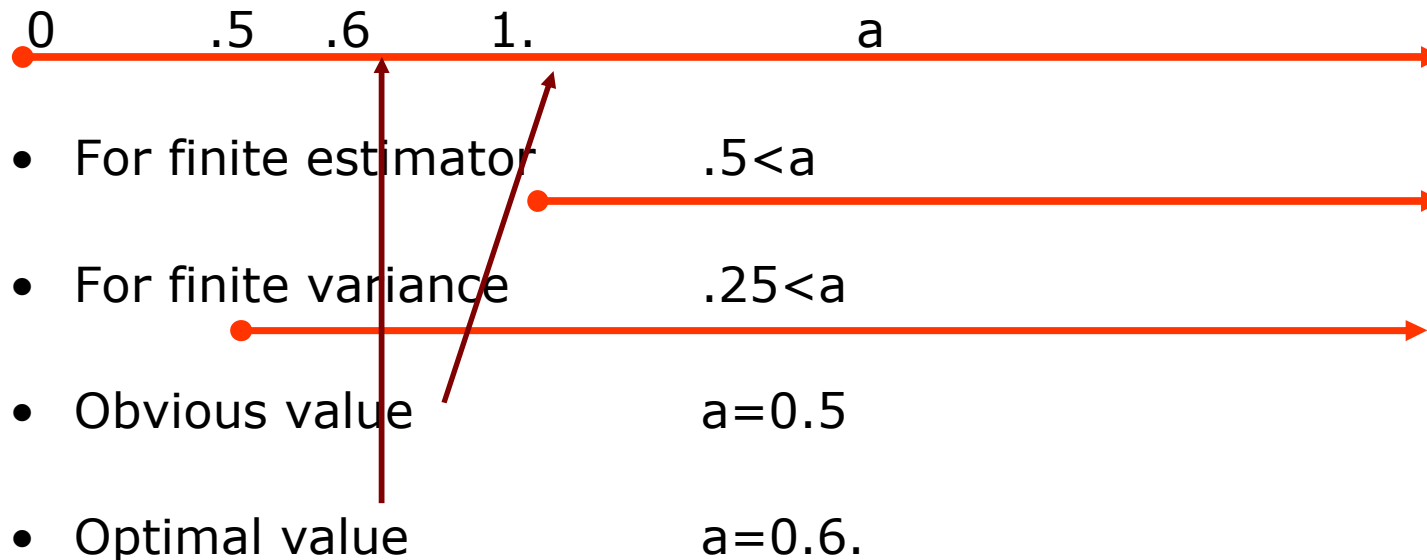
## Variance integrand





# What are allowed values of a?

- Clearly for  $p(x)$  to exist:  $0 < a$



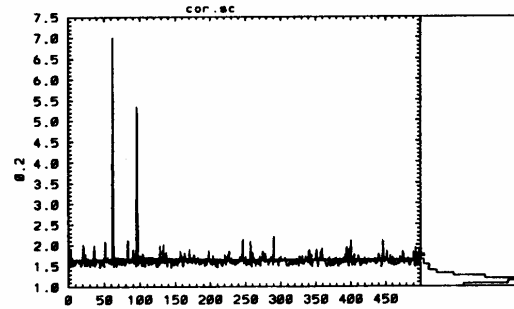
$$v = \int dx \frac{f(x)^2}{p(x)} = \int dx \frac{(2\pi a)^{1/2}}{1+x^2} e^{-x^2 \left(2 - \frac{1}{2a}\right)}$$

What does infinite variance look like?

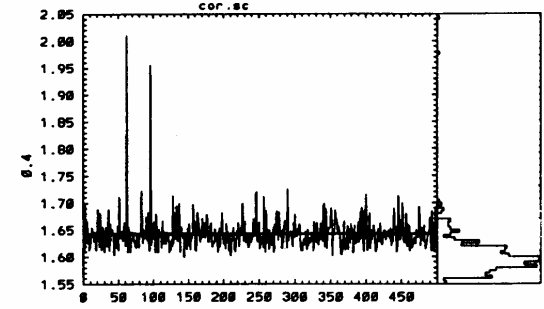
Spikes

Long tails on the distributions

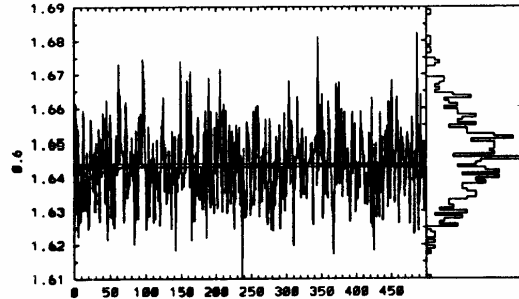
$d = .2$   $.014 = \epsilon$



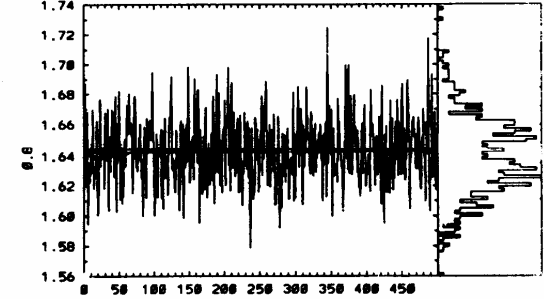
$d = .4$   $.0014 = \epsilon$



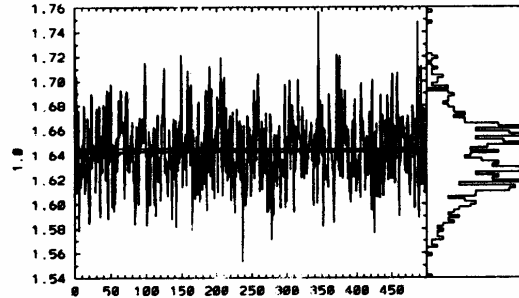
$d = .6$   $.00049$



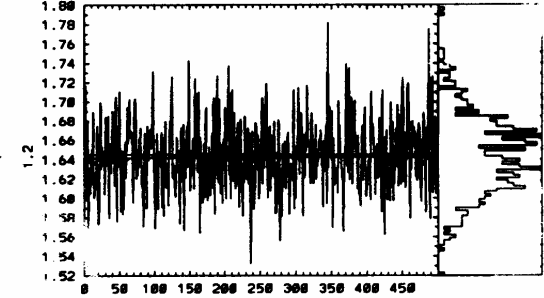
$d = .8$   $.00097$



$d = 1.0$   $.0013$



$d = 1.2$   $.0016$



- Basic idea of importance sampling is to sample more in regions where function is large.
- Find a convenient approximation to  $|f(x)|$ .
- Do not under-sample. That could cause infinite variance.
- Over-sampling results in loss of efficiency but not infinite variance.
- **Always derive analytically conditions for finite variance.**
- To debug: test that estimated value is independent of important sampling.
- *Sign problem*: zero variance is not possible for oscillatory integral. **"Monte Carlo can add but not subtract."**

# Correlated Sampling

Suppose we want a function of 2 integrals:

$$G(F_1, F_2) \text{ where } F_k = \int dx f_k(x)$$

Use the same  $p(x)$  and random numbers to get both integrals.

What is optimal  $p(x)$ ? 
$$p^*(x) \propto \left| f_1(x) \frac{dG}{dF_1} + f_2(x) \frac{dG}{dF_2} \right|$$

It is a weighted average of the distributions for  $F_1$  and  $F_2$ .

Consider  $G = F_1/F_2$  (like Boltzmann distribution)

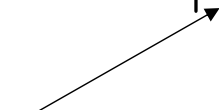
$$p^*(x) \propto \left| f_2(x) \left( \frac{f_1(x)}{f_2(x)} - G \right) \right|$$

# Sampling Boltzmann distribution

- Suppose we want to calculate a whole set of averages:

$$\langle O_k \rangle = \frac{\int dR O_k(R) e^{-V(R)/kT}}{\int dR e^{-V(R)/kT}}$$

- Optimal sampling is:

$$p_k^*(x) \propto \left| \underbrace{e^{-V(R)/kT}}_{\text{constant}} \left( \underbrace{O_k(R) - \langle O_k \rangle}_{\text{variable}} \right) \right|$$


- We need to sample this only. Avoid undersampling.
- The Boltzmann distribution is very highly peaked.

# Random Walks

- It is very difficult to sample directly a general probability distribution.  
If we sample from another distribution, the overlap will be order  $\exp(-aN)$  where  $N$  is the number of variables.
- Markov chains (random walks) allow you sample any distribution based on detailed balance and transition rules.
- These methods were introduced by Metropolis et al in 1953 who applied it to a hard sphere liquid.
- One of the most powerful and most used algorithms.



Markov 1856-1922

# 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

**T**HE 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.

# Markov chain or Random Walk

- Markov chain is a random walk through phase space:

$$s_1 \Rightarrow s_2 \Rightarrow s_3 \Rightarrow s_4 \Rightarrow \dots$$

Here “ $s$ ” is the state of the system

- The transition probability is:  $P(s_n \rightarrow s_{n+1})$  *stochastic matrix*
- In a Markov chain, the distribution of  $s_{n+1}$  depends only on  $s_n$  (by definition). **A drunkard has no memory.**
- Let  $f_n(s)$  be the probability after “ $n$ ” steps. It evolves according to a “master equation.”

$$f_{n+1}(s') = \sum_s f_n(s) P(s \rightarrow s')$$

- The stationary states are eigenfunctions of  $P$ .

$$\sum_s \pi(s) P(s \rightarrow s') = \varepsilon \pi(s')$$



- Because P is positive, the eigenvalues have  $\varepsilon \leq 1$ . An equilibrium state must have  $\varepsilon = 1$ .
- How many equilibrium states are there?
- If it is *ergodic*, then it will converge to a unique stationary distribution (only one eigenfunction=1)
- In contrast to MD, ergodicity can be proven
- Conditions:
  - One can move everywhere in a finite number of steps with non-zero probability. No barriers
  - Non-periodic transition rules. (for example hopping on a bi-partite lattice)
  - Average return time is finite. (no expanding universe) Not a problem in a finite system.
- If ergodic, convergence is geometrical and monotonic.

$$f_n(s) = \pi(s) + \sum_{\lambda} \varepsilon_{\lambda}^n c_{\lambda} \phi_{\lambda}(s)$$

# Metropolis algorithm

Three key concepts:

1. Sample by using an ergodic random walk.
2. Determine equilibrium state by using detailed balance
3. Achieve detailed balance by using rejections.

**Detailed balance:**  $\pi(s) P(s \rightarrow s') = \pi(s') P(s' \rightarrow s)$ .

*Rate balance from  $s$  to  $s'$ .*

Put  $\pi(s)$  into the master equation.

$$\sum_s \pi(s) P(s \rightarrow s') = \sum_s \pi(s') P(s' \rightarrow s) = \pi(s') \sum_s P(s' \rightarrow s) = \pi(s')$$

- Hence  $\pi(s)$  is an eigenfunction.
- If  $P(s \Rightarrow s')$  is ergodic then  $\pi(s)$  is the unique steady state solution.

# Rejection Method

Metropolis achieves detailed balance by *rejecting* moves.

Break up transition probability into sampling and acceptance:

$$P(s \rightarrow s') = T(s \rightarrow s') A(s \rightarrow s')$$

$$T(s \rightarrow s') = \text{sampling probability}$$

$$A(s \rightarrow s') = \text{acceptance probability}$$

The optimal acceptance probability that gives detailed balance is:

$$A(s \rightarrow s') = \min \left[ 1, \frac{T(s' \rightarrow s)\pi(s')}{T(s \rightarrow s')\pi(s)} \right]$$

Note that normalization of  $\pi(s)$  is not needed or used!

# The “Classic” Metropolis method

***Metropolis-Rosenbluth -Teller (1953) method for sampling the Boltzmann distribution is:***

- Move from  $s$  to  $s'$  with probability  $T(s \rightarrow s') = \text{constant}$
- Accept with move with probability:

$$A(s \rightarrow s') = \min [ 1 , \exp ( - (E(s') - E(s)) / k_B T ) ]$$

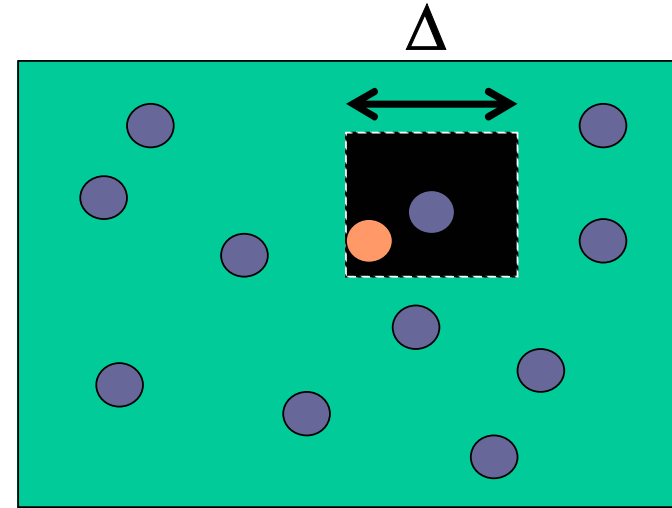
- Repeat many times

- Given ergodicity, the distribution of  $s$  will be the canonical distribution:  $\pi(s) = \exp(-E(s)/k_B T) / Z$
- **Convergence is guaranteed but the rate is not!**

# How to sample

$$S_{\text{new}} = S_{\text{old}} + \Delta \text{ (sprng - 0.5)}$$

Uniform distribution in  
a cube of side " $\Delta$ ".



Note: It is more efficient to move one particle at a time because only the energy of that particle comes in and the movement and acceptance ratio will be larger.

$$A(s \rightarrow s') = \exp(-\beta(V(s') - V(s))) = \exp\left(-\beta \sum_{j \neq i} (v(r_i' - r_j) - v(r_i - r_j))\right)$$

# MONTE CARLO CODE

```
call initstate(s_old)
E_old = action(s_old)
LOOP{
  call sample(s_old,s_new,T_new,1)
  E_new = action(s_new)
  call sample(s_new,s_old,T_old,0)
  A=exp(-E_new+E_old) T_old/T_new
  if(A.gt.sprng()) {
    s_old=s_new
    E_old=E_new
    naccept=naccept+1}
  call averages(s_old)
}
```

Initialize the state

Sample snew  
Trial action

Find prob. of going backward

Acceptance prob.

Accept the move  
Collect statistics

# Overview of MCMC

- Decide how to move from state to state.
- Initialize the state
- Throw away first  $k$  states as being out of equilibrium.
- Then collect statistics but be careful about correlations.

Common errors:

1. If you can move from  $s$  to  $s'$ , the reverse move must also be possible.
2. Accepted and rejected states count the same!

**Exact:** no time step error, no ergodic problems *in principle* but no dynamics either.

Always measure acceptance ratio. Adjust to *roughly* 0.5 by varying the “step size”

A 20% acceptance ratio actually achieves better diffusion.

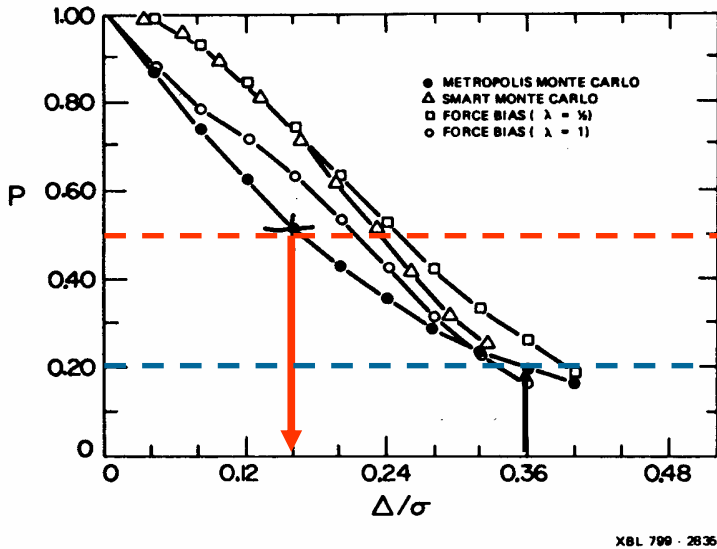


Fig. 1. Average acceptance probability.

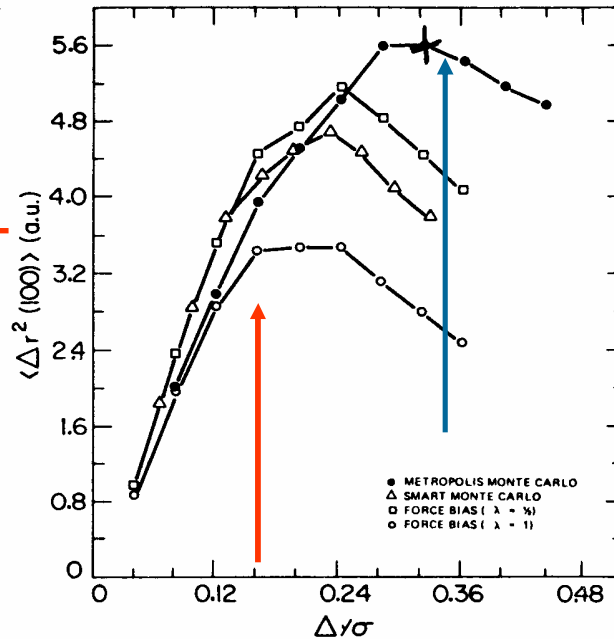


Fig. 3.  $\langle (\vec{r}(i) - \vec{r}(i + 100))^2 \rangle$

$\vec{r}(i) = 3n$  vector of argon positions at step  $i$ .



Variance of energy (local quantity) is not as sensitive to step size. MC is a robust method!

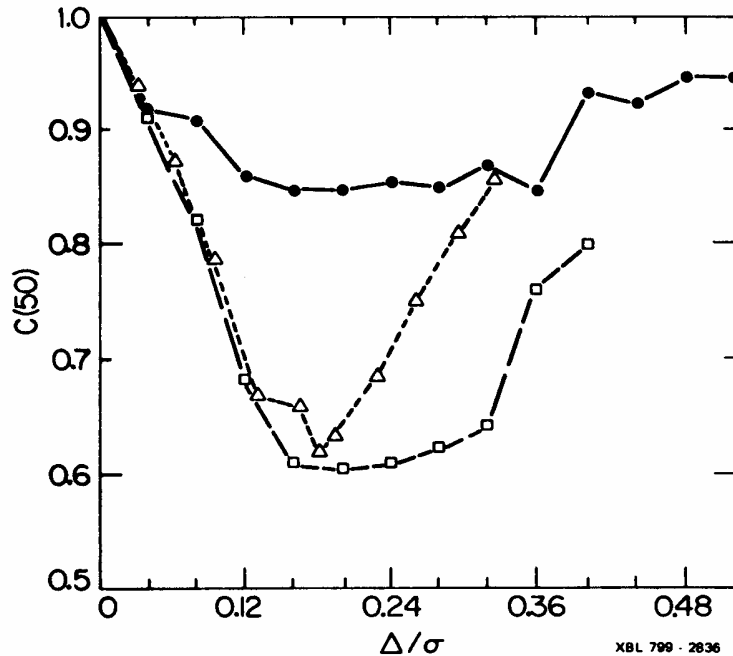


Fig. 2.  $\frac{\langle \Delta V(i) \Delta V(i + 50) \rangle}{\langle \Delta V(i)^2 \rangle}$

where  $i$  = step number and  $\Delta V$  is the deviation of potential energy from the mean.

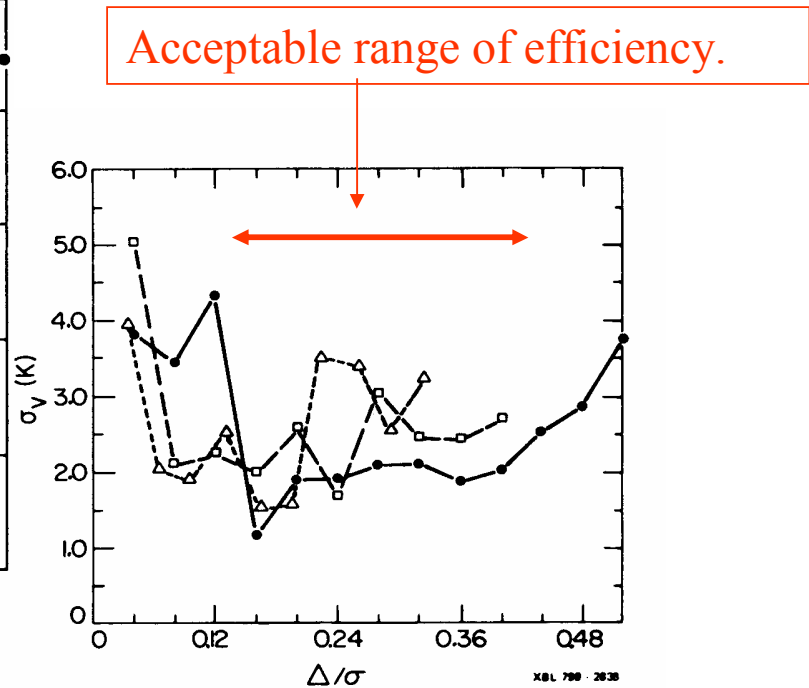


Fig. 4. The variance of the total potential energy for calculations with the same number of steps.

# Comparison between MC and MD

## Which is better?

- MD can compute dynamics. MC has a kinetics but dynamics is not necessarily physical. MC dynamics is useful for studying long-term diffusive process.
- MC is simpler: no forces, no time step errors and a direct simulation of the canonical ensemble.
- In MD you can only work on how to make the CPUtime/physical time faster by inventing better transition rules. Ergodicity is less of a problem. MD is sometimes very effective in highly constrained systems.
- MC is more general, it can handle discrete degrees of freedom (e. g. spin models, quantum systems), grand canonical ensemble...

So you need both! The best is to have both in the same code so you can use MC to warm up the dynamics.

# Optimizing the moves

- Any transition rule is allowed as long as you can go anywhere in phase space with a finite number of steps. (ergodic)
- Try to find a  $T(s \Rightarrow s') \approx \pi(s')$ . If you can the acceptance ratio will be 1.
- We can use the forces to push the walk in the right direction. Taylor expand about the current point.

$$V(\mathbf{r}) = V(\mathbf{r}_0) - \mathbf{F}(\mathbf{r}_0)(\mathbf{r} - \mathbf{r}_0)$$

- Set  $T(s \Rightarrow s') \approx \exp[-\beta(V(\mathbf{r}_0) - \mathbf{F}(\mathbf{r}_0)(\mathbf{r} - \mathbf{r}_0))]$
- Leads to Force-Bias Monte Carlo
- Related to Brownian motion (Smoluchowski Eq.)

# Brownian Dynamics

Consider a big molecule in a solvent. In the high viscosity limit the “master equation” is:

$$\frac{\partial \rho(R, t)}{\partial t} = D \nabla^2 \rho(R, t) - \beta D \nabla [F(R) \rho(R, t)]$$

$$R(t + \tau) = R(t) + \tau \beta D F(R(t)) + \eta(t)$$

$$\langle \eta(t) \rangle = 0 \quad \langle \eta(t)^2 \rangle = 2Dh$$

Enforce detailed balance by rejections! (hybrid method)

$$T(R \rightarrow R') = c \exp\left(-\frac{(R' - R - \beta D \tau F(R))^2}{2D\tau}\right)$$

Also the equation for Diffusion Quantum Monte Carlo!

# Problems with estimating errors

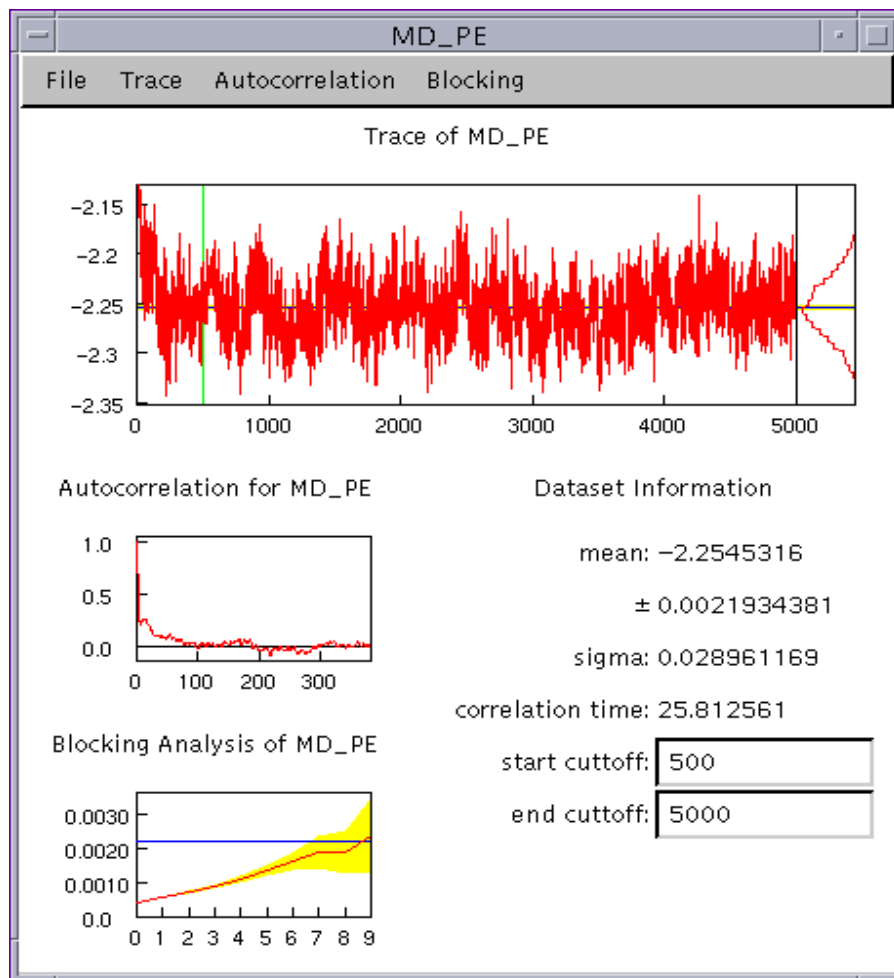
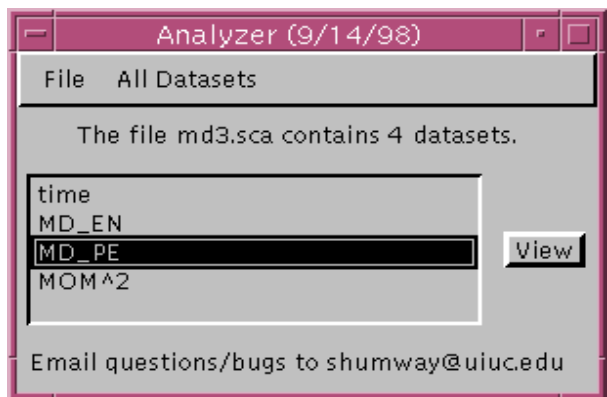
- Any good simulation quotes systematic and statistical errors for anything important.
- The error and mean are simultaneously determined from the same data. HOW?
- Central limit theorem: the distribution of an average approaches a normal distribution (if the variance is finite). One standard deviation means  $\sim 2/3$  of the time the correct answer is within  $\sigma$  of the sample average.
- Problem in simulations is that data is correlated in time. It takes a “correlation” time to be “ergodic”
- We must throw away the initial transient
- Get rid of autocorrelation.
- We need  $\geq 20$  independent data points to estimate errors.

# Estimating Errors

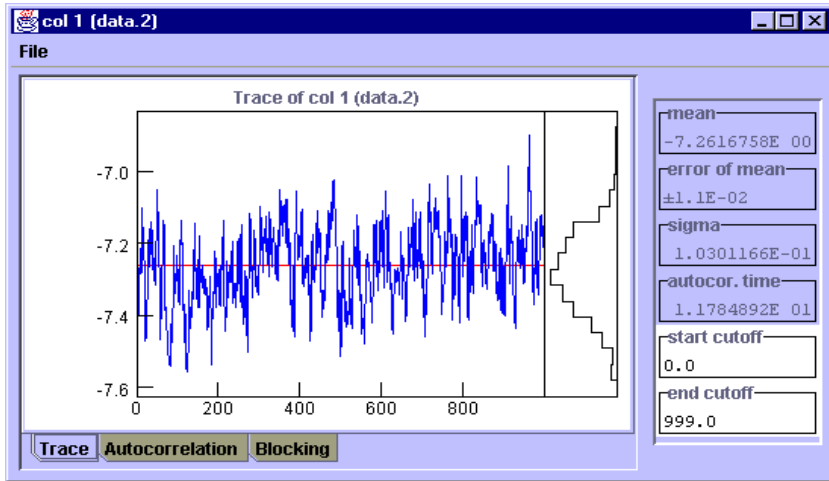
- **Trace of  $A(t)$ :**
- **Equilibration time.**
- **Histogram** of values of  $A$  (  $P(A)$  ).
- **Mean** of  $A$  ( $a$ ).
- **Variance** of  $A$  (  $v$  ).
- **estimate of the mean:**  $\Sigma A(t)/N$
- **estimate of the variance**
- **Autocorrelation** of  $A$  ( $C(t)$ ).
- **Correlation time** ( $\kappa$  ).
- The (estimated) **error** of the (estimated) **mean** ( $\sigma$  ).
- **Efficiency** [=  $1/(\text{CPU time} * \text{error}^2)$ ]

# DataSpork

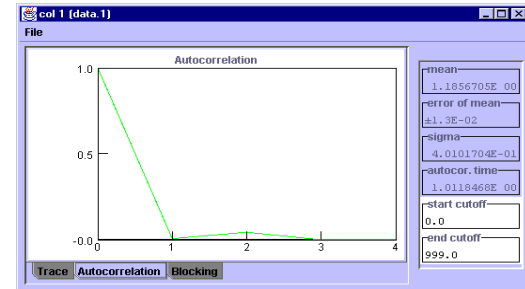
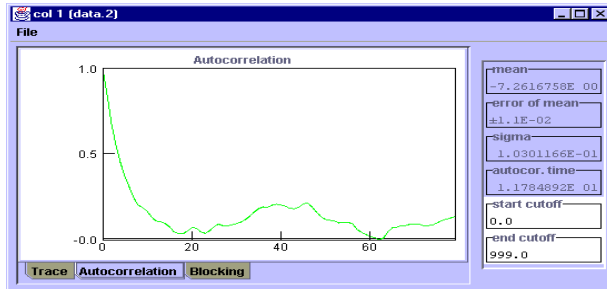
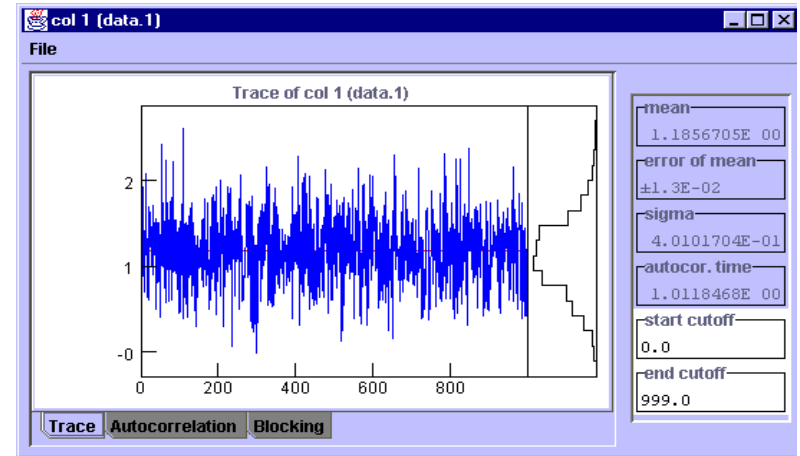
Interactive code to  
perform statistical  
analysis of data



# Correlated data



# Uncorrelated data





# Estimating Errors

- Uncorrelated data

$$\{a_t\} \quad 0 < t \leq N$$

$$\langle a_t \rangle \approx \bar{a} = \frac{1}{N} \sum_t a_t$$

$$error(\bar{a}) = \left\langle (\bar{a} - \langle a \rangle)^2 \right\rangle^{1/2} \approx \left[ \frac{\sum_t \delta a_t^2}{N(N-1)} \right]^{1/2}$$

$$\delta a_t \equiv a_t - \bar{a}$$

- Correlated data

$$error(\bar{a}) = \left\langle (\bar{a} - \langle a \rangle)^2 \right\rangle^{1/2} \approx \left\langle \frac{\kappa \sum_t (a_t - \bar{a})^2}{N(N-1)} \right\rangle^{1/2}$$

$$\kappa = 1 + 2 \sum_{t=1}^{\infty} \frac{\langle \delta a_t \delta a_0 \rangle}{\langle \delta a^2 \rangle} = \text{correlation time}$$

- Blocking method: average together data in blocks longer than the correlation time until it is uncorrelated.

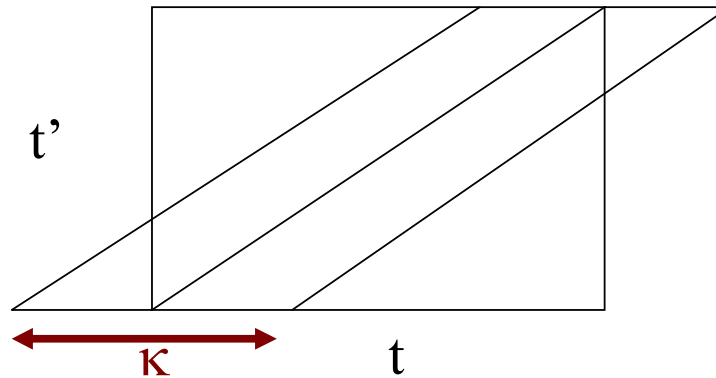
# Proof of variance estimator

$$\text{error}(\bar{a}) = \left\langle (\bar{a} - \langle a \rangle)^2 \right\rangle^{1/2} \approx \left\langle \frac{\kappa \sum (a_t - \bar{a})^2}{N(N-1)} \right\rangle^{1/2}$$

$$\kappa = 1 + 2 \sum_{t=1}^{\infty} \frac{\langle \delta a_t \delta a_0 \rangle}{\langle \delta a^2 \rangle} = \text{correlation time} \approx 2 \int_0^{\infty} \frac{dt}{\delta t} C(t)$$

$$C(t, t') \equiv \frac{\langle \delta a_t \delta a_{t'} \rangle}{\langle \delta a^2 \rangle} = C(|t - t'|) = \text{autocorrelation function}$$

$$\left\langle (\bar{a} - \langle a \rangle)^2 \right\rangle = \left\langle \frac{1}{N^2} \sum_{t, t'}^N \delta a_t \delta a_{t'} \right\rangle = \frac{\langle \delta a^2 \rangle}{N^2} \sum_{t, t'}^N C_{|t-t'|} \approx \frac{\langle \delta a^2 \rangle}{N^2} \sum_{t'=1}^N \sum_{t=-\infty}^{\infty} C_t = \langle \delta a^2 \rangle \frac{\kappa}{N}$$



# Statistical thinking is slippery: be careful

- “Shouldn’t the energy settle down to a constant”
  - NO. It fluctuates forever. It is the overall mean which converges.
- “The cumulative energy has converged”.
  - BEWARE. Even pathological cases have smooth cumulative energy curves.
- “Data set A differs from B by 2 error bars. Therefore it must be different”.
  - This is normal in 1 out of 10 cases. **If things agree too well, something is wrong!**
- “My procedure is too complicated to compute errors”
  - **NO**. Run your whole code 10 times and compute the mean and variance from the different runs. If it is important, you **MUST** estimate its errors.