Monte Carlo integration

Lode Pollet

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

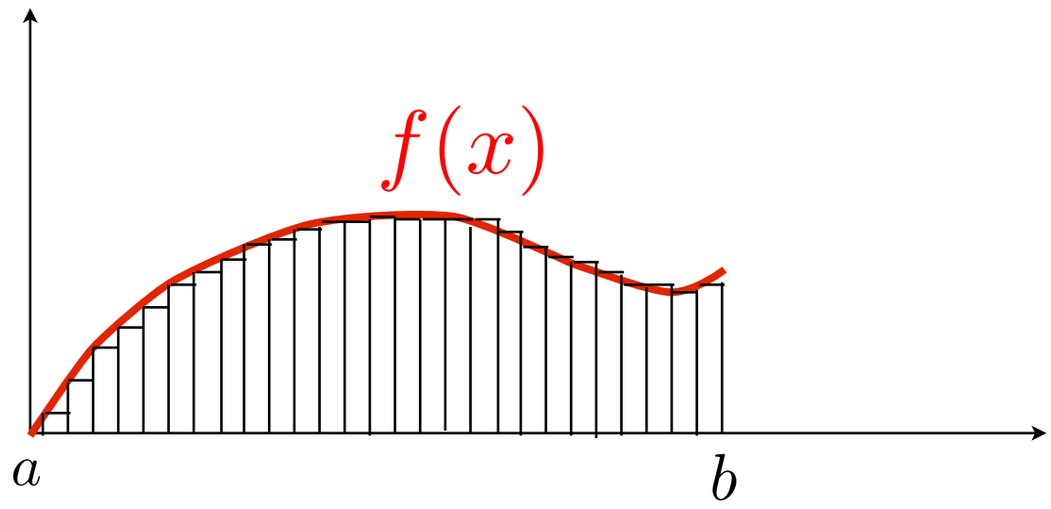
erc European Research Council
Established by the European Commission
Supporting top researchers from anywhere in the world

nim nanosystems initiative munich

MUNICH QUANTUM CENTER

Why?

$$h = \frac{b-a}{N}$$



discrete sum:

$$I = \int_a^b f(x)dx = h\sum_{i=1}^{N} f(a+ih) + \mathcal{O}(1/N)$$

higher order integrators

trapezoidal $$I = h\left(\frac{1}{2}f(a) + \sum_{i=1}^{N-1} f(a+ih) + \frac{1}{2}f(b)\right) \quad + \mathcal{O}(1/N^2)$$

Simpson $$I = \frac{h}{3}\left(f(a) + \sum_{i=1}^{N-1}(3-(-1)^i)f(a+ih) + f(b)\right) \quad + \mathcal{O}(1/N^4)$$

(N even)

Why?

consider multi-dimensional integration

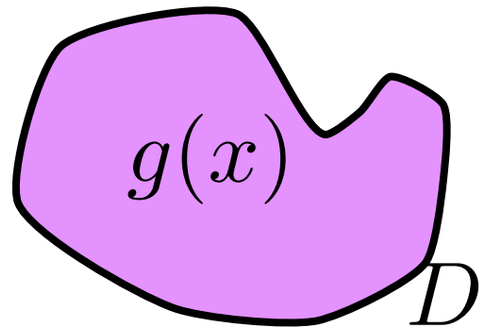
Convergence in higher dimensions is slow

for N points in d dimensions, the spacing is $N^{-1/d}$

Simpson rule converges hence as $N^{-4/d}$

In classical statistical mechanics the integrals are 6N-dimensional
(3N positions, 3N momenta)

Monte Carlo integration



$$I = \int_D g(x)dx$$

identical and independent random samples uniformly drawn from *D*

$$\hat{I}_m = \frac{1}{m}\left(g(x^{(1)}) + \cdots + g(x^{(m)})\right)$$

law of the large numbers

$$P\left[\lim_{m\to\infty} |\hat{I}_m - I| = 0\right] = 1$$

Central limit theorem

The mean of a large number of iid samples, each with well-defined value and variance, will be approximately normal distributed independent of the underlying distribution

central limit theorem

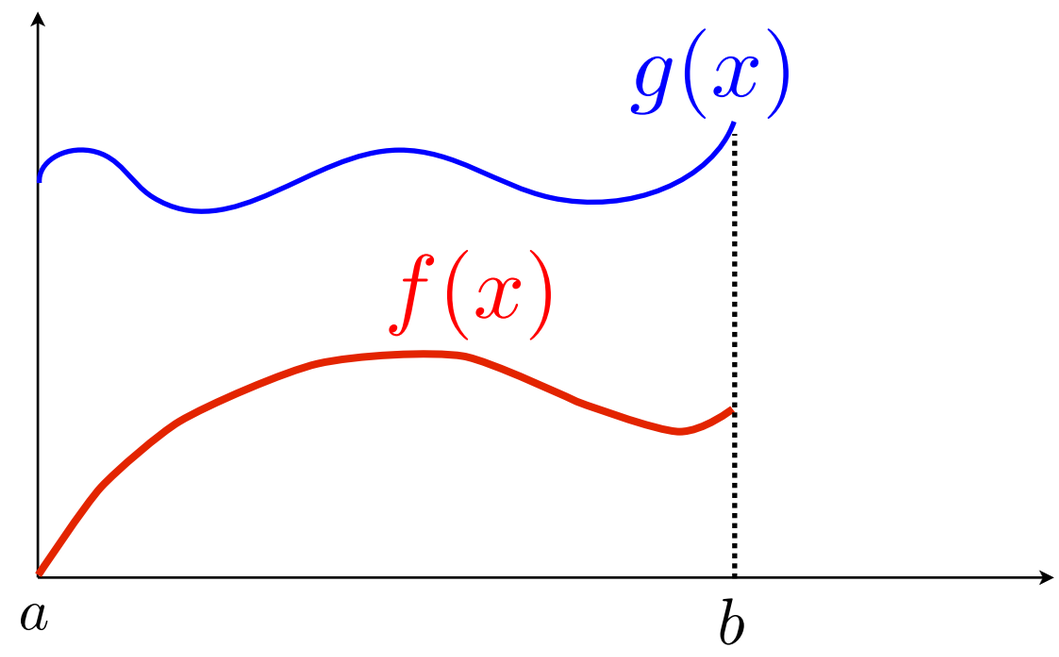
$$\lim_{m\to\infty} \mathrm{Cdf}\sqrt{m}\left(\hat{I}_m - I\right) = \mathrm{Cdf}\,N(0,\sigma^2)$$

convergence is

$$\mathcal{O}(m^{-1/2})$$

regardless of the dimensionality of *D*

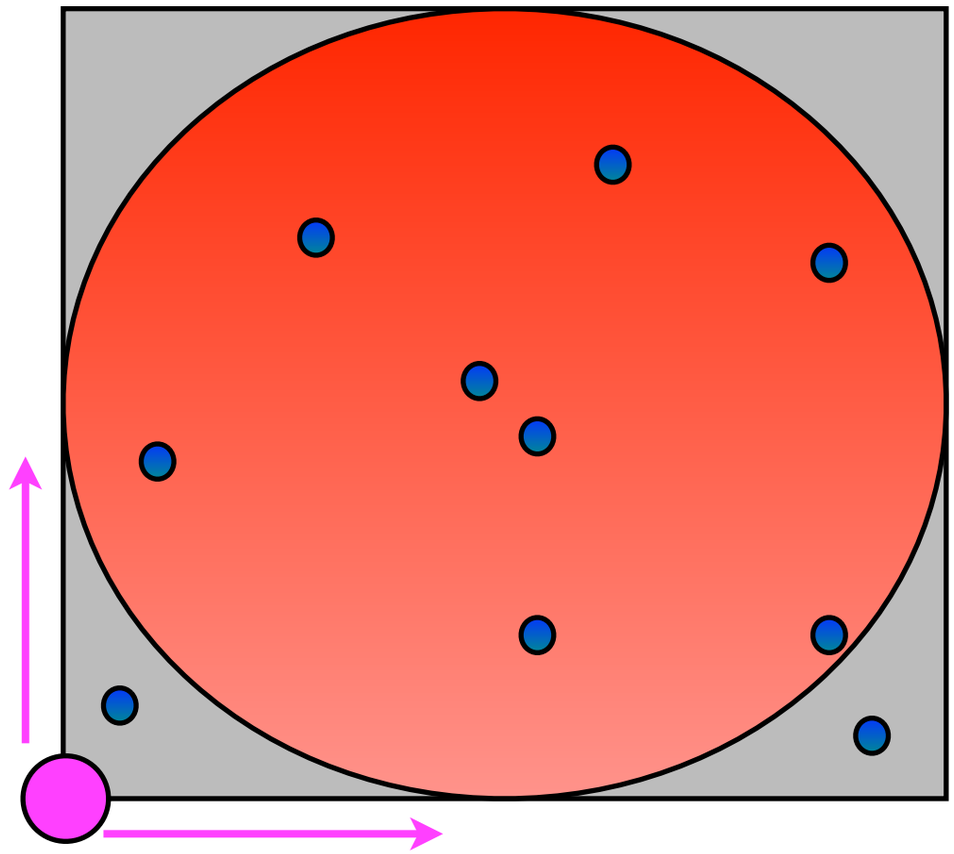
Von Neumann rejection



say f(x) hard and g(x) easy to
generate random numbers from
and g(x) >= f(x) everywhere

then generate random numbers x_i between [a,b] and [0,g(x_i)].
If it falls below the function f, keep it otherwise reject it

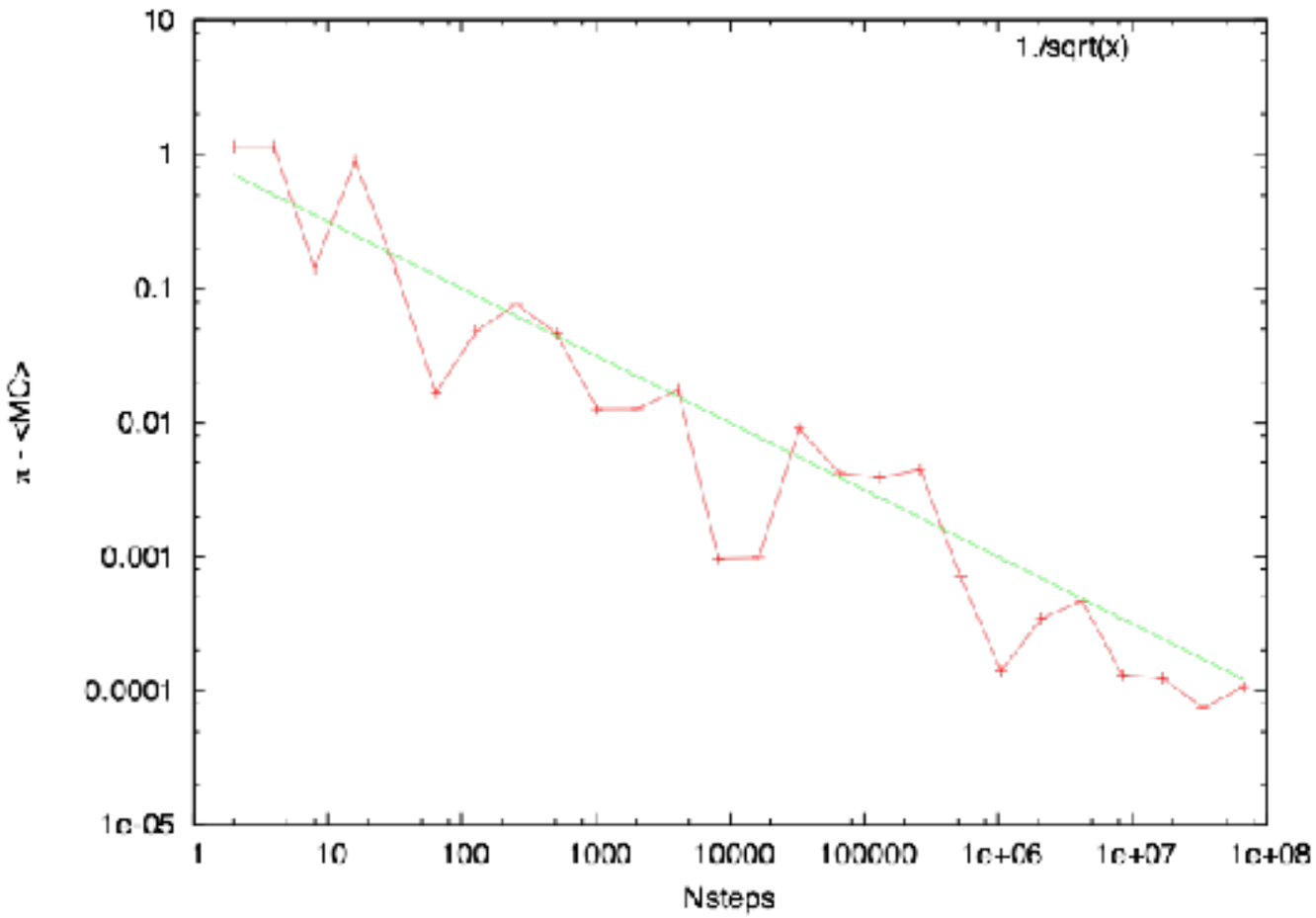
Direct sampling



Known geometry
converges to π/4

Direct sampling

```cpp
#include <iostream>
#include <random>
#include <iomanip>
#include <math.h>

using namespace std;

int main() {
  cout << setprecision(10);
  random_device rd;
  mt19937 MyGenerator(rd());
  uniform_real_distribution<double> rnd(0,1);
  MyGenerator.seed(40);

  double iprint = 2.;
  double s = 0.;
  double inc = 0.;
  for (;;) {
    double x = 2*rnd(MyGenerator) - 1;
    double y = 2*rnd(MyGenerator) - 1;
    if (x*x + y*y < 1) s+= 1;
    inc += 1;
    if (inc == iprint) {
      cout << inc << "\t" << s/inc*4 << "\t" << M_PI << "\t" << abs(M_PI - s/inc*4) << "\n";
      iprint *= 2;
    }
    if (inc >= 1e8) break;

  }
  return(0);

}
```



Importance sampling

$$\langle f \rangle = \frac{1}{V}\int f(x)dx = \frac{1}{V}\int \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

$$\Delta = \sqrt{\frac{\mathrm{Var}\, f/p}{N}}$$

- imagine function f is sharply peaked
- then the variance can be reduced by finding p(x) such that p(x) is close to f(x) and that it is easy to generate random numbers according to p(x)

Importance sampling

$$I = \int_0^\infty g(x) e^{-x} dx$$

draw random numbers that are exponentially distributed, then

$$\hat{I}_m = \frac{1}{m}\left(g(x^{(1)}) + \cdots + g(x^{(m)})\right)$$

how?

$u \in [0, 1[$ standard random number generator

$p = -\ln u \quad \Longrightarrow \quad p \in [0, \infty[$

inverse transformation is needed

change of variables

general

$$y = F(y) = \int_0^y f(u)du$$

$$x = F^{-1}(y)$$

$$f(x) = x$$

$$y = F(x) = \int_0^x x'dx' = \frac{x^2}{2}$$

$$x = 2\sqrt{y}$$

exponential

$$p = -\ln(u)$$

Box-Mueller (gaussian)

$$p_1 = R\cos(\theta) = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$$

$$p_2 = R\sin(\theta) = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$$

Link with statistical mechanics

$$\langle Q \rangle = \frac{1}{Z} \mathrm{Tr}\left[Q e^{-\beta H}\right] = \frac{\mathrm{Tr}[Q e^{-\beta H}]}{\mathrm{Tr}[e^{-\beta H}]}$$

unnormalized weights

$$W(x) = e^{-\beta E(x)}$$

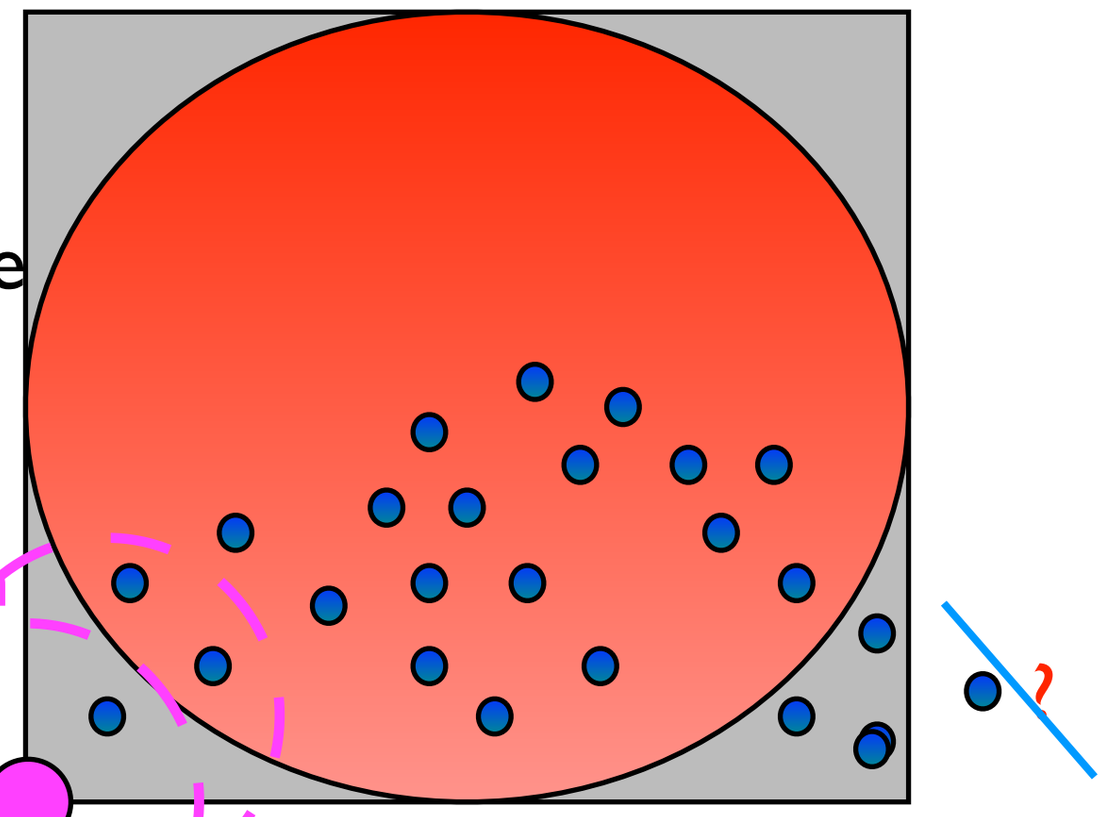
how do we get random variables that are distributed according to W(x) ?

Markov Chains

Small steps random walk through configuration space

at each time : measure

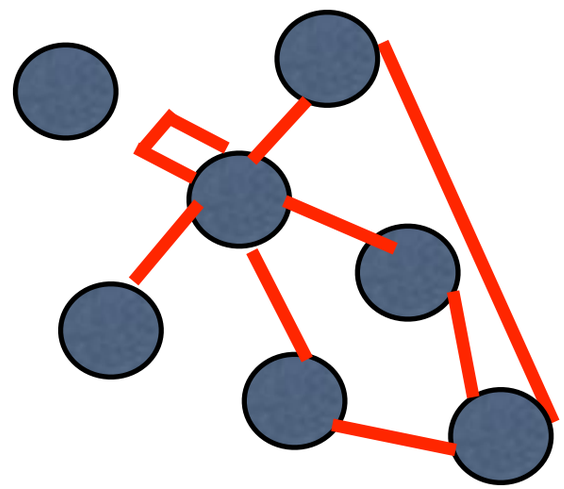
transition function



Rejection : stay at same configuration, update clock and measure

Markov Chains

- A Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the property that the future state depends only on the past via the present state.



$$P[X_{n+1} = x | X_n = x_n, \ldots, X_1 = x_1, X_0 = x_0] = P[X_{n+1} = x | X_n = x_n]$$

transition function

$$T(x, y)$$

$$\sum_y T(x, y) = 1$$

conservation of probability

irreducible : it must be possible to reach any configuration x from any other configuration y in a finite number of steps.

Markov Chains

- irreducible
- aperiodic

} convergence to the stationary distribution W

transition kernel has one eigenvalue 1,
while all other eigenvalues satisfy

$$|\lambda_j| < 1, j = 2, \ldots N$$

The second largest eigenvalue
determines the correlations in the
Markov process

Detailed balance

A transition rule *T(x,y)* leaves the target distribution *W(x)* invariant if

$$\sum_x W(x)T(x,y) \sim W(y)$$

This will certainly be the case if detailed balance is fulfilled,

$$W(x)T(x,y) = W(y)T(y,x)$$

Metropolis algorithm

we cannot construct a
transition kernel T that fulfills
detailed balance.

proposal function $P(x,y)$

acceptance factor

$$q = \min\left[1, \frac{W(y)P(y,x)}{W(x)P(x,y)}\right]$$

go to the proposed configuration
y with prob q, otherwise stay in x

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,

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† con-

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,

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† con-

Peskun ordering

Peskun theorem: let T^A and T^B be two transition kernels that satisfy detailed balance (and are properly normalized). Let all the off-diagonal elements of the TB be larger than the corresponding ones of T^A. Then T^B will lead to a smaller asymptotic variance for all observables than T^A

$$\lambda_2^A > \lambda_2^B$$

for example let there be 2 weights with $W_1 < W_2$. Then the most efficient sampler is

$$T_{ij} = \begin{bmatrix} 0 & 1 \\ \frac{W_1}{W_2} & 1 - \frac{W_1}{W_2} \end{bmatrix}$$

it has an eigenvalue $\lambda_2 = -\frac{W_1}{W_2}$

Note that this is precisely the Metropolis algorithm! (note: Metropolis is for more than 2 weights not the most efficient sampler)

> Note that we have found a case where the second largest eigenvalue is smaller than in direct sampling; i.e., we are sampling *more efficiently* with correlated measurements!

heat bath

Another choice corresponds to

$$T_{ij} = \begin{bmatrix} \frac{W_1}{W_1+W_2} & \frac{W_2}{W_1+W_2} \\ \frac{W_1}{W_1+W_2} & \frac{W_2}{W_1+W_2} \end{bmatrix}$$

this corresponds to heat-bath updates. Its second largest eigenvalue is 0. To compute the transition probability to state j, compute the weight of state j and normalize wrt the weights of all reachable states.

Peskun's theorem implies an *ordering of the weights*, and tells us that we can have zero's on the diagonal elements (except for the largest weight). This ‚metropolizing' procedure can be carried out iteratively, and one shows that the best possible sampler is

$$T_{ij}^{Opt} = \begin{bmatrix} 0 & \frac{W_2}{W_1}y_1 & \frac{W_3}{W_1}y_1 & \dots & \frac{W_n}{W_1}y_1 \\ y_1 & 0 & \frac{W_3}{W_2}y_2 & \dots & \frac{W_n}{W_2}y_2 \\ y_1 & y_2 & 0 & \dots & \frac{W_n}{W_3}y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_1 & y_2 & y_3 & \dots & 1-y_1-y_2-\dots \end{bmatrix}$$

$$y_1 = \frac{\pi_1}{1-\pi_1} \qquad \lambda_2 = -y_1$$

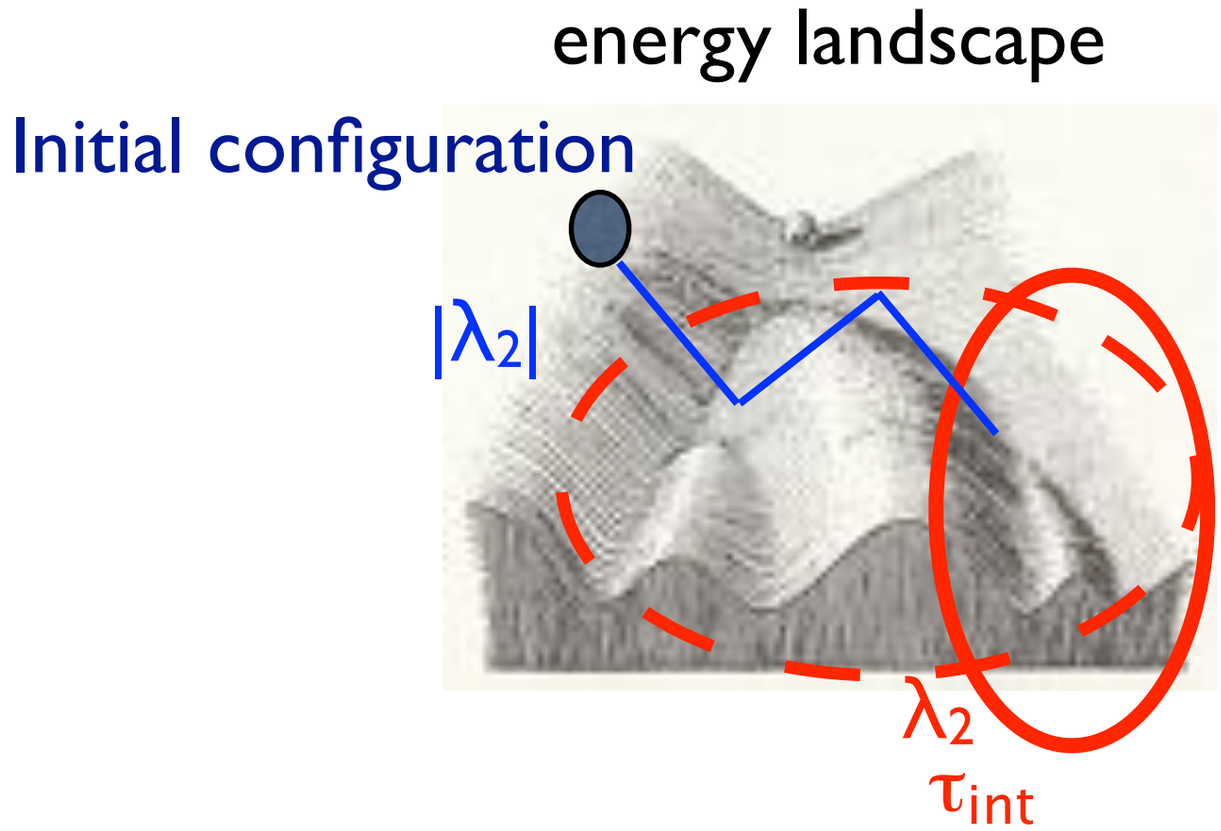
$$y_2 = (1-y_1)\frac{\pi_2}{1-\pi_1-\pi_2}$$

etc

drawbacks: you need to know all weights (over all phase space) and be able to order and normalize them; i.e. impossible in practice for realistic problems

L. Pollet et al., Phys. Rev. E 70, 056705 (2004)

Markov chain



Discard the first 20% of the Markov steps !

Markov Chain should be sufficiently long

Homework

calculate (mean value, no error analysis) :

$$I = \int_0^6 \exp(-x/2) dx$$

by :

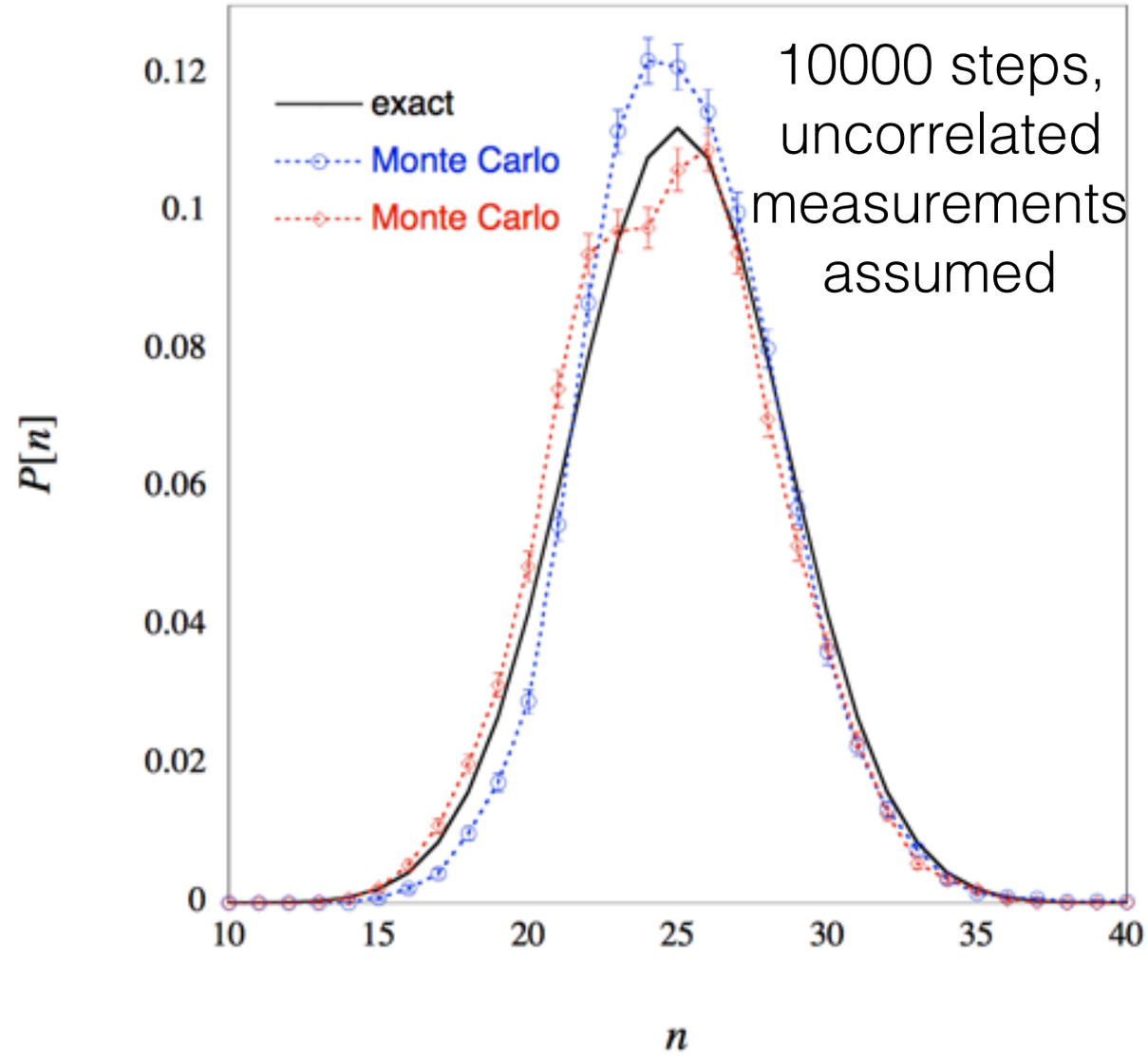
1. direct integration (analytical/Monte Carlo by exponentially distributed random numbers)
2. Markov chain Monte Carlo : choose a step size d (wisely), and update the current position according to the Metropolis algorithm by choosing a random step of. Don't forget that in every step you can move to larger or smaller x values. Show that you satisfy detailed balance.
3. modification (advanced). Suppose the following. Suppose you start with moving to the right. If you accept the move, then always choose moving to the right in the next step. If you reject the move, then start moving to the left. So, you keep moving in one direction until you have a rejection (bounce), and then you change the direction and keep moving in this direction until there is another bounce after which you change the direction again. Does this Monte Carlo Markov chain produce the right answer ?

Dogs and fleas model

http://arxiv.org/pdf/0906.0943.pdf

- Two dogs play:
  - Anick has 50 fleas
  - Burnside has no fleas
- During play fleas jump from one dog to the other;
  ie at every step one random flea jumps to the other dog
- what is the distribution of fleas after they played?

the shape is impossible and points at an error



autocorrelation function

Markov chains trivially correlate measurements

autocorrelations decay exponentially $\langle A_t A_{t+\Delta}\rangle - \langle A\rangle^2 \propto \exp(-\Delta/\tau_A^{(exp)})$

integrated autocorrelation time $\tau_A^{(int)} = \frac{\sum_{\Delta=1}^{\infty}\left(\langle A_t A_{t+\Delta}\rangle - \langle A\rangle^2\right)}{\langle A^2\rangle - \langle A\rangle^2}$

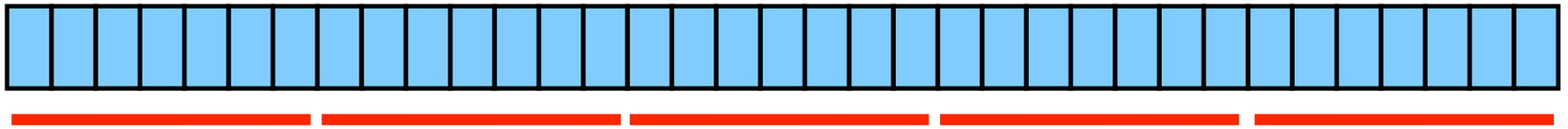
$$
\begin{aligned}
(\Delta A)^2 &= \langle(\overline{A} - \langle A\rangle)^2\rangle = \langle\left(\frac{1}{N}\sum_{t=1}^{N} A(t) - \langle A\rangle\right)^2\rangle \\
&= \langle\frac{1}{N^2}\sum_{i=1}^{N}\left(A(t)^2 - \langle A\rangle^2\right)\rangle + \frac{2}{N^2}\sum_{t=1}^{N}\sum_{\Delta=1}^{N-t}\left(\langle A(t)A(t+\Delta)\rangle - \langle < A >^2\rangle\right) \\
&\approx \frac{1}{N}\mathrm{Var}A\,(1 + 2\tau_A^{(int)}) \\
&\approx \frac{1}{N-1}\langle\overline{A^2} - \overline{A}^2\rangle(1 + 2\tau_A^{(int)})
\end{aligned}
$$

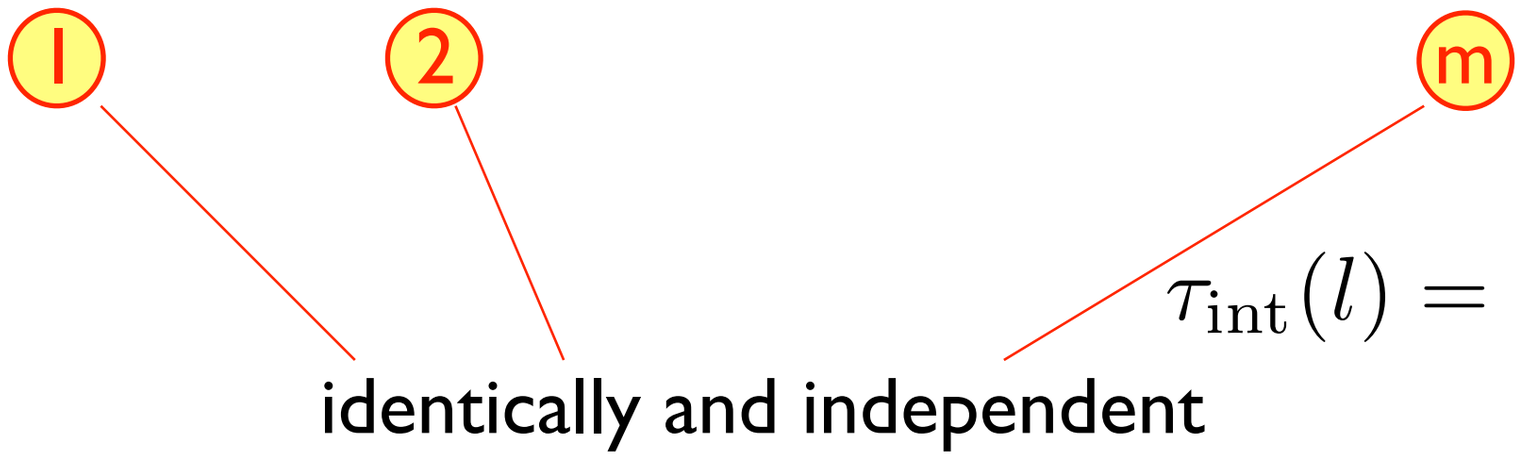
number of independent measurements is reduced, but central limit theorem still holds

Binning analysis

How to get correct error bars?

- Markov chain correlates measurements
- if chain is long enough, then the configuration is independent of the initial one

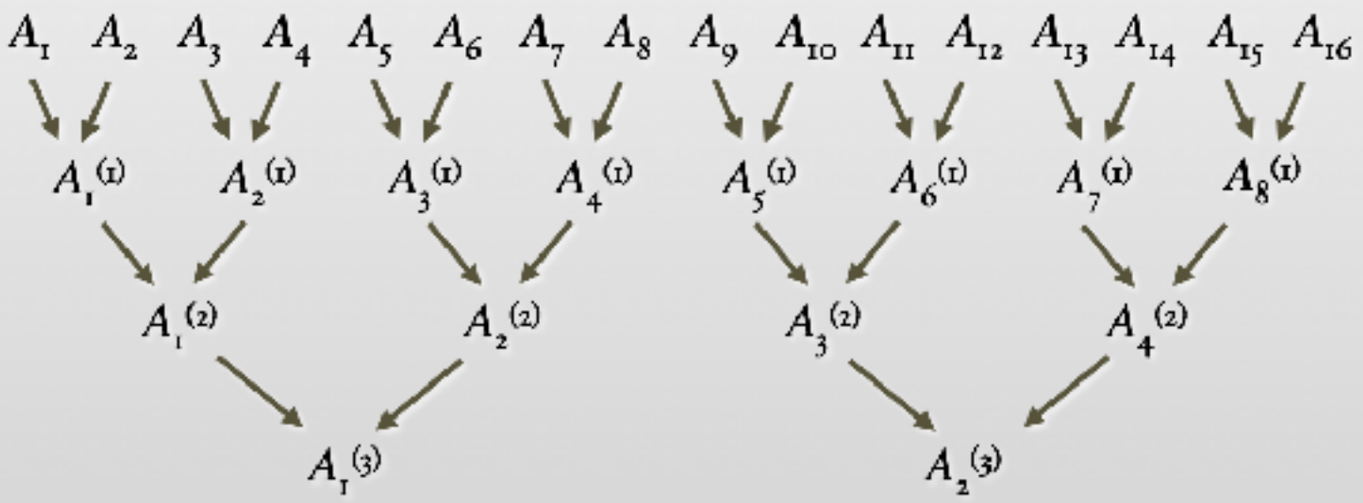




$$\tau_{\mathrm{int}}(l) = \frac{l\sigma^2(l)}{2\sigma^2}$$

http://arxiv.org/pdf/0906.0943.pdf; slides M. Troyer

- Take averages of consecutive measurements: averages become less correlated and naive error estimates converge to real error

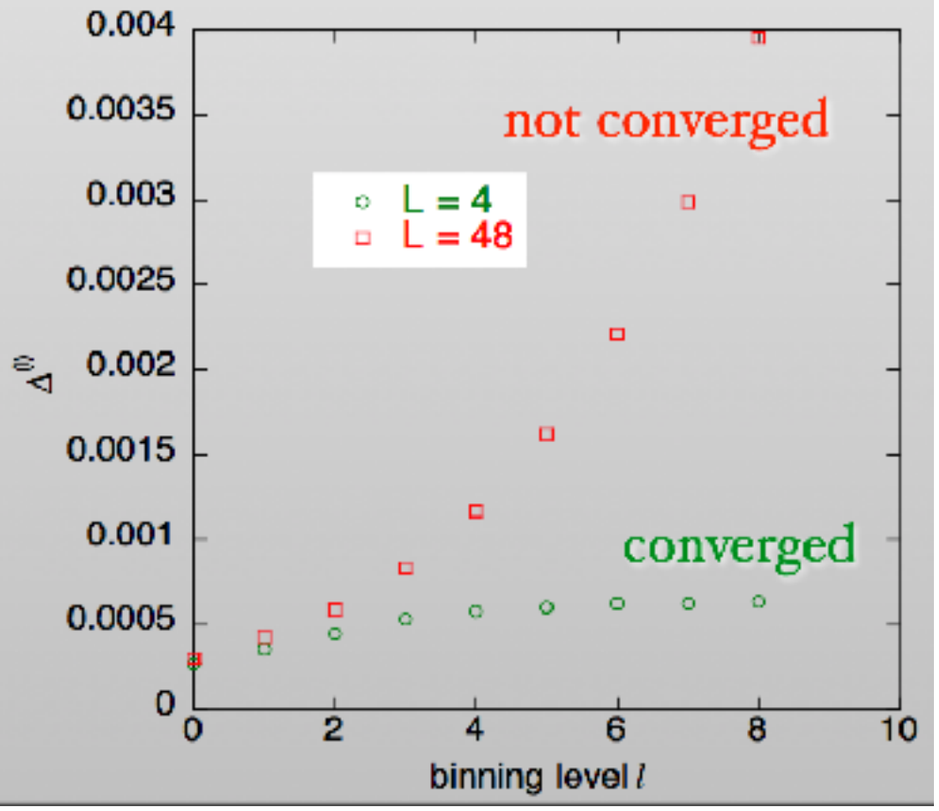


$$A_i^{(l)} = \frac{1}{2}\left(A_{2i-1}^{(l-1)} + A_{2i}^{l}\right)$$

$$\Delta^{(l)} = \sqrt{\operatorname{Var} A^{(l)}/M^{(l)}} \xrightarrow{l\to\infty} \Delta = \sqrt{(1+2\tau_A)\operatorname{Var} A/M}$$

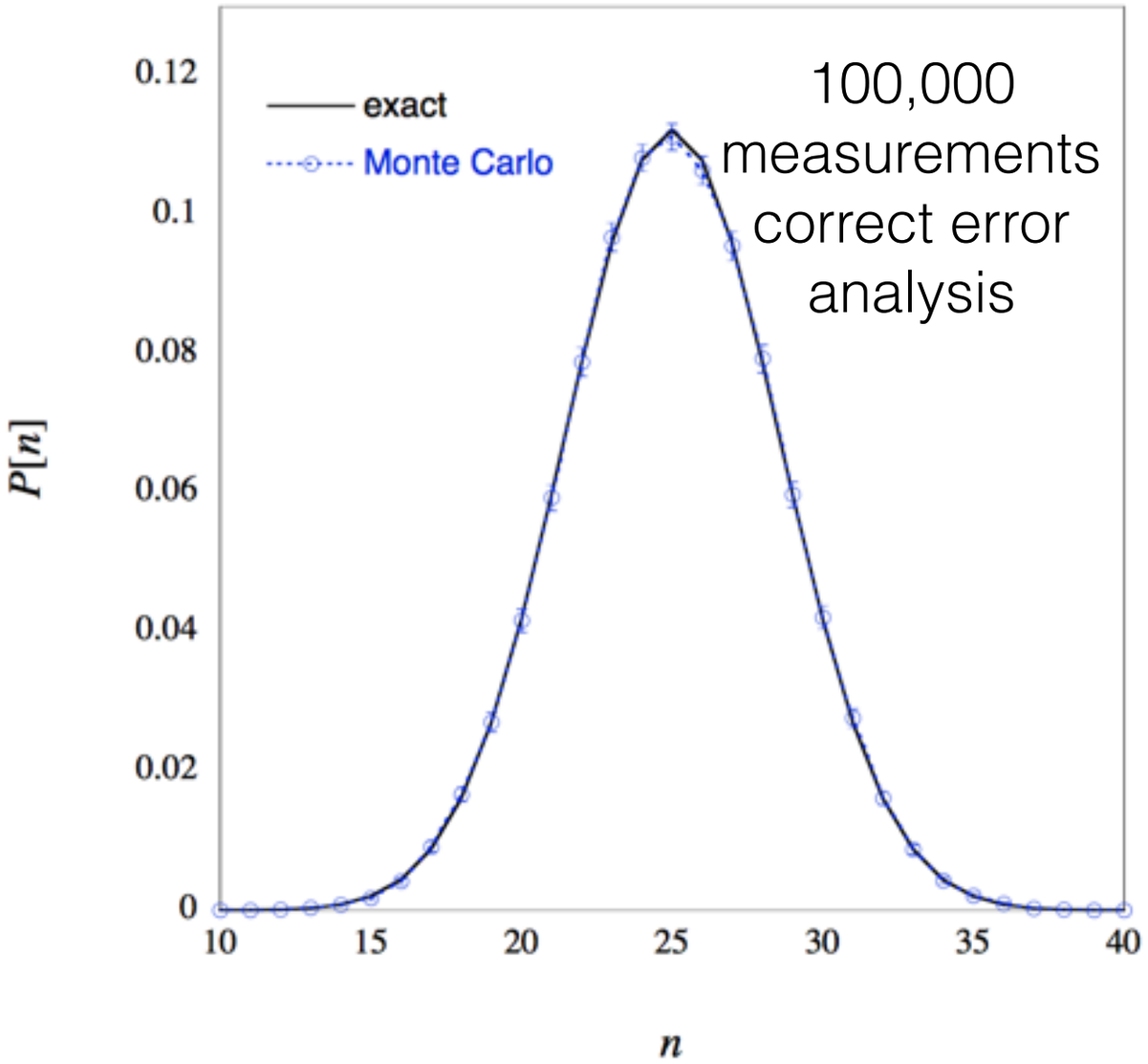
$$\tau_A = \lim_{l\to\infty} \frac{1}{2}\left(\frac{2^l \operatorname{Var} A^{(l)}}{\operatorname{Var} A^{(0)}} - 1\right)$$

a smart implementation needs only
O(log(*N*)) memory for *N* measurements

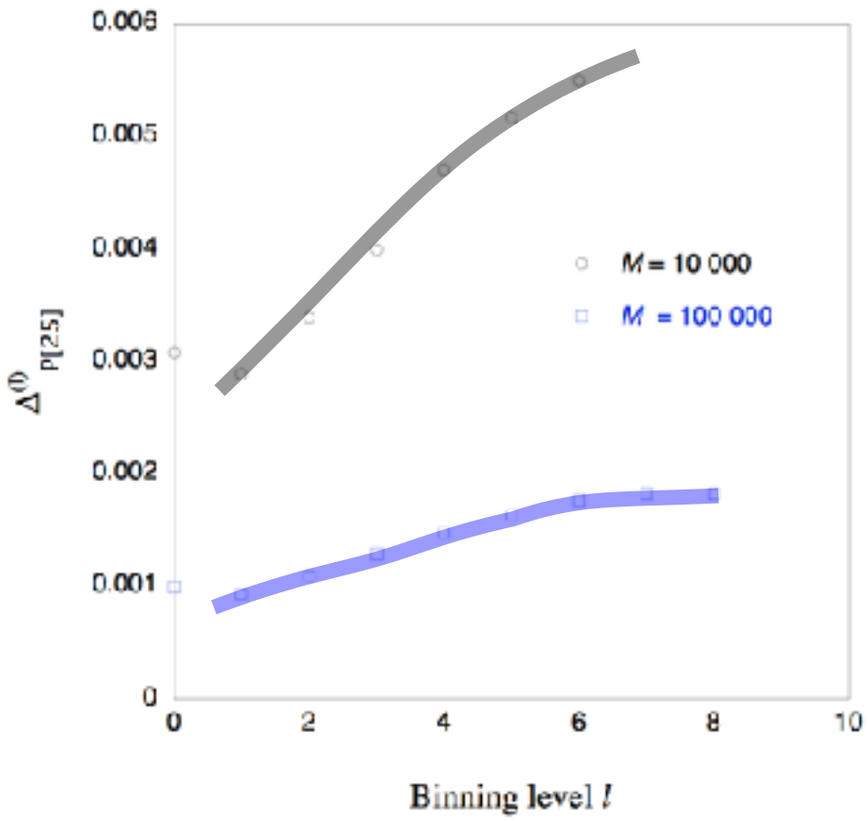


Dogs and fleas:binning analysis

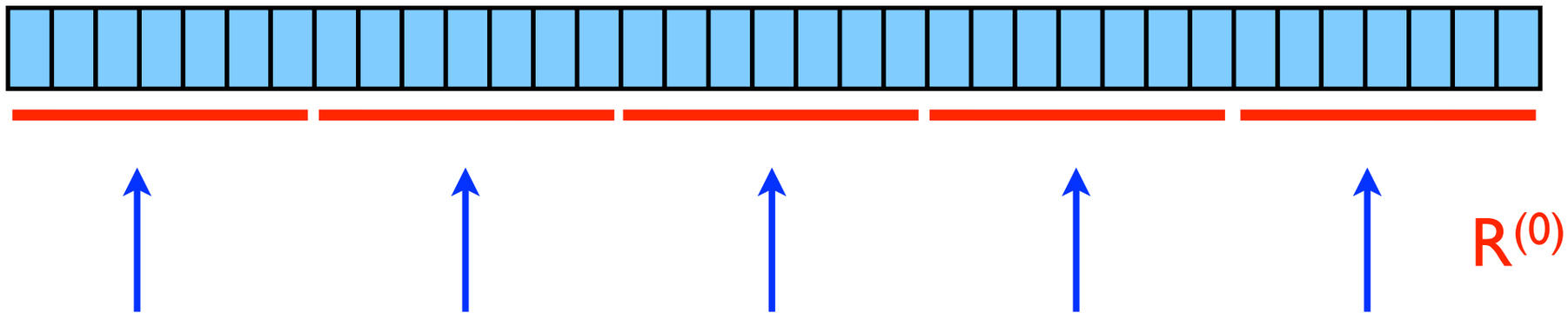
http://arxiv.org/pdf/0906.0943.pdf



using a correct binning analysis the Monte Carlo agrees within error bar with the exact solution



Jackknife analysis



$R^{(j)}$, j = 1, k

Bias = (k-1)(R^{av} - $R^{(0)}$) R = $R^{(0)}$ - Bias

$$\delta R = \sqrt{k-1}\left(\frac{1}{k}\sum_{j}^{k}(R^{(j)})^2 - (R^{\mathrm{av}})^2\right)^{1/2}$$

further reading: http://arxiv.org/abs/1210.3781

Autocorrelation effects

The Metropolis algorithm creates a Markov chain

$$c_1 \to c_2 \to c_3 \to \ldots c_n \to c_{n+1} \to \ldots$$

Successive measurements are correlated, leading to an increased statistical error

$$\Delta A = \sqrt{\langle (\bar{A} - \langle A \rangle)^2 \rangle} = \sqrt{\frac{\mathrm{Var} A}{M}(1 + 2\tau_A)}$$

At second order phase transitions one observes critically slowing down,

$$\tau \propto L^z$$

Typically, z is about 2. For the Ising model with single spin flop, z = 2.2(1)

At first order transitions, the tunneling rate between the free energy minima of the two co-existing phases is exponential

$$\tau \propto \exp(L^{d-1})$$

Continuous time quantum Monte Carlo

Lode Pollet
LMU Munich

Consider the Hamiltonian

$$H = h\sigma_z + \Gamma\sigma_x$$

This is a single-particle problem that can easily be diagonalized and is hence a very good problem for benchmarking our first continuous-time QMC algorithm

We are interested in computing $\langle\sigma_x\rangle$ and $\langle\sigma_z\rangle$ in a path-integral formulation.

We work in the σ_z basis,

$$\sigma_z \left|\uparrow\right\rangle = \left|\uparrow\right\rangle \qquad \sigma_x \left|\uparrow\right\rangle = \left|\downarrow\right\rangle$$

$$\sigma_z \left|\downarrow\right\rangle = \left|\downarrow\right\rangle \qquad \sigma_x \left|\downarrow\right\rangle = \left|\uparrow\right\rangle$$

The exact answers are

$$\langle\sigma_x\rangle = \frac{-\Gamma}{\sqrt{\Gamma^2 + h^2}} \tanh\left(\beta\sqrt{\Gamma^2 + h^2}\right)$$

$$\langle\sigma_z\rangle = \frac{-h}{\sqrt{\Gamma^2 + h^2}} \tanh\left(\beta\sqrt{\Gamma^2 + h^2}\right)$$

Starting from the partition function

$$Z = \mathrm{Tr} e^{-\beta H} = \sum_{|\alpha\rangle=|\uparrow\rangle,|\downarrow\rangle} \left\langle \alpha | e^{-\beta(h\sigma_z + \Gamma\sigma_z)} | \alpha \right\rangle$$

We see that the term with the field along the z-axis is diagonal in this basis.

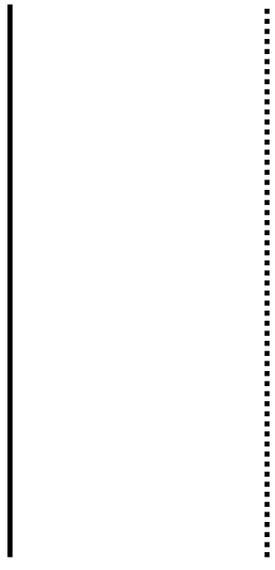
Introducing the interaction picture operators

$$\sigma_x(\tau) = e^{\tau h \sigma_z} \sigma_x e^{-\tau h \sigma_z}$$

the partition function can be written as

$$Z = \sum_{|\alpha\rangle = |\uparrow\rangle, |\downarrow\rangle} \sum_{n=0}^{\infty} \frac{(-\Gamma)^n}{n!} \int_0^\beta d\tau_1 \ldots \int_0^\beta d\tau_n \left\langle \alpha | e^{-\beta h \sigma_z} \mathcal{T}_\tau \left[\sigma_x(\tau_1) \sigma_x(\tau_2) \ldots \sigma_x(\tau_n) \right] \alpha \right\rangle$$

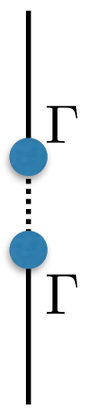
To lowest order (n=0) this is just σ_z propagating from 0 to β:



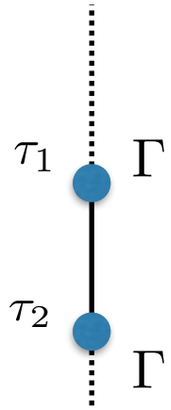
$$Z_0 = e^{-\beta h} + e^{\beta h}$$

To first order (n=1) there is no valid configuration that satisfies the trace. In other words, we need an even number of σ_x operators. The $(-1)^n$ factor will hence always be positive (+1) and there is no sign problem.

In second order we have the following



and



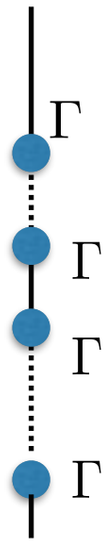
$$Z_2 = \Gamma^2 \sum_{|\alpha\rangle,|\alpha_a\rangle=|\uparrow\rangle,|\downarrow\rangle} \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \left\langle \alpha | e^{-\beta h\sigma_z} e^{\tau_1 h\sigma_z} \sigma_x | \alpha_1 \right\rangle \left\langle \alpha_1 | e^{-\tau_1 h\sigma_z} e^{\tau_2 h\sigma_z} \sigma_x e^{-\tau_2 h\sigma_z} | \alpha \right\rangle$$

$$= \Gamma^2 \sum_{|\alpha\rangle,|\alpha_a\rangle=|\uparrow\rangle,|\downarrow\rangle} \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \left\langle \alpha | e^{-(\beta-\tau_1) h\sigma_z} \sigma_x | \alpha_1 \right\rangle \left\langle \alpha_1 | e^{-(\tau_1-\tau_2) h\sigma_z} e^{\tau_2 h\sigma_z} \sigma_x e^{-\tau_2 h\sigma_z} | \alpha \right\rangle$$

Although higher orders can be written and worked out explicitly as well, the integrals will quickly become too complicated.

generalization:

The typical structure is



The cost of a vertex is Γ

The cost of the lines between the vertices is $e^{-(\tau_2-\tau_1)h\sigma_z}$

Note in particular the periodic boundary conditions in β; ie time differences are measured modulo β.

limits:

high temperature: preferably a straight world line, few vertices

low temperature, h >> Γ : preferably a straight world line, few vertices

low temperature, Γ >> h: many vertices (spin wants to align along the x-axis!)

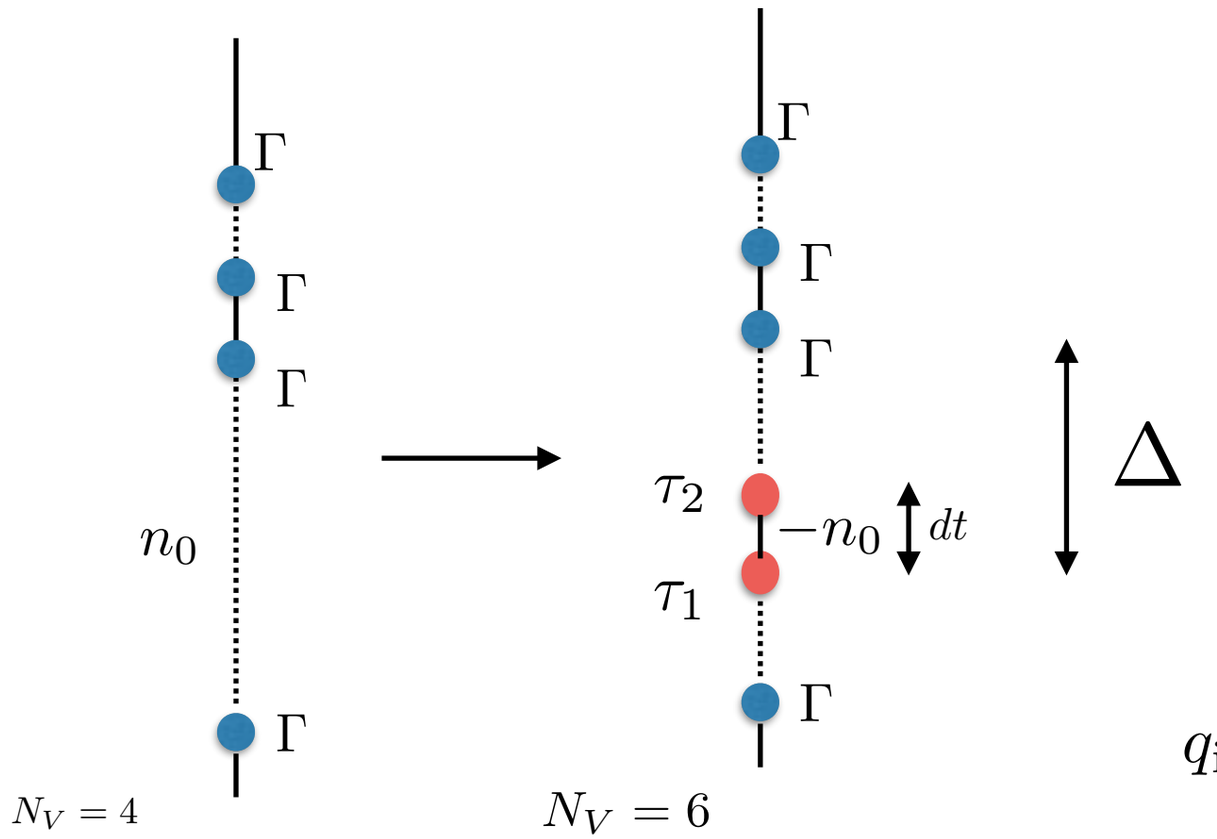
Monte Carlo sampling

our basis states from the quantization along the z-axis are accurate at high temperature and/or strong fields in the z direction. In those limits, we expect low expansion orders. When the field along the x-axis is strong our basis states do not resemble the true eigenstates of the system. We need an algorithm that can efficiently sample this regime.

The minimum set of updates that satisfy ergodicity are INSERT and REMOVE

INSERT update

1. select at random a time $\tau_1 \in [0, \beta[$
2. Find the interval Δ over which the occupation does not change. In case there are no vertices yet then Δ = β
3. select a random time interval $dt \in]0, \Delta[$
4. The time of the second vertex is $\tau_2 = (\tau_1 + dt) \bmod \beta$
5. The spin is flipped in between where between means times greater than τ_1 and less than τ_2



$$W(X) = e^{-dt \cdot h \cdot n_0}$$

$$W(Y) = \Gamma^2 e^{+dt \cdot h \cdot n_0} d\tau_1 d\tau_2$$

$$P(X \to Y) = \frac{1}{\beta \Delta} d\tau_1 d\tau_2$$

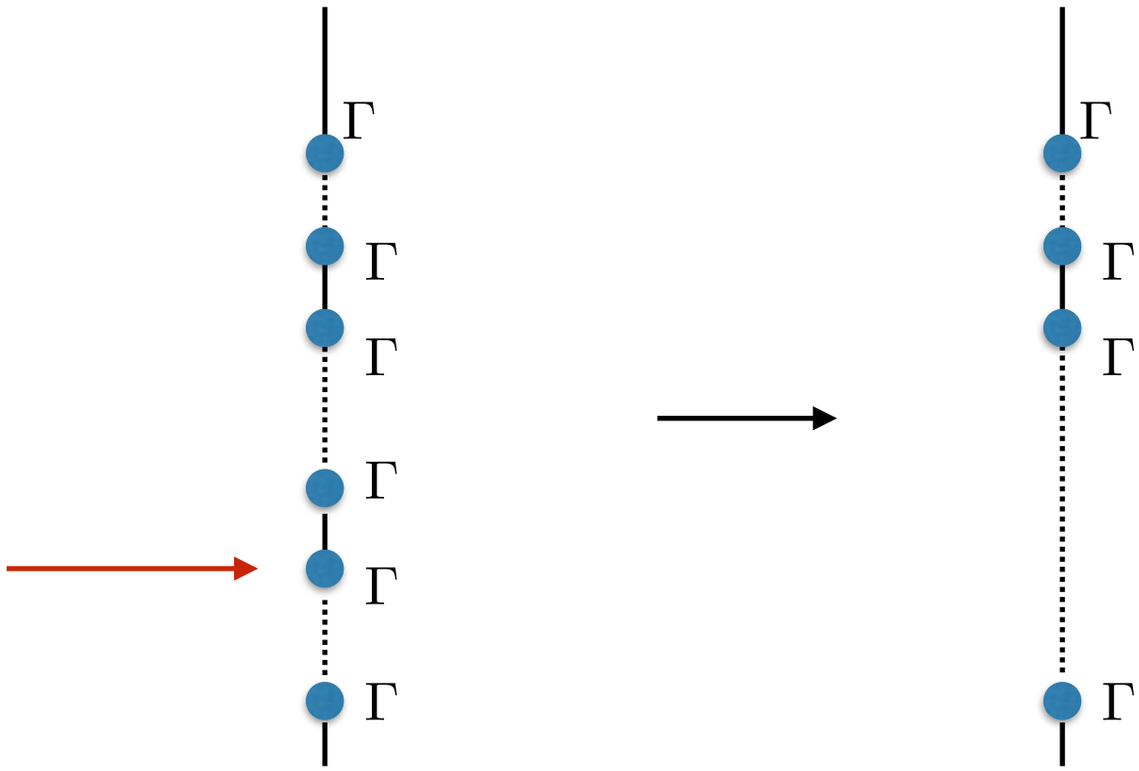
$$P(Y \to X) = \frac{1}{N_V + 2}$$

$$q_{\text{ins}} = \min \left[1, \frac{W(Y) P(Y \to X)}{W(X) P(X \to Y)} \right]$$

REMOVE update

1. select at random a vertex v1

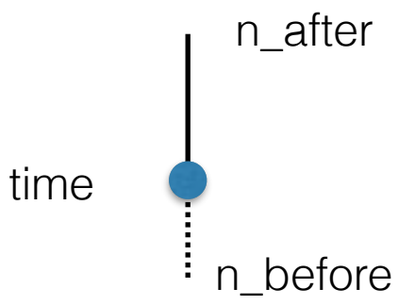
2. select deterministically the chronologically next vertex v2 (if v1 was the last vertex, then v2 is the first vertex)



$N_V = 6$ $N_V = 4$

Data structure

The central element is the vertex, specified by its time and the occupation number before (and after):




```
class vertex_type {
public:
  void time(const double t) { mtime = t;}
  void before(const int n) { n_before = n;}
  void after(const int n) { n_after = n;}
  double get_time() const { return mtime;}
  int before() const { return n_before;}
  int after() const { return n_after;}
private:
  double mtime;
  int n_before;
  int n_after;
};
```

the configuration is a chronological set of vertices. Many implementations are possible, such as the STL *vector* or the STL *list*:

class template
<list>

std::list

```
template < class T, class Alloc = allocator<T> > class list;
```

List

Lists are sequence containers that allow constant time insert and erase operations anywhere within the sequence, and iteration in both directions.

List containers are implemented as doubly-linked lists; Doubly linked lists can store each of the elements they contain in different and unrelated storage locations. The ordering is kept internally by the association to each element of a link to the element preceding it and a link to the element following it.

another possibility is to use a *binary search tree* implemented in the „map“ :

class template
<map>

std::map

```
template < class Key,                                    // map::key_type
           class T,                                      // map::mapped_type
           class Compare = less<Key>,                    // map::key_compare
           class Alloc = allocator<pair<const Key,T> >   // map::allocator_type
           > class map;
```

Map
Maps are associative containers that store elements formed by a combination of a *key value* and a *mapped value*, following a specific order.

In a `map`, the *key values* are generally used to sort and uniquely identify the elements, while the *mapped values* store the content associated to this *key*. The types of *key* and *mapped value* may differ, and are grouped together in member type `value_type`, which is a pair type combining both:

```
typedef pair<const Key, T> value_type;
```

Internally, the elements in a `map` are always sorted by its *key* following a specific *strict weak ordering* criterion indicated by its internal comparison object (of type `Compare`).

`map` containers are generally slower than unordered_map containers to access individual elements by their *key*, but they allow the direct iteration on subsets based on their order.

The mapped values in a map can be accessed directly by their corresponding key using the *bracket operator* ((operator[]).

Maps are typically implemented as *binary search trees*.

note: one can also consider a hash table (std::unordered_map in C++11), or construct a statically allocated doubly-linked lookup table (as one would do in Fortran, and which is almost certainly the best choice)

Every data structure that you use will require changes to the coding. It is one of the biggest design questions when developing a new algorithm. We will use the C++ STL list.

Estimators:

The magnetization along the z-direction can be measured in two ways:

1. One keeps track of the occupation n_0 at time τ=0 during the simulation. The measurement is then this number $n_0 = \pm 1$

2. One evaluates : $\frac{1}{\beta}\int_0^\beta \sigma_z(\tau)d\tau$ This quantity can also be updated on the track

Perhaps one naively expects the second way to be better since it involves „more information“. This is however wrong; both ways are equally good (as we will see) and there is hence no gain in using the second way

The magnetization along the x-direction can be measured as

$$\langle \sigma_x \rangle = -\frac{\langle n_v \rangle}{\beta\Gamma}$$

where n_v is the number of vertices in the configuration.
Prove this relation in an analogous way to how the energy is computed in the SSE formalism.

results:

high temperature:

```
Parameters : beta = 0.5 gamma = 0.2 h_mag = 0.2 Ntherm = 100000 Nloop = 100000 Nsweep = 100
Initial values : n0 = -1 nv = 0
Ncorr : 1 Nloop : 100000
 Magnetization(z) : -0.0993386
<sigma_z> : -0.110991 +/- 0.00868948 tau 381.744
<sigma_z> : -0.111069 +/- 0.00869026 tau 383.128
 Magnetization(x) : 0.0993386
<sigma_x> : 0.101372 +/- 0.000867264 tau 1.36222


# UPDATE STATISTICS
# row 1 : all updates
# row 2 : possible updates
# row 3 : rejected updates
# row 4 : accepted updates
# row 5 : acceptance factor (possible)
# row 6 : acceptance factor (total)
INSERT      5.00054e+06  5.00054e+06  4.9754e+06   25142    0.00502786  0.00502786
REMOVE      4.99946e+06  25142        0            25142    1           0.00502894
# Histogram : 0.994934   0.0050628    2.9e-06  0    0   0    0    0    0    0    0    0    0    0    0    0    0
```

low temperature, strong z-field

```
Parameters : beta = 10 gamma = 0.05 h_mag = 0.4 Ntherm = 100000 Nloop = 100000 Nsweep = 100
Initial values : n0 = -1 nv = 0
Ncorr : 1 Nloop : 100000
 Magnetization(z) : -0.991653
<sigma_z> : -0.991607 +/- 0.000119053 tau 3.73956
<sigma_z> : -0.991588 +/- 6.50863e-05 tau 3.68196
 Magnetization(x) : 0.123957
<sigma_x> : 0.1249 +/- 0.000578426 tau 2.8777


# UPDATE STATISTICS
# row 1 : all updates
# row 2 : possible updates
# row 3 : rejected updates
# row 4 : accepted updates
# row 5 : acceptance factor (possible)
# row 6 : acceptance factor (total)
INSERT      5.00026e+06  5.00026e+06  4.92128e+06  78981    0.0157954  0.0157954
REMOVE      4.99974e+06  154388       75407        78981    0.511575   0.015797
# Histogram : 0.969124   0.0305312    0.0003415    3.6e-06  0    0    0    0    0    0    0    0    0    0    0    0
```

low temperature, strong x-field

```
Parameters : beta = 10 gamma = 0.4 h_mag = 0.05 Ntherm = 100000 Nloop = 100000 Nsweep = 100
Initial values : n0 = -1 nv = 0
Ncorr : 1 Nloop : 100000
 Magnetization(z) : -0.123957
<sigma_z> : -0.121814 +/- 0.00102484 tau 4.83063
<sigma_z> : -0.123657 +/- 0.00037024 tau 2.30787
 Magnetization(x) : 0.991653
<sigma_x> : 0.991833 +/- 0.000407013 tau 2.77057


# UPDATE STATISTICS
# row 1 : all updates
# row 2 : possible updates
# row 3 : rejected updates
# row 4 : accepted updates
# row 5 : acceptance factor (possible)
# row 6 : acceptance factor (total)
INSERT      4.99869e+06  4.99869e+06  2.39976e+06  2.59892e+06    0.519921   0.519921
REMOVE      5.00131e+06  4.80139e+06  2.20246e+06  2.59892e+06    0.541286   0.519649
# Histogram : 0.0399968   0.296042 0.388064 0.205439 0.05861750.0104647  0.0012574  0.0001125  6.7e-06    2e-07 0
```

The results are essentially always within 1 sigma of the analytical answer

The integrated autocorrelation times are always good at low temperature. The acceptance ratio is good at low temperature and strong x-field. At high temperature the autocorrelation time is bad for the <sigma_z> but good for the <sigma_x> field. This is explained by the histogram of occupation numbers: 99% of the time we are in the n_v=0 sector

(note: generating the times of the kinks according to an exponential distribution reduces the autocorrelation time only by about 10%)

The algorithm behaves as expected and is good in the regime where it was designed to work (and where high expansion orders matter). There is no point in trying to improve it further since it works essentially fine.

Nevertheless, the problem at high temperature can be fixed by adding an update (SPIN_FLIP) which attempts to flip the spin over the full β - interval.

It suffices if this is allowed in the absence of any kinks. The acceptance factor is then

$$r = \exp\left(2\beta h n_0\right)$$

with n_0 the occupation in the 'old' configuration

At high temperature the results are then

```
Parameters : beta = 0.5 gamma = 0.2 h_mag = 0.2 Ntherm = 1000000 Nloop = 100000 Nsweep = 100
Initial values : n0 = -1 nv = 0
Ncorr : 1 Nloop : 100000
 Magnetization(z) : -0.0993386
<sigma_z> : -0.100033 +/- 0.000633487 tau 1.52681
 Magnetization(z) : -0.0993386
<sigma_z> : -0.100032 +/- 0.000633948 tau 1.53661
 Magnetization(x) : 0.0993386
<sigma_x> : 0.100248 +/- 0.000938422 tau 1.70367


# UPDATE STATISTICS
# row 1 : all updates
# row 2 : possible updates
# row 3 : rejected updates
# row 4 : accepted updates
# row 5 : acceptance factor (possible)
# row 6 : acceptance factor (total)
SPIN_FLIP    2.00092e+06  1.9908e+06   199121   1.79168e+06   0.899979   0.895425
INSERT       3.99776e+06  3.99776e+06  3.97779e+06  19967    0.00499455 0.00499455
REMOVE       4.00132e+06  19967    0   19967    1   0.00499011
# Histogram : 0.994992   0.0050044    4e-06    0   0   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

whereas at low temperature the integrated autocorrelation times remain the same

```
Parameters : beta = 10 gamma = 0.05 h_mag = 0.4 Ntherm = 1000000 Nloop = 100000 Nsweep = 100
Initial values : n0 = -1 nv = 0
Ncorr : 1 Nloop : 100000
 Magnetization(z) : -0.991653
<sigma_z> : -0.991601 +/- 0.000121103 tau 3.88379
 Magnetization(z) : -0.991653
<sigma_z> : -0.991628 +/- 6.29931e-05 tau 3.41675
 Magnetization(x) : 0.123957
<sigma_x> : 0.12477 +/- 0.000689489 tau 4.30959


# UPDATE STATISTICS
# row 1 : all updates
# row 2 : possible updates
# row 3 : rejected updates
# row 4 : accepted updates
# row 5 : acceptance factor (possible)
# row 6 : acceptance factor (total)
SPIN_FLIP    2.00076e+06  1.93886e+06  1.93759e+06  1269     0.000654508     0.000634258
INSERT       3.99796e+06  3.99796e+06  3.93491e+06  63041    0.0157683  0.0157683
REMOVE       4.00128e+06  123342   60301    63041    0.511107 0.0157552
# Histogram : 0.969140.0305303    0.0003276    2.3e-06 0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

and

```
Parameters : beta = 10 gamma = 0.4 h_mag = 0.05 Ntherm = 1000000 Nloop = 100000 Nsweep = 100
Initial values : n0 = -1 nv = 0
Ncorr : 1 Nloop : 100000
 Magnetization(z) : -0.123957
<sigma_z> : -0.121593 +/- 0.00110555 tau 5.70289
 Magnetization(z) : -0.123957
<sigma_z> : -0.123292 +/- 0.000390398 tau 2.62269
 Magnetization(x) : 0.991653
<sigma_x> : 0.991717 +/- 0.000469308 tau 3.85539


# UPDATE STATISTICS
# row 1 : all updates
# row 2 : possible updates
# row 3 : rejected updates
# row 4 : accepted updates
# row 5 : acceptance factor (possible)
# row 6 : acceptance factor (total)
SPIN_FLIP    2.00144e+06  79988    36976    43012    0.537731 0.0214905
INSERT       3.99889e+06  3.99889e+06  1.91978e+06  2.07911e+06    0.519921   0.519921
REMOVE       3.99967e+06  3.8396e+06   1.76049e+06  2.07911e+06    0.541491   0.51982
# Histogram : 0.0400099   0.295998 0.387858 0.20598  0.05845150.0103647  0.0012196  0.0001079  9.6e-06    0    0    0    0    0    0    0
```

A good rule is to keep the program as simple as possible, analyze, and only improve if and where needed

Simple extensions: polaron models with bosonic bath: see Fröhlich polaron slides

Simple extensions: spin-boson model

$$H = \frac{\Delta}{2}\sigma_x + \frac{1}{2}\sigma_z \sum_i \lambda_i (a_i + a_i^\dagger) + \sum_i \omega_i a_i^\dagger a_i$$

= a two-level system coupled to a bosonic bath with spectral function

$$J(\omega) = \pi \sum_i \lambda_i^2 \delta(\omega - \omega_i) = 2\ \pi \alpha \omega_{\rm cutoff}^{1-s} \omega^s$$

s = 1: Ohmic; s < 1 : subohmic

The bath is integrated out and gives rise to a retarded interaction between the spins:

$$S[\sigma] = -\int_0^\beta d\tau \int_0^\tau d\tau' \sigma(\tau') K_\beta(\tau - \tau') \sigma(\tau')$$

$$K_\beta(\tau) = \int_0^\infty d\omega \frac{J(\omega)}{\pi} \frac{\cosh\left(\frac{\hbar\beta\omega}{2}[1 - 2\tau/\beta]\right)}{\sinh\left(\frac{\hbar\beta\omega}{2}\right)}$$

Phys.Rev.Lett.102:030601,2009

http://arxiv.org/abs/0909.4822

http://arxiv.org/abs/1106.2654

http://arxiv.org/abs/0911.4490

At a critical coupling strength the system undergoes a quantum phase transition between a delocalized phase and a localized phase. The phase transition belongs to the universality class of the Ising model with long-range interactions

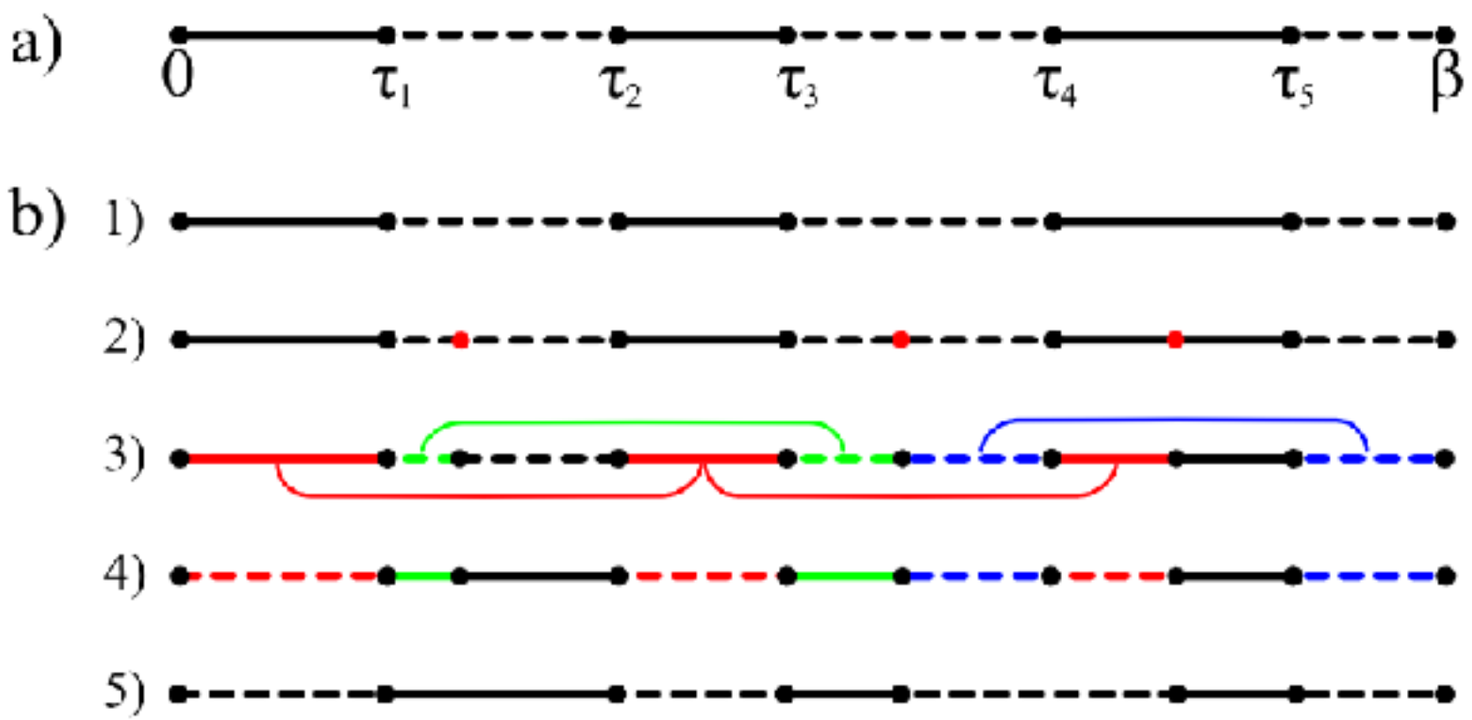
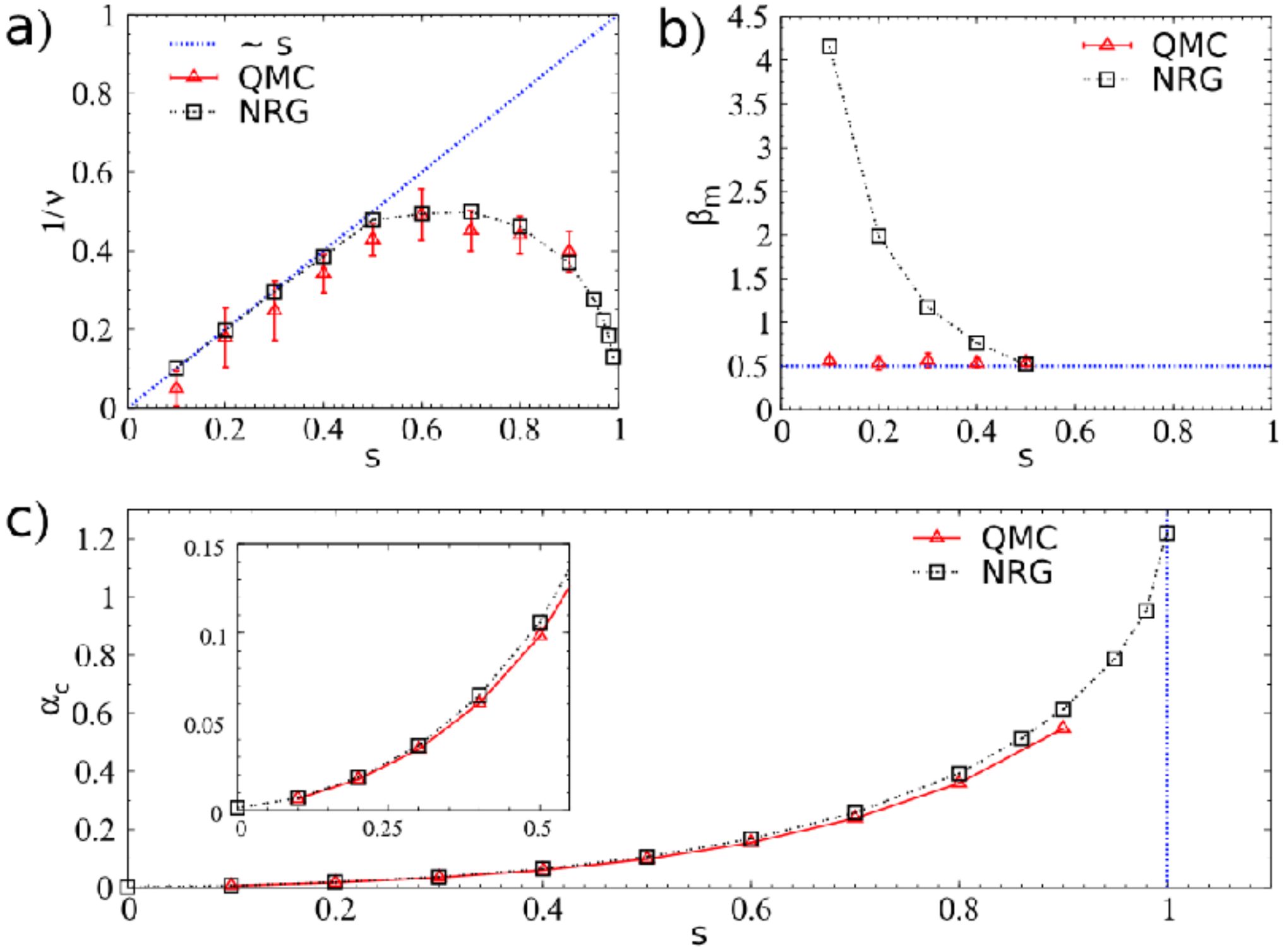


FIG. 1: (Color online) **a** Realization of an imaginary time world line of a spin-1/2 in a transverse field. **b)** Sketch of the continuous time cluster update: 1) Starting configuration. 2) Random insertion of new potential spin flips (red dots) with Poissonian statistics 3) Connection of segments with probabilities given by eq.(8). Different colors indicate the resulting clusters. 4) Each cluster is flipped with probability 1/2 (the blue one was not flipped). 5) Resulting new imaginary time world line.

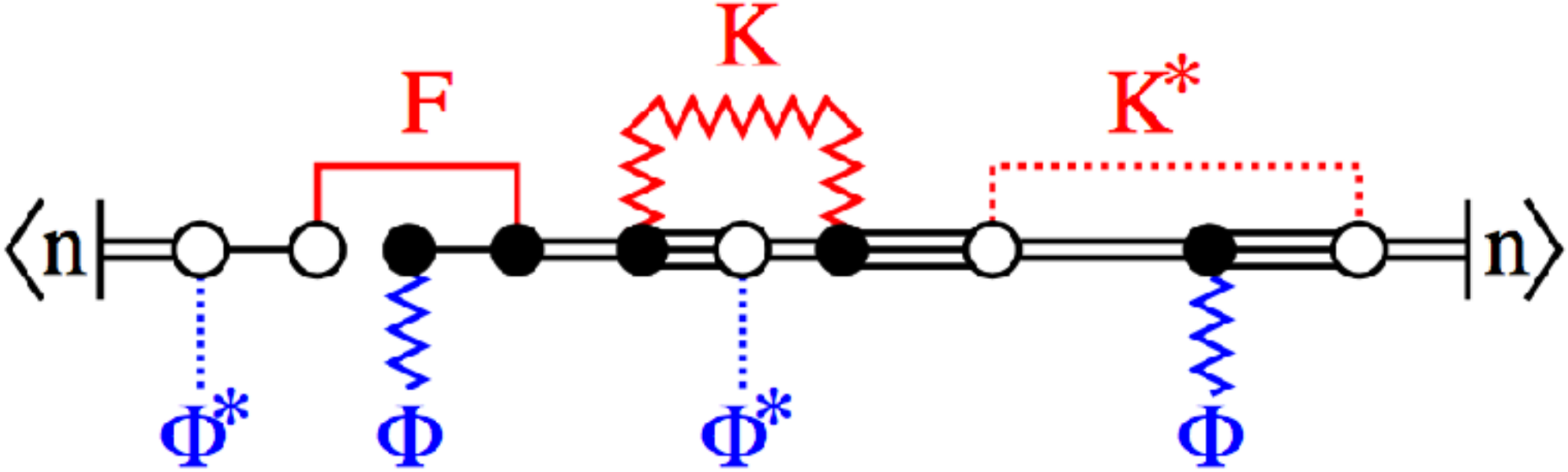
Phys.Rev.Lett.102:030601,2009



Phys.Rev.Lett.102:030601,2009

Simple extensions: impurity solver for bosonic dynamical mean-field theory

$$S_{\rm imp} = -\frac{1}{2}\int_0^\beta d\tau d\tau' \mathbf{b}^\dagger(\tau)\mathbf{\Delta}(\tau-\tau')\mathbf{b}(\tau') - \tilde{\mu}\int_0^\beta d\tau n(\tau)$$
$$+\frac{U}{2}\int_0^\beta d\tau n(\tau)[n(\tau)-1] - \kappa\mathbf{\Phi}^\dagger\int_0^\beta d\tau \mathbf{b}(\tau).$$



Phys. Rev. Lett. 105, 096402 (2010)

New J. Phys. 13, 075013 (2011)

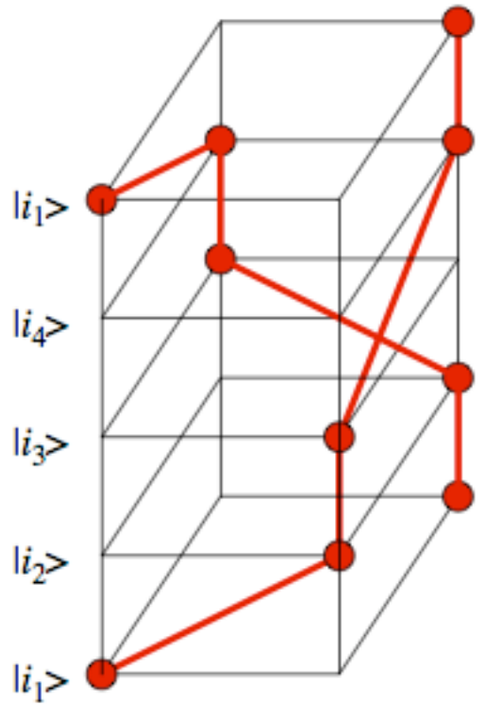
fermions and sign problem

In mapping the quantum to the classical system

$$Z = \mathrm{Tr}\exp^{-\beta H} = \sum_i p_i$$

some of the p_i may $p_i < 0$

e.g. 2 electrons might exchange places



Troyer and Wiese have shown that the sign problem is NP-hard

http://arxiv.org/abs/cond-mat/0408370; Phys.Rev.Lett. 94 (2005) 170201

consequence : exponential scaling

Implications

evaluation in case of negative weights:

$$\langle A \rangle = \frac{\sum_c A(c)p(c)}{\sum_c p(c)}$$
$$= \frac{\sum_c A(c)s(c)|p(c)| / \sum_c |p(c)|}{\sum_c s(c)|p(c)| / \sum_c |p(c)|} \equiv \frac{\langle As \rangle'}{\langle s \rangle'}.$$

with $\langle s \rangle = Z/Z'$ the average sign given by

$Z = \sum_i p_i$ 'fermionic' system

$Z' = \sum_i |p_i|$ 'bosonic' system

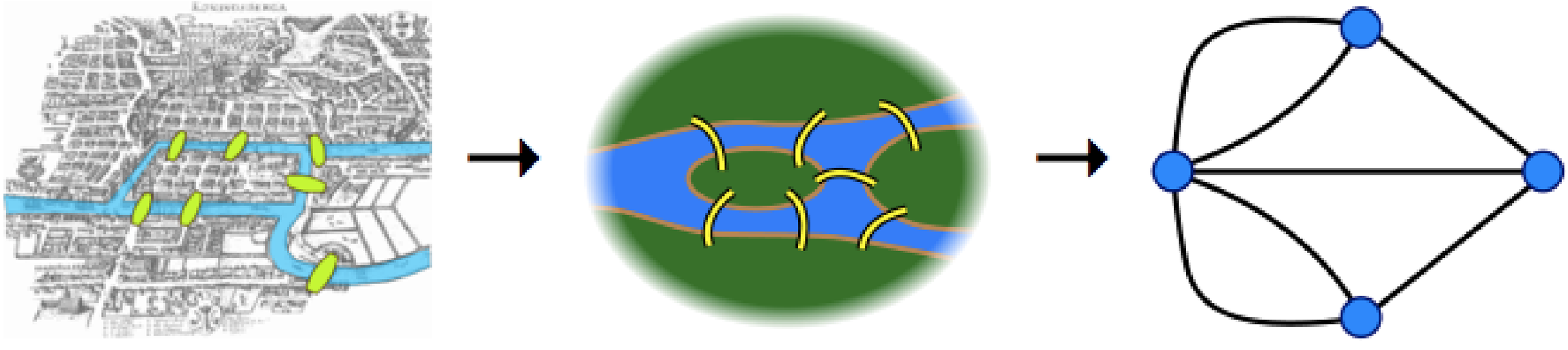
$$Z/Z' = \exp(-\beta N \Delta f)$$

The variance can become exponentially large :

$$\frac{\Delta s}{\langle s \rangle} = \frac{\sqrt{(\langle s^2 \rangle - \langle s \rangle^2)/M}}{\langle s \rangle} = \frac{\sqrt{1 - \langle s \rangle^2}}{\sqrt{M}\langle s \rangle} \sim \frac{e^{\beta N \Delta f}}{\sqrt{M}}.$$

The sign problem is basis dependent : e.g., if we know the full spectrum, all weights are positive
However, still no solution
The situation is reminiscent of NP hard problems (no proof of exponential scaling, but no solution that scales polynomially is known)
A solution to the sign-problem is a solution that does not scale exponentially (stronger than positive weights) when the bosonic problem is easy (polynomial)

Eulerian Circuit Problem

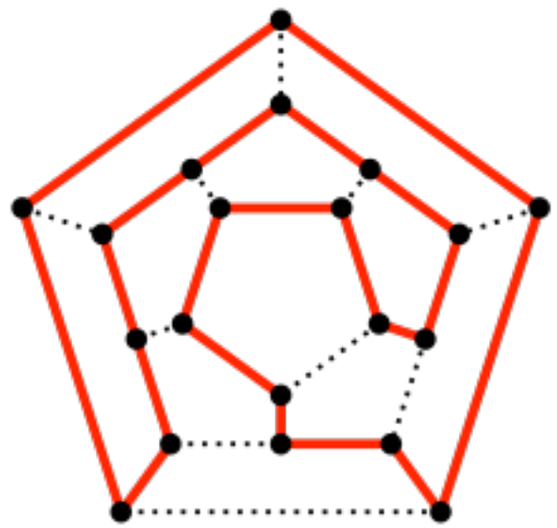


- 7 bridges of Königsberg
- is there a roundtrip that crosses each bridge exactly once?

•Euler (1735) : it exists if and only if the graph is connected and there are no nodes of odd degree at all

- can be evaluated in polynomial time; is in complexity class P

Hamiltonian cycle problem



- is there a path that crosses each *vertex* exactly once?
- expensive task by evaluating all paths
- no solution in polynomial time is known
- is NP-complete

Complexity classes

Turing machine : abstract notion of a CPU

Complexity class P :

P is defined as the set of all languages which can be decided by a deterministic polynomial-time Turing machine

Complexity class NP :

- polynomial time on a non-deterministic Turing machine : it can evaluate both branches of an if-statement, but the branches cannot merge again. It has an exponential number of CPUs but no communication between them is allowed
- solves the Hamiltonian cycle problem in polynomial time; also determines whether a spin glass has an energy lower than a predefined value E_0
- a solution to the problem can be verified in polynomial time on a polynomial Turing machine
- cannot calculate the partition function of the spin glass since the sum over the states cannot be performed

(There exist many more complexity classes)

Concepts

Polynomial reduction

- two polynomial decision problems *P* and *Q*
- $Q \leq P$ means there is a polynomial solution for *Q*, provided there is one for *P*
- many problems have been reduced to others

NP-hardness

- a problem is NP-hard if $Q \leq P$ for all *Q* in NP
- as hard as the hardest problems in NP; not necessarily in NP or even a decision problem

NP-complete

- a problem is NP-complete if it is NP-hard and if it is in NP
- most problems that are NP-hard are shown to be NP-complete

It is an open question whether *P=NP* ?

- one of the millenium challenges of the Clay Math Foundation ($1 million)

proof that sign problem is NP-hard

consider 3d frustrated Ising model (glass)

$$H = -\sum_{\langle j,k \rangle} J_{jk}\sigma_j\sigma_k \qquad\qquad J_{j,k} = 0, \pm J$$

does there exist a state with energy less than a bound E_0?

Is a NP-complete problem. F. Barahona, J. Phys. A 15, 3241 (1982).

view it as a quantum problem in basis where H is not diagonal :

$$H = -\sum_{\langle j,k \rangle} J_{j,k}\sigma_j^x\sigma_k^x \qquad\qquad J_{j,k} = 0, \pm J$$

random signs appear in off-diagonal matrix elements

bosonic model (ferromagnet, $J_{jk} > 0$) easy to solve

Hence, the sign problem causes NP-hardness