Quantum Monte Carlo

1. Introduction to Monte Carlo: sampling, random numbers, Markov chains, estimating errors
2. Variational Monte Carlo: sampling and wavefunctions.
3. Diffusion Monte Carlo: branching random walks, fermion sign problem
4. Introduction to Path Integrals: formalism, sampling, the action.
5. Boson & Fermion Path Integrals: permutations, exchange moves, superfluidity and bose condensation.

Goal: to solve quantum many-body systems with computer simulation.

Examples: liquid and solid helium, electron gas, hydrogen,…

I only will cover continuum quantum Monte Carlo, not lattice models!

Monte Carlo and Random Walks

Today we start with an introduction to 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 randomly sampling an integral.
  - Often a Markov chain, in particular Metropolis MC
- References
  - *Allen & Tildesley "Computer Simulation of Liquids"*
  - *Frenkel & Smit "Molecule Simulations"*
  - *Thijssen, "Computational Physics"*
  - *Kalos & Whitlock, "Monte Carlo Methods"*
  - *"Numerical Recipes"*

MC is advantageous for high dimensional integrals

Consider an integral in the unit hypercube:

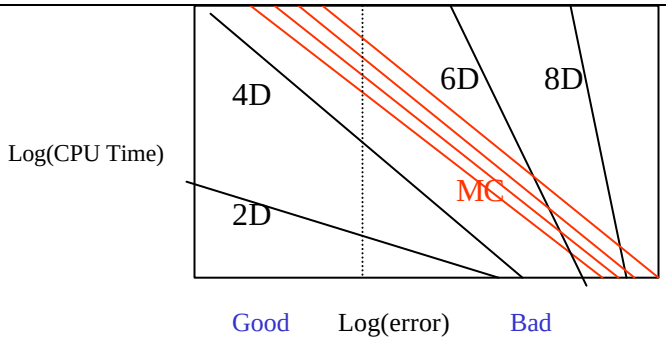
$$I = \int_0^1 dx_1 ... dx_D f(x_1, ... x_D)$$

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 trapizoidal rule goes as $\varepsilon = f''(x)\, h^2$.
- The CPU time $\propto \varepsilon^{-D/2}$.
- By sampling CPU time $\propto \varepsilon^{-2}$. 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

Central Limit Theorem (Gauss)

Sample N values from p(x)dx. (x_1, x_2, x_3,... x_N)

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

What is the pdf of mean?

Solve by fourier transforms.

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

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

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

Then

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

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

Taylor expand:

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

cumulants

cumulants k_n

mean=k_1

variance=k_2

skewness =k_3

kurtosis=k_4

What happens to the reduced moments?

$$\hat{k}_n = k_n N^{1-n}$$

Hence the n=1 moment remains invariant.

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

$$\lim_{N\to\infty} c_y(k) = e^{ikk_1 - k^2 k_2/2N - ik^3 k_3/6N^2 \ldots}$$

set $k_3 = k_4 = \ldots = 0$ and fourier transform

$$p(y) = \left(N/2\pi k_2\right)^{1/2} e^{-\frac{N(y-k_1)^2}{2k_2}}$$

Given enough averaging almost anything becomes a Gaussian distribution.

Approach to normality

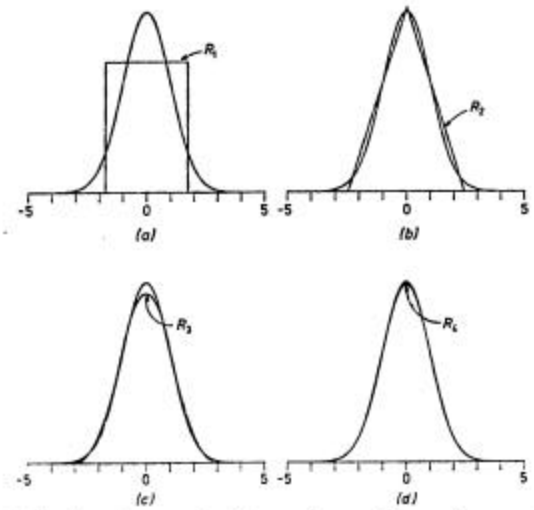


Figure 1. Distributions of sums of uniform random numbers, each compared with the normal distribution. (*a*) R_1, the uniform distribution. (*b*) R_2, the sum of two uniformly distributed numbers. (*c*) R_3, the sum of three uniformly distributed numbers. (*d*) R_{12}, the sum of twelve uniformly distributed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

Conditions on Central Limit Theorem

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

- We need the first three moments to exist.
  - If I_0 is not defined⇒p(x) not a pdf
  - If I_1 does not exist ⇒integral not mathematically well-posed.
  - If I_2 does not exist ⇒ infinite variance. Important to know if variance is finite for Monte Carlo.
- Divergence could happen because of tails of distribution

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

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

$$\lim_{x \to \pm 0} x^3 p(x) \to \textit{finite}$$

Random Number Generation

Also read "Numerical Recipes".

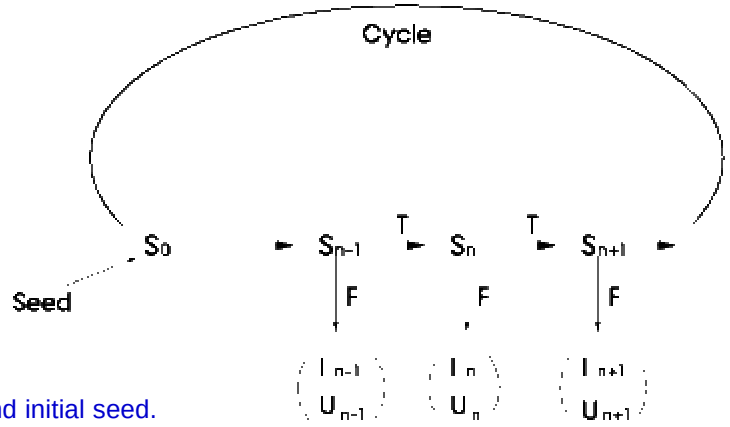
What is a random number?

- 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.

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) ~ 100 second
  - drand48 (1 processor) ~ 1 year
  - drand48 (100 processor) ~ 3 days
  - SPRNG LFG (10^5 procs) ~ 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!

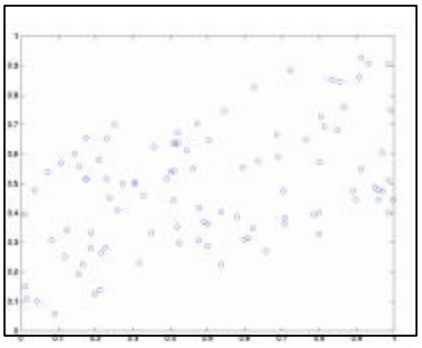
Common PRNG Generators

• 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
	Recurrence	Parallelization

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:

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

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

Recommendation: for very 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

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:

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

Optimal sampling: $\frac{du}{d\,p(x)} = 0$ 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: $$f(x)/p^*(x) = \frac{sign(f(x))}{\int dx |f(x)|}$$

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

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

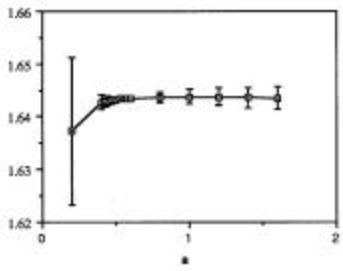
Importance sampling is a general technique: it works in many dimensions.

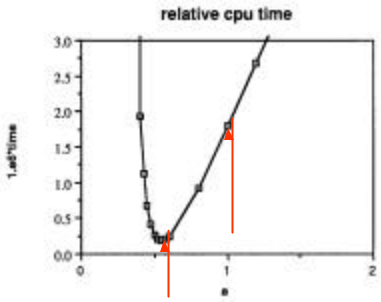
Example of important sampling.

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

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

Mean value is independent of a. CPU time is not

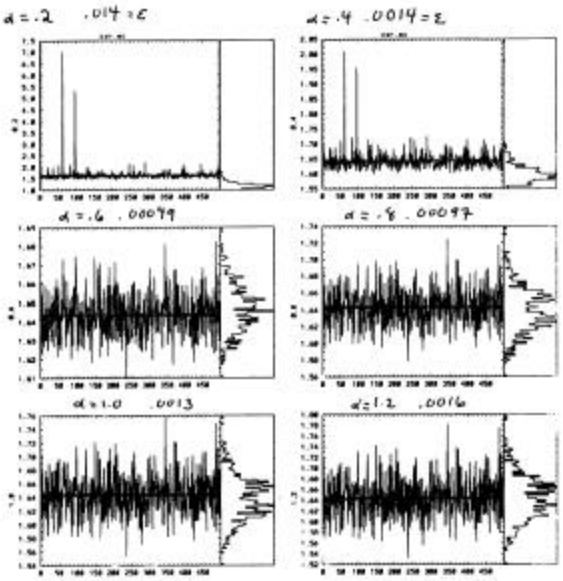




What does infinite variance look like?

Spikes

Long tails on the distributions



- 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.
- Do not ignore the spikes—understand them!
- *Sign problem*: zero variance is not possible for oscillatory integral. *"Monte Carlo can add but not subtract."*

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. (1953) applied to a hard sphere liquid.
- One of the most powerful and most used algorithms in science.

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

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.

Markov chain or Random Walk

- Markov chain is a random walk through phase space:

  $s_1 \to s_2 \to s_3 \to s_4 \to \ldots$

  Here "*s*" is the state of the system
- The transition probability is: $P(s_n \to s_{n+1})$ *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."

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

- The stationary states are eigenfunctions of *P*.

$$\sum_s p(s) P(s \to s') = \varepsilon p(s')$$

- 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.

$$f_n(s) = p(s) + \sum_l \varepsilon_l^n c_l f_l(s)$$

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: $p(s)\,P(s \to s') = p(s')\,P(s' \to s)$.

Rate balance from s to s'.

Put $p(s)$ into the master equation.

$$\sum_s p(s)P(s \to s') = \sum_s p(s')P(s' \to s) = p(s')\sum_s P(s' \to s) = p(s')$$

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

Rejection Method

Metropolis achieves detailed balance by *rejecting* moves.

Break up transition probability into sampling and acceptance:

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

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

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

The optimal acceptance probability that gives detailed balance is:

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

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→s')= constant
- Accept with move with probability:

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

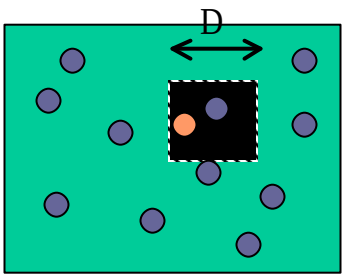
- Repeat many times

- 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_new = S_old + Δ (sprng - 0.5)

Uniform distribution in a cube of side "Δ".



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.

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

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!
3. No a priori criterion for deciding how long to run:
   - Try different starting places
   - Occasionally, do a very long run
   - Think about the physics
   - Monitor lots of different properties, including order parameters

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.

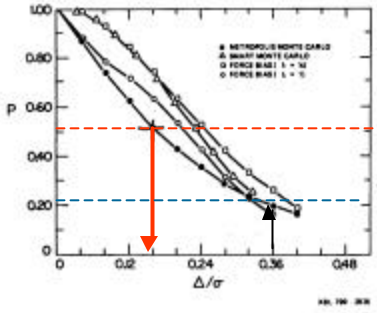


Fig. 1. Average acceptance probability.

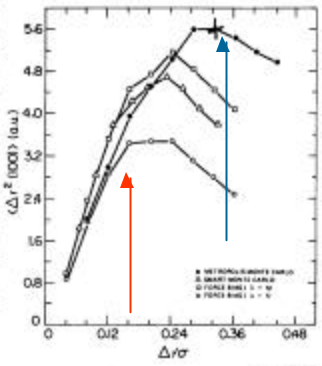


Fig. 3. $\langle(\vec{r}(i) - \vec{r}(i+100))^2\rangle$

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

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.

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

- Set $T(s \Rightarrow s') \approx \exp[-\beta(V(r_0) - F(r_0)(r-r_o))]$
- Leads to Force-Bias Monte Carlo
- Related to Brownian motion (Smoluchowski Eq.)
- Local implementation is called the "heat bath" method.
- Will use in Diffusion Quantum Monte Carlo.

Problems with estimating errors

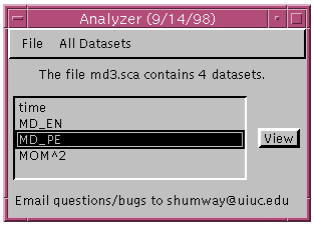
- Any good simulation quotes systematic and statistical errors for anything important.
- The error and mean are simultaneously determined from the same data. HOW?
- Central limit theorem: the distribution of an average approaches a normal distribution (if the variance is finite). One standard deviation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

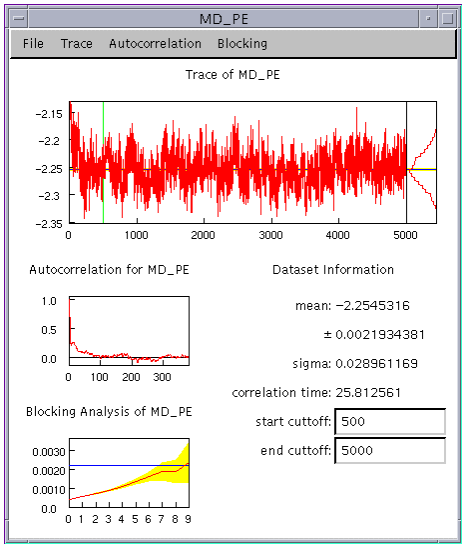
DataSpork

You need to be able to look at the data and estimate the error bars.

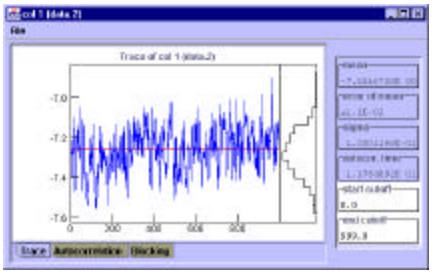
Interactive code to perform statistical analysis of data.

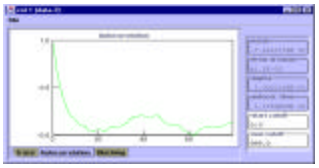
Download from Materials Computation Center at U of Illinois.



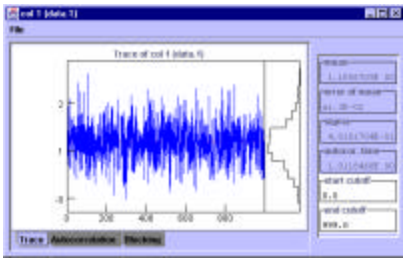


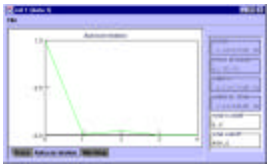
Correlated data





Uncorrelated data





Estimating Errors

- Uncorrelated data

$$\{a_t\} \qquad 0 < t \le N$$

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

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

$$da_t = a_t - \bar{a}$$

- Correlated data

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

$$k = 1 + 2\sum_{t=1}^{\infty} \frac{\langle da_t da_0 \rangle}{\langle da^2 \rangle} = \text{correlation time}$$

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

Proof of variance estimate

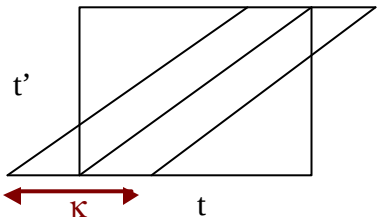
$$error(\bar{a}) = \left\langle \left(\bar{a} - \langle a \rangle\right)^2 \right\rangle^{1/2} \approx \left\langle \frac{k \sum_t (a_t - \bar{a})^2}{N(N-1)} \right\rangle^{1/2}$$

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

$$k = 1 + 2\sum_{t=1}^{\infty} C(t) = \text{correlation time} \approx 2\int_0^{\infty} \frac{dt}{dt} \mathrm{C(t)}$$

$$C(t,t') \equiv \frac{\langle da_t da_{t'} \rangle}{\langle da^2 \rangle} = C(|t - t'|) = \text{autocorrelation function}$$

$$\left\langle \left(\bar{a} - \langle a \rangle\right)^2 \right\rangle = \left\langle \frac{1}{N^2} \sum_{t,t'}^{N} da_t da_{t'} \right\rangle = \frac{\langle da^2 \rangle}{N^2} \sum_{t,t'}^{N} C_{|t'-t|} < \frac{\langle da^2 \rangle}{N^2} \sum_{t'=1}^{N} \sum_{t=-\infty}^{\infty} C_t = \langle da^2 \rangle \frac{k}{N}$$



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:** ΣA(t)/N
- **estimate of the variance**
- **Autocorrelation** of A (C(t)).
- **Correlation time** (κ).
- The (estimated) **error** of the (estimated) **mean** (σ).
- **Efficiency** [= 1/(CPU time * error ^{2})]

Efficiency of MC

statistical error ~ sqrt(variance/computer time).

DEFINE:

$$\text{efficiency} = z = \frac{1}{uT}$$

$n = \text{error}^2$ of mean

$T = \text{total computer time}$

(independent of the length of run)

To improve efficiency, one can either:

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

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 a quantity is important, you MUST estimate its errors.