# Slides from FYS4410 Lectures

Morten Hjorth-Jensen

[1]Department of Physics and Center of Mathematics for Applications
University of Oslo, N-0316 Oslo, Norway

Spring 2007

## 18 January - 1 March

- Part I: Statistical Physics and Monte Carlo Simulations
    1. Simulations of Phase Transitions, examples with spin models such as the Ising Model and the Potts Model.
    2. Parallelization (MPI) and high-performance computing topics (OOP). Choose between F95 and/or C++. Python also possible as programming language.
    3. Algorithms for Monte Carlo Simulations, Metropolis, Wolff, Swendsen-Wang and heat bath.
    4. Histogram method and statistical analysis of data
    5. Finite size scaling and Monte Carlo renormalization group
    6. Project 1, deadline march 5

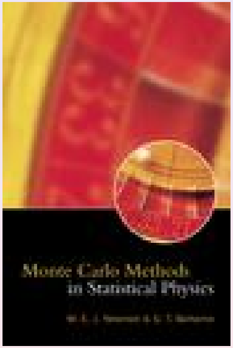
## 2 march -31 March

- Part II: Quantum Mechanical Systems
    1. Coupled cluster methods, approximation to configuration interaction methods
    2. widely used in quantum physics, atomic, molecular,solid state and nuclear physics
    3. Coupled cluster with single and doubles excitations
    4. Simulation of quantum dots (electrons in harmonic oscillator-like traps)
    5. Parallelization of sparse matrix multiplications
    6. Project 2, deadline april 16

## Lectures and ComputerLab

- Lectures: Thursday (14.15-16.00, room FV329) and friday day (14.15-16, room FV311)
- Detailed lecture notes, all programs presented and projects can be found at the homepage of the course.
- Computerlab: 16-19 thursday, room FV329
- Weekly plans and all other information are on the official webpage.
- The thursday lectures will be used to demonstrate algorithms etc, while the friday lectures will be more like classical blackboard sessions.
- For the first part, chapters 8-12 of the FYS3150/4150 lecture notes give a good starting point, together with Newman and Barkema's text, see next slide. For the FYS3150 lectures see `http://www.uio.no/studier/emner/matnat/fys/FYS3150/h06/`

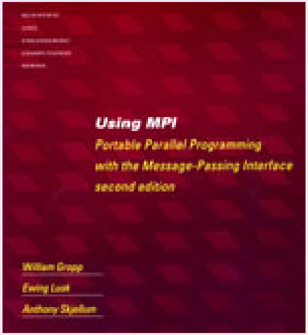
## Monte Carlo text



**Newman and Barkema**

- Monte Carlo Methods in Statistical Physics
- Chapters 1-4 and 8
- see `http://www.oup.com/uk/catalogue/?ci=9780198517979`

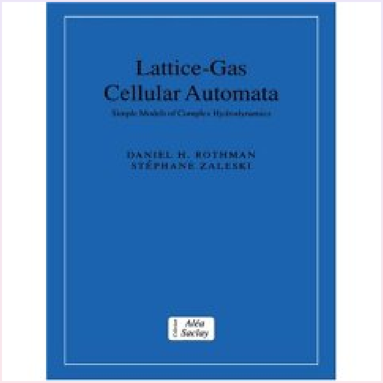
# MPI text


Using MPI
Portable Parallel Programming
with the Message-Passing Interface
second edition
William Gropp
Ewing Lusk
Anthony Skjellum

**Gropp, Lusk and Sjellum**

- Using MPI
- Chapters 1-5
- see `http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=10761`

# Lattice Gas and Cellular Automata text



## Rothman and Zaleski

- Lattice Gas and Cellular Automata
- see `http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521607605`

## Topics for Week 3, January 15-19

### Review of Statistical Physics and Parallelization

- Introduction to Message Passing Interface and parallelization.
- How to use the local cluster.
- Presentation of topics to be covered in part I
- Review of Statistical physics, with an emphasis on the canonical ensemble and the Ising model.

## Message Passing Interface (MPI) and the local cluster

### Basic info for C/C++ users

The basic instructions for using the local cluster are at `http://folk.uio.no/jonkni/gvd/workdoc/mpich2-fys/mpich2-fys.html`. Follow these instructions in detail. The file 'machines' can be pulled down from MHJ's part under the webpage of the course, see programs and MPI examples.

```
Use the example code calc_pi.c
compile and load with
mpicc -O2 -o calc_pi.x  calc_pi.c
run with 20 nodes
mpirun -np 20 ./calc_pi.x
```

For better machines, see `www.notur.no`

## Message Passing Interface (MPI) and the local cluster

### Basic info for Fortran95 users

```
Use the example code paraising.f90
compile and load with
mpif90 -O2 -o ising.exe  paraising.f90
run with 20 nodes
mpirun -np 20 ./ising.exe < input.dat
```

# Message Passing Interface (MPI) and the local cluster

## Profiling

For both Fortran and C/C++ users it is recommended that you compile with the profiling option $-pg$ in the beginning. This allows you to profile the code and figure out possible bottlenecks of CPU consumption.

```
mpicc -O2 -pg -o calc_pi.x  calc_pi.c
```

Your codes produces the profiling info in *gmon.out*. Run thereafter

```
gprof calc_pi.x  >  profileinfo
```

The file *profileinfo* contains then information about the CPU consumption of the individual functions. The strategy is then to start with the most time-consuming functions in order to see where to improve. When you are done, you should remove the $-pg$ option. Saves CPU time.

## What is Message Passing Interface (MPI)?

MPI is a library, not a language. It specifies the names, calling sequences and results of functions or subroutines to be called from C or Fortran programs, and the classes and methods that make up the MPI C++ library. The programs that users write in Fortran, C or C++ are compiled with ordinary compilers and linked with the MPI library.

MPI is a specification, not a particular implementation. MPI programs should be able to run on all possible machines and run all MPI implementetations without change.

An MPI computation is a collection of processes communicating with messages.

## MPI

MPI is a library specification for the message passing interface, proposed as a standard.

- independent of hardware;
- not a language or compiler specification;
- not a specific implementation or product.

We will only learn MPI-1 in this course.
A message passing standard for portability and ease-of-use.
Primarily for SPMD/MIMD. Designed for high performance.
Insert communication and synchronization functions where necessary.

## Demands from the HPC community

In the field of scientific computing, there is an ever-lasting wish to do larger simulations using shorter computer time.
Development of the capacity for single-processor computers can hardly keep up with the pace of scientific computing:

- processor speed
- memory size/speed

Solution: parallel computing!

## The basic ideas of parallel computing

- Pursuit of shorter computation time and larger simulation size gives rise to parallel computing.
- Multiple processors are involved to solve a global problem.
- The essence is to divide the entire computation evenly among collaborative processors.

## A rough classification of hardware models

- Conventional single-processor computers can be called SISD (single-instruction-single-data) machines.
- SIMD (single-instruction-multiple-data) machines incorporate the idea of parallel processing, which use a large number of process- ing units to execute the same instruction on different data.
- Modern parallel computers are so-called MIMD (multiple-instruction- multiple-data) machines and can execute different instruction streams in parallel on different data.

## Shared memory and distributed memory

- One way of categorizing modern parallel computers is to look at the memory configuration.
- In shared memory systems the CPUs share the same address space. Any CPU can access any data in the global memory.
- In distributed memory systems each CPU has its own memory. The CPUs are connected by some network and may exchange messages.
- A recent trend is ccNUMA (cache-coherent-non-uniform-memory- access) systems which are clusters of SMP (symmetric multi-processing) machines and have a virtual shared memory.

## Different parallel programming paradigms

- **Task parallelism** the work of a global problem can be divided into a number of independent tasks, which rarely need to synchronize. Monte Carlo simulation is one example. However this paradigm is of limited use.
- **Data parallelism** use of multiple threads (e.g. one thread per processor) to dissect loops over arrays etc. This paradigm requires a single memory address space. Communication and synchronization between processors are often hidden, thus easy to program. However, the user surrenders much control to a specialized compiler. Examples of data parallelism are compiler-based parallelization and OpenMP directives.

## Different parallel programming paradigms

- **Message-passing** all involved processors have an independent memory address space. The user is responsible for partition- ing the data/work of a global problem and distributing the subproblems to the processors. Collaboration between processors is achieved by explicit message passing, which is used for data transfer plus synchronization.
- This paradigm is the most general one where the user has full control. Better parallel efficiency is usually achieved by explicit message passing. However, message-passing programming is more difficult.

## SPMD

Although message-passing programming supports MIMD, it suffices with an SPMD (single-program-multiple-data) model, which is flexible enough for practical cases:

- Same executable for all the processors.
- Each processor works primarily with its assigned local data.
- Progression of code is allowed to differ between synchronization points.
- Possible to have a master/slave model. The standard option in Monte Carlo calculations

## Today's situation of parallel computing

- Distributed memory is the dominant hardware configuration. There is a large diversity in these machines, from MPP (massively parallel pro cessing) systems to clusters of off-the-shelf PCs, which are very cost-effective.
- Message-passing is a mature programming paradigm and widely accepted. It often provides an efficient match to the hardware. It is primarily for the distributed memory systems, but can also be used on shared memory systems.

In these lectures we consider only message-passing for writing parallel programs.

## Some useful concepts

- Speed-up

$$S(P) = \frac{T(1)}{T(P)}$$

$T(1)$ is the time for one processor while $T(P)$ is the time for $P$ processors.

- Efficiency

$$\eta(P) = \frac{S(P)}{P}$$

- Latency and bandwidth – the cost model of sending a message of length L between two processors:

$$t_C(L) = \tau + \beta L.$$

## Overhead present in parallel computing

- **Uneven load balance**: not all the processors can perform useful work at all time.
- **Overhead of synchronization.**
- **Overhead of communication**.
- Extra computation due to parallelization.

Due to the above overhead and that certain part of a sequential algorithm cannot be parallelized,

$$T(P) \geq \frac{T(1)}{P} \rightarrow S(P) \leq P$$

However, superlinear speed-up ($S(P) > P$) may sometimes occur due to for example cache effects.

## Parallelizing a sequential algorithm

- Identify the part(s) of a sequential algorithm that can be executed in parallel.
- Distribute the global work and data among $P$ processors.

## Process and processor

- We refer to process as a logical unit which executes its own code, in an MIMD style.
- The processor is a physical device on which one or several processes are executed.
- The MPI standard uses the concept process consistently throughout its documentation.
- However, we only consider situations where one processor is responsible for one process and therefore use the two terms interchangeably.

## Bindings to MPI routines

MPI is a message-passing library where all the routines have corresponding C/C++-binding

```
    MPI_Command_name
```

and Fortran-binding (routine names are in uppercase, but can also be in lower case)

```
    MPI_COMMAND_NAME
```

The discussion in these slides focuses on the C++ binding.

## Communicator

- A group of MPI processes with a name (context).
- Any process is identified by its rank. The rank is only meaningful within a particular communicator.
- By default communicator MPI_COMM_WORLD contains all the MPI processes.
- Mechanism to identify subset of processes.
- Promotes modular design of parallel libraries.

## The 6 most important MPI routines

- MPI_ Init - initiate an MPI computation
- MPI_Finalize - terminate the MPI computation and clean up
- MPI_Comm_size - how many processes participate in a given MPI communicator?
- MPI_Comm_rank - which one am I? (A number between 0 and size-1.)
- MPI_Send - send a message to a particular pro cess within an MPI communicator
- MPI_Recv - receive a message from a particular pro cess within an MPI communicator

## The first MPI C/C++ program

Let every process write ”Hello world” on the standard output.

```
#include <stdio.h>
#include <mpi.h>
int main (int nargs, char** args)
{
  int size, my_rank;
  MPI_Init (&nargs, &args);
  MPI_Comm_size (MPI_COMM_WORLD, &size);
  MPI_Comm_rank (MPI_COMM_WORLD, &my_rank);
  printf("Hello world, I’ve rank %d
          out of %d procs.\n",
          my_rank,size);
  MPI_Finalize ();
  return 0;
}
```

## The Fortran program

```
PROGRAM hello
   INCLUDE "mpif.h"
   INTEGER:: size, my_rank, ierr

   CALL  MPI_INIT(ierr)
   CALL MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
   CALL MPI_COMM_RANK(MPI_COMM_WORLD, my_rank, ierr
   WRITE(*,*)"Hello world, I’ve rank ",my_rank," ou
   CALL MPI_FINALIZE(ierr)

END PROGRAM hello
```

## Computing $\pi$ in parallel

```
#include "mpi.h"
#include <stdio.h>
int main (int nargs, char** args)
{
 int size, my_rank, i, n = 1000;
 double l_sum, g_sum, x, h;
 MPI_Init (&nargs, &args);
 MPI_Comm_size (MPI_COMM_WORLD, &size);
 MPI_Comm_rank (MPI_COMM_WORLD, &my_rank);
 if (my_rank==0 && nargs>1)
   n = atoi(args[1]); h = 1.0/n;
 MPI_Bcast (&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

## Computing $\pi$ in parallel

```
 l_sum = 0.;
for (i=my_rank; i<n; i+=size) {
  x = (i+0.5)*h;
  l_sum += 4.0/(1.0+x*x);
}
l_sum *= h;
```

Here you see what we integrate

$$\pi = \int_0^1 dx \frac{4}{1+x^2}$$

## Computing $\pi$ in parallel

```
if (my_rank==0) {
  MPI_Status status;
  g_sum = l_sum;
  for (i=1; i<size; i++) {
    MPI_Recv(&l_sum,1,MPI_DOUBLE,MPI_ANY_SOURCE
             500,MPI_COMM_WORLD,&status);
    g_sum += l_sum;
  }
  printf("result=%g\n",g_sum);
 }
 else
 MPI_Send(&l_sum,1,MPI_DOUBLE,0,500,
          MPI_COMM_WORLD);
   MPI_Finalize ();
```

## Monte Carlo integration

- Brute force integration of $f(x)$

$$E[f] = \langle f \rangle = \int_a^b fp(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)p(x_i),$$

- $$\sigma_f^2 = E[f^2] - (E[f])^2 = \langle f^2 \rangle - \langle f \rangle^2$$

## Monte Carlo integration, change of Variables

The starting point is always the uniform distribution

$$p(x)dx = \left\{ \begin{array}{cc} dx & 0 \le x \le 1 \\ 0 & else \end{array} \right.$$

with $p(x) = 1$ and satisfying

$$\int_{-\infty}^{\infty} p(x)dx = 1.$$

When we attempt a transformation to a new variable $x \rightarrow y$ we have to conserve the probability

$$p(y)dy = p(x)dx = dx,$$

Assume that $p(y)$ is a PDF different from the uniform PDF $p(x) = 1$ with $x \in [0, 1]$. If we integrate the last expression we arrive at

$$x(y) = \int_0^y p(y')dy',$$

which is nothing but the cumulative distribution of $p(y)$, i.e.,

$$x(y) = P(y) = \int_0^y p(y')dy'.$$

## Monte Carlo integration

Example: Suppose we have the general uniform distribution

$$p(y)dy = \begin{cases} \frac{dy}{b-a} & a \leq y \leq b \\ 0 & else \end{cases}$$

If we wish to relate this distribution to the one in the interval $x \in [0,1]$ we have

$$p(y)dy = \frac{dy}{b-a} = dx,$$

and integrating we obtain the cumulative function

$$x(y) = \int_a^y \frac{dy'}{b-a},$$

yielding

$$y = a + (b-a)x,$$

a well-known result!

## Importance Sampling

Let us assume that $p(y)$ is a PDF whose behavior resembles that of a function $F$ defined in a certain interval $[a, b]$. The normalization condition is

$$\int_a^b p(y)dy = 1.$$

We can rewrite our integral as

$$I = \int_a^b F(y)dy = \int_a^b p(y)\frac{F(y)}{p(y)}dy.$$

Since random numbers are generated for the uniform distribution $p(x)$ with $x \in [0, 1]$, we need to perform a change of variables $x \rightarrow y$ through

$$x(y) = \int_a^y p(y')dy',$$

where we used

$$p(x)dx = dx = p(y)dy.$$

If we can invert $x(y)$, we find $y(x)$ as well.

## Monte Carlo integration

$$I = \int_a^b p(y) \frac{F(y)}{p(y)} dy = \int_a^b \frac{F(y(x))}{p(y(x))} dx,$$

meaning that a Monte Carlo evaluation of the above integral gives

$$\int_a^b \frac{F(y(x))}{p(y(x))} dx = \frac{1}{N} \sum_{i=1}^{N} \frac{F(y(x_i))}{p(y(x_i))}.$$

The advantage of such a change of variables in case $p(y)$ follows closely $F$ is that the integrand becomes smooth and we can sample over relevant values for the integrand. It is however not trivial to find such a function $p$. The conditions on $p$ which allow us to perform these transformations are

1. $p$ is normalizable and positive definite,
2. it is analytically integrable and
3. the integral is invertible, allowing us thereby to express a new variable in terms of the old one.

# Monte Carlo integration

The variance is now with the definition

$$\tilde{F} = \frac{F(y(x))}{p(y(x))},$$

given by

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\tilde{F}\right)^2 - \left(\frac{1}{N}\sum_{i=1}^{N}\tilde{F}\right)^2.$$

If the relation between the two functions $F/p$ results in an almost constant, then we can get a variance close to zero, or zero!!

## Monte Carlo integration

The algorithm for this procedure is

- Use the uniform distribution to find the random variable $y$ in the interval [0,1]. $p(x)$ is a user provided PDF.
- Evaluate thereafter

$$I = \int_a^b F(x)dx = \int_a^b p(x)\frac{F(x)}{p(x)}dx,$$

by rewriting

$$\int_a^b p(x)\frac{F(x)}{p(x)}dx = \int_a^b \frac{F(x(y))}{p(x(y))}dy,$$

since

$$\frac{dy}{dx} = p(x).$$

- Perform then a Monte Carlo sampling for

$$\int_a^b \frac{F(x(y))}{p(x(y))}dy \approx \frac{1}{N}\sum_{i=1}^{N}\frac{F(x(y_i))}{p(x(y_i))},$$

with $y_i \in [0,1]$,

- and evaluate the variance.

## Exercises

- Run the parallel program to obtain $\pi$, calc_pi.c
- Rewrite the serial program multidim.cpp on the webpage at `http://www.uio.no/studier/emner/matnat/fys/FYS4410/v07` as a parallel program.
  This programs computes the integral

  $$\int_{-\infty}^{\infty} \mathbf{dxdy} g(\mathbf{x},\mathbf{y}),$$

  where

  $$g(\mathbf{x},\mathbf{y}) = \exp\left(-\mathbf{x}^2 - \mathbf{y}^2 - (\mathbf{x}-\mathbf{y})^2/2\right),$$

  with $d = 6$. See chapter 8 of the FYS3150 lectures at `http://www.uio.no/studier/emner/matnat/fys/FYS3150/h06`

## Topics for Week 4, January 22-26

### Statistical Physics and Parallelization

- Review of Statistical physics, with an emphasis on the canonical ensemble, Landau's mean field and the Ising model.
- Simple parallelization of the Ising model, discussion of the codes in both C/C++ and Fortran95. Further discussion of MPI.
- Discussion of correlation functions

## Most Common Ensembles in Statistical Physics

| | Microcanonical | Canonical | Grand can. | Pressure can. |
|---|---|---|---|---|
| Exchange of heat with the environment | no | yes | yes | yes |
| Exchange of particles with the environemt | no | no | yes | no |
| Thermodynamical parameters | $V, \mathcal{M}, \mathcal{D}$ <br> $E$ <br> $N$ | $V, \mathcal{M}, \mathcal{D}$ <br> $T$ <br> $N$ | $V, \mathcal{M}, \mathcal{D}$ <br> $T$ <br> $\mu$ | $P, \mathcal{H}, \mathcal{E}$ <br> $T$ <br> $N$ |
| Potential | Entropy | Helmholtz | $PV$ | Gibbs |
| Energy | Internal | Internal | Internal | Enthalpy |

# Microcanonical Ensemble

Entropy

$$S = k_B ln\Omega \tag{1}$$

$$dS = \frac{1}{T}dE + \frac{p}{T}dV - \frac{\mu}{T}dN \tag{2}$$

Temperature

$$\frac{1}{k_B T} = \left(\frac{\partial ln\Omega}{\partial E}\right)_{N,V} \tag{3}$$

Pressure

$$\frac{p}{k_B T} = \left(\frac{\partial ln\Omega}{\partial V}\right)_{N,E} \tag{4}$$

Chemical potential

$$\frac{\mu}{k_B T} = -\left(\frac{\partial ln\Omega}{\partial N}\right)_{V,E} \tag{5}$$

## Canonical Ensemble

Helmholtz Free Energy

$$F = -k_B T \ln Z \tag{6}$$

$$dF = -SdT - pdV + \mu dN \tag{7}$$

Entropy

$$S = k_B \ln Z + k_B T \left(\frac{\partial \ln Z}{\partial T}\right)_{N,V} \tag{8}$$

Pressure

$$p = k_B T \left(\frac{\partial \ln Z}{\partial V}\right)_{N,T} \tag{9}$$

Chemical Potential

$$\mu = -k_B T \left(\frac{\partial \ln Z}{\partial N}\right)_{V,T} \tag{10}$$

Energy (internal only)

$$E = k_B T^2 \left(\frac{\partial \ln Z}{\partial T}\right)_{V,N} \tag{11}$$

## Grand Canonical Ensemble

Potential

$$pV = k_BTln\Xi \tag{12}$$

$$d(pV) = SdT + Nd\mu + pdV \tag{13}$$

Entropy

$$S = k_Bln\Xi + k_BT\left(\frac{\partial ln\Xi}{\partial T}\right)_{V,\mu} \tag{14}$$

Particles

$$N = k_BT\left(\frac{\partial ln\Xi}{\partial \mu}\right)_{V,T} \tag{15}$$

Pressure

$$p = k_BT\left(\frac{\partial ln\Xi}{\partial V}\right)_{\mu,T} \tag{16}$$

## Pressure Canonical Ensemble

Gibbs Free Energy

$$G = -k_B T ln\Delta \tag{17}$$

$$dG = -SdT + Vdp + \mu dN \tag{18}$$

Entropy

$$S = k_B ln\Delta + k_B T \left(\frac{\partial ln\Delta}{\partial T}\right)_{p,N} \tag{19}$$

Volume

$$V = -k_B T \left(\frac{\partial ln\Delta}{\partial p}\right)_{N,T} \tag{20}$$

Chemical potential

$$\mu = -k_B T \left(\frac{\partial ln\Delta}{\partial N}\right)_{p,T} \tag{21}$$

## Expectation Values

At a given temperature we have the probability distribution

$$P_i(\beta) = \frac{e^{-\beta E_i}}{Z} \tag{22}$$

with $\beta = 1/kT$ being the inverse temperature, $k$ the Boltzmann constant, $E_i$ is the energy of a state $i$ while $Z$ is the partition function for the canonical ensemble defined as

$$Z = \sum_{i=1}^{M} e^{-\beta E_i}, \tag{23}$$

where the sum extends over all states $M$. $P_i$ expresses the probability of finding the system in a given configuration $i$.

## Expectation Values

For a system described by the canonical ensemble, the energy is an expectation value since we allow energy to be exchanged with the surroundings (a heat bath with temperature $T$). This expectation value, the mean energy, can be calculated using the probability distribution $P_i$ as

$$\langle E\rangle = \sum_{i=1}^{M} E_i P_i(\beta) = \frac{1}{Z}\sum_{i=1}^{M} E_i e^{-\beta E_i}, \tag{24}$$

with a corresponding variance defined as

$$\sigma_E^2 = \langle E^2\rangle - \langle E\rangle^2 = \frac{1}{Z}\sum_{i=1}^{M} E_i^2 e^{-\beta E_i} - \left(\frac{1}{Z}\sum_{i=1}^{M} E_i e^{-\beta E_i}\right)^2. \tag{25}$$

If we divide the latter quantity with $kT^2$ we obtain the specific heat at constant volume

$$C_V = \frac{1}{kT^2}\left(\langle E^2\rangle - \langle E\rangle^2\right). \tag{26}$$

## Expectation Values

We can also write

$$\langle E \rangle = -\frac{\partial lnZ}{\partial \beta}. \tag{27}$$

The specific heat is

$$C_V = \frac{1}{kT^2}\frac{\partial^2 lnZ}{\partial \beta^2} \tag{28}$$

These expressions link a physical quantity (in thermodynamics) with the microphysics given by the partition function. Statistical physics is the field where one relates microscopic quantities to observables at finite temperature.

## Expectation Values

$$\langle \mathcal{M} \rangle = \sum_i^M \mathcal{M}_i P_i(\beta) = \frac{1}{Z}\sum_i^M \mathcal{M}_i e^{-\beta E_i}, \qquad (29)$$

and the corresponding variance

$$\sigma_{\mathcal{M}}^2 = \langle \mathcal{M}^2 \rangle - \langle \mathcal{M} \rangle^2 = \frac{1}{Z}\sum_{i=1}^M \mathcal{M}_i^2 e^{-\beta E_i} - \left(\frac{1}{Z}\sum_{i=1}^M \mathcal{M}_i e^{-\beta E_i}\right)^2. \qquad (30)$$

This quantity defines also the susceptibility $\chi$

$$\chi = \frac{1}{kT}\left(\langle \mathcal{M}^2 \rangle - \langle \mathcal{M} \rangle^2\right). \qquad (31)$$

## Phase Transitions

NOTE: Helmholtz free energy and canonical ensemble

$$F = \langle E\rangle - TS = -kTlnZ$$

meaning $lnZ = -F/kT = -F\beta$ and

$$\langle E\rangle = -\frac{\partial lnZ}{\partial\beta} = \frac{\partial(\beta F)}{\partial\beta}.$$

and

$$C_V = -\frac{1}{kT^2}\frac{\partial^2(\beta F)}{\partial\beta^2}.$$

We can relate observables to various derivatives of the partition function and the free energy. When a given derivative of the free energy or the partition function is discontinuous or diverges (logarithmic divergence for the heat capacity from the Ising model) we talk of a phase transition of order of the derivative.

## Phase Transitions

- An important quantity is the correlation length ($\xi$, to be discussed during friday's lecture). The correlation length defines the length scale at which the overall properties of a material start to differ from its bulk properties. It is the distance over which the fluctuations of the microscopic degrees of freedom (for example the position of atoms) are significantly correlated with each other. Usually it is of the order of few interatomic spacings for a solid.
- The correlation length $\xi$ depends however on external conditions such as pressure and temperature.
- A phase transition is marked by abrupt macroscopic changes as external parameters are changed, such as an increase of temperature.
- The point where a phase transition takes place is called a critical point.

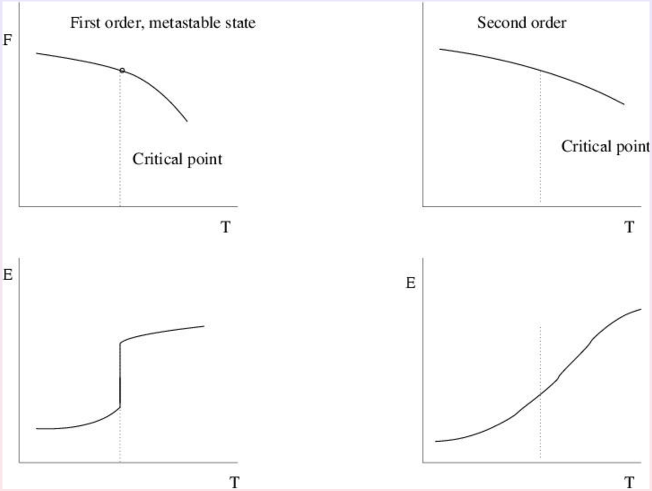
## Two Scenarios for Phase Transitions

1. First order/discontinuous phase transitions: Two or more states on either side of the critical point also coexist exactly at the critical point. As we pass through the critical point we observe a discontinuous behavior of thermodynamical functions, see figure in forthcoming slides. The correlation length is mormally finite at the critical point. Phenomena such as hysteris occur, viz. there is a continuation of state below the critical point into one above the critical point. This continuation is metastable so that the system may take a macroscopically long time to readjust. Classical example, melting of ice.
2. Second order or continuous transitions: The correlation length diverges at the critical point, fluctuations are correlated over all distance scales, which forces the system to be in a unique critical phase. The two phases on either side of the critical point become identical. Smooth behavior of first derivatives of the partition function, while second derivatives diverge. Strong correlations make a perturbative treatment impossible. Renormalization group theory.

## Examples of Phase Transitions

| System | Transition | Order Parameter |
|---|---|---|
| Liquid-gas | Condensation/evaporation | Density difference $\Delta\rho = \rho_{liquid} - \rho_{gas}$ |
| Binary liquid | mixture/Unmixing | Composition difference |
| Quantum liquid | Normal fluid/superfluid | $< \phi >$, $\psi$ = wavefunction |
| Liquid-solid | Melting/crystallisation | Reciprocal lattice vector |
| Magnetic solid | Ferromagnetic | Spontaneous magnetisation $M$ |
|  | Antiferromagnetic | Sublattice magnetisation $M$ |
| Dielectric solid | Ferroelectric | Polarization $P$ |
|  | Antiferroelectric | Sublattice polarisation $P$ |

# Examples of Phase Transitions



## Ising and Potts Model

The model we will employ first in our studies of phase transitions at finite temperature for magnetic systems is the so-called Ising model. In its simplest form the energy is expressed as

$$E = -J \sum_{<kl>}^{N} s_k s_l - B \sum_{k}^{N} s_k, \tag{32}$$

with $s_k = \pm 1$, $N$ is the total number of spins, $J$ is a coupling constant expressing the strength of the interaction between neighboring spins and $B$ is an external magnetic field interacting with the magnetic moment set up by the spins. The symbol $< kl >$ indicates that we sum over nearest neighbors only.

## Ising Model

Notice that for $J > 0$ it is energetically favorable for neighboring spins to be aligned. This feature leads to, at low enough temperatures, to a cooperative phenomenon called spontaneous magnetization. That is, through interactions between nearest neighbors, a given magnetic moment can influence the alignment of spins that are separated from the given spin by a macroscopic distance. These long range correlations between spins are associated with a long-range order in which the lattice has a net magnetization in the absence of a magnetic field. This phase is normally called the ferromagnetic phase. With $J < 0$, we have a so-called antiferromagnetic case. At a critical temperature we have a phase transition to a disordered phase, a so-called paramagnetic phase.

# Potts Model

Energy given by

$$E = -J \sum_{<kl>}^{N} \delta_{s_l, s_k},$$

where the spin $s_k$ at lattice position $k$ can take the values $1, 2, \ldots, q$. $N$ is the total number of spins. For $q = 2$ the Potts model corresponds to the Ising model, we can rewrite the last equation as

$$E = -\frac{J}{2} \sum_{<kl>}^{N} 2(\delta_{s_l, s_k} - \frac{1}{2}) - \sum_{<kl>}^{N} \frac{J}{2}.$$

Now, $2(\delta_{s_l, s_k} - \frac{1}{2})$ is +1 when $s_l = s_k$ and $-1$ when they are different. Equivalent except the last term which is a constant and that $J \rightarrow J/2$.

## Analytic Results: one-dimensional Ising model

For the one-dimensional Ising model we can compute rather easily the exact partition function for a system of $N$ spins. Let us consider first the case with free ends. The energy reads

$$E = -J \sum_{j=1}^{N-1} s_j s_{j+1}.$$

The partition function for $N$ spins is given by

$$Z_N = \sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} \exp\left(\beta J \sum_{j=1}^{N-1} s_j s_{j+1}\right), \tag{33}$$

and since the last spin occurs only once in the last sum in the exponential, we can single out the last spin as follows

$$\sum_{s_N=\pm 1} \exp\left(\beta J s_{N-1} s_N\right) = 2cosh(\beta J). \tag{34}$$

The partition function consists then of a part from the last spin and one from the remaining spins resulting in

$$Z_N = Z_{N-1} 2cosh(\beta J). \tag{35}$$

## Analytic Results: one-dimensional Ising model

We can repeat this process and obtain

$$Z_N = (2cosh(\beta J))^{N-2} Z_2, \tag{36}$$

with $Z_2$ given by

$$Z_2 = \sum_{s_1=\pm 1} \sum_{s_2=\pm 1} \exp(\beta J s_1 s_2) = 4cosh(\beta J), \tag{37}$$

resulting in

$$Z_N = 2(2cosh(\beta J))^{N-1}. \tag{38}$$

In the thermodynamical limit where we let $N \to \infty$, the way we treat the ends does not matter. However, since our computations will always be carried out with a limited value of $N$, we need to consider other boundary conditions as well. Here we limit the attention to periodic boundary conditions.

# Analytic Results: one-dimensional Ising model

We can then calculate the mean energy with free ends from the above formula for the partition function using

$$\langle E \rangle = -\frac{\partial lnZ}{\partial \beta} = -(N-1)Jtanh(\beta J). \tag{39}$$

Helmholtz's free energy is given by

$$F = -k_B T lnZ_N = -Nk_B T ln\left(2cosh(\beta J)\right). \tag{40}$$

The specific heat in one-dimension with free ends is

$$C_V = \frac{1}{kT^2}\frac{\partial^2}{\partial\beta^2} lnZ_N = (N-1)k\left(\frac{\beta J}{cosh(\beta J)}\right)^2. \tag{41}$$

Note well that this expression for the specific heat from the one-dimensional Ising model does not diverge, thus we do not have a second order phase transition.

## Analytic Results: one-dimensional Ising model

If we use periodic boundary conditions, the partition function is given by

$$Z_N = \sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} \exp\left(\beta J \sum_{j=1}^{N} s_j s_{j+1}\right), \tag{42}$$

where the sum in the exponential runs from 1 to $N$ since the energy is defined as

$$E = -J \sum_{j=1}^{N} s_j s_{j+1}.$$

We can then rewrite the partition function as

$$Z_N = \sum_{\{s_i=\pm 1\}} \prod_{i=1}^{N} \exp\left(\beta J s_j s_{j+1}\right), \tag{43}$$

where the first sum is meant to represent all lattice sites. Introducing the matrix $\hat{\mathbf{T}}$ (the so-called transfer matrix)

$$\hat{\mathbf{T}} = \left( \begin{array}{cc} e^{\beta J} & e^{-\beta J} \\ e^{-\beta J} & e^{\beta J} \end{array} \right), \tag{44}$$

## Analytic Results: one-dimensional Ising model

$$Z_N = \sum_{\{s_i=\pm 1\}} \hat{\mathbf{T}}_{s_1 s_2}\hat{\mathbf{T}}_{s_2 s_3}\dots\hat{\mathbf{T}}_{s_N s_1} = Tr\hat{\mathbf{T}}^N. \tag{45}$$

The $2\times 2$ matrix $\hat{\mathbf{T}}$ is easily diagonalized with eigenvalues $\lambda_1 = 2cosh(\beta J)$ and $\lambda_2 = 2sinh(\beta J)$. Similarly, the matrix $\hat{\mathbf{T}}^N$ has eigenvalues $\lambda_1^N$ and $\lambda_2^N$ and the trace of $\hat{\mathbf{T}}^N$ is just the sum over eigenvalues resulting in a partition function

$$Z_N = \lambda_1^N + \lambda_2^N = 2^N\left([cosh(\beta J)]^N + [sinh(\beta J)]^N\right). \tag{46}$$

Helmholtz's free energy is in this case

$$F = -k_BTln(\lambda_1^N+\lambda_2^N) = -k_BT\left\{Nln(\lambda_1) + ln\left(1+(\frac{\lambda_2}{\lambda_1})^N\right)\right\} \tag{47}$$

which in the limit $N\to\infty$ results in $F = -k_BTNln(\lambda_1)$

# Analytic Results: one-dimensional Ising model

Hitherto we have limited ourselves to studies of systems with zero external magnetic field, viz $\mathcal{H} = \prime$. We will mostly study systems which exhibit a spontaneous magnitization. It is however instructive to extend the one-dimensional Ising model to $\mathcal{H} \neq \prime$, yielding a partition function (with periodic boundary conditions)

$$Z_N = \sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} \exp\left(\beta \sum_{j=1}^{N} (Js_j s_{j+1} + \frac{\mathcal{H}}{2}(s_i + s_{j+1})\right), \qquad (48)$$

which yields a new transfer matrix with matrix elements $t_{11} = e^{\beta(J+\mathcal{H})}$, $t_{1-1} = e^{-\beta J}$, $t_{-11} = e^{\beta J}$ and $t_{-1-1} = e^{\beta(J-\mathcal{H})}$ with eigenvalues

$$\lambda_1 = e^{\beta J} cosh(\beta J) + \left(e^{2\beta J} sinh^2(\beta \mathcal{H}) + ]\S_{\surd}(-\beta \in \mathcal{J})\right)^{1/2}, \qquad (49)$$

and

$$\lambda_2 = e^{\beta J} cosh(\beta J) - \left(e^{2\beta J} sinh^2(\beta \mathcal{H}) + ]\S_{\surd}(-\beta \in \mathcal{J})\right)^{1/2}. \qquad (50)$$

## Analytic Results: one-dimensional Ising model

It is now useful to compute the expectation value of the magnetisation per spin

$$\langle \mathcal{M}/N \rangle = \frac{1}{NZ}\sum_{i}^{M} \mathcal{M}_i e^{-\beta E_i} = -\frac{1}{N}\frac{\partial F}{\partial \mathcal{H}}, \tag{51}$$

resulting in

$$\langle \mathcal{M}/N \rangle = \frac{sinh(\beta\mathcal{H})}{\left(sinh^2(\beta\mathcal{H}) + \exp\left(-\beta\in\mathcal{J}\right)\right)^{1/2}}. \tag{52}$$

We see that for $\mathcal{H} = \prime$ the magnetisation is zero. This means that for a one-dimensional Ising model we cannot have a spontaneous magnetization. And there is no second order phase transition as well.

## Mean Field Theory and the Ising Model

In studies of phase transitions we are interested in minimizing the free energy by varying the average magnetisation, which is the order parameter (disappears at $T_C$). In mean field theory the local magnetisation is a treated as a constant, all effects from fluctuations are neglected. A way to achieve this is to rewrite by adding and subtracting the mean magnetisation $\langle s\rangle$

$$s_is_j = (s_i - \langle s\rangle + \langle s\rangle)(s_i - \langle s\rangle + \langle s\rangle) \approx \langle s\rangle^2 + \langle s\rangle(s_i - \langle s\rangle) + \langle s\rangle(s_j - \langle s\rangle), \quad (53)$$

where we have ignored terms of the order $(s_i - \langle s\rangle)(s_i - \langle s\rangle)$, which leads to correlations between neighbouring spins. In mean field theory we ignore correlations.

## Mean Field Theory and the Ising Model

This means that we can rewrite the Hamiltonian

$$E = -J \sum_{<ij>}^{N} s_k s_l - B \sum_{i}^{N} s_i, \tag{54}$$

as

$$E = -J \sum_{<ij>} \langle s \rangle^2 + \langle s \rangle (s_i - \langle s \rangle) + \langle s \rangle (s_j - \langle s \rangle) - B \sum_i s_i, \tag{55}$$

resulting in

$$E = -(B + zJ\langle s \rangle) \sum_i s_i + zJ\langle s \rangle^2, \tag{56}$$

with $z$ the nuber of nearest neighbours for a given site $i$. We can define an effective field which all spins see, namely

$$B_{\mathrm{eff}} = (B + zJ\langle s \rangle). \tag{57}$$

## Mean Field Theory and the Ising Model

How do we get $\langle s \rangle$)?
Here we use the canonical ensemble. The partition function reads in this case

$$Z = e^{-NzJ\langle s \rangle^2/kT}\left(2cosh(B_{\mathrm{eff}}/kT)\right)^N, \tag{58}$$

with a free energy

$$F = -kTlnZ = -NkTln(2) + NzJ\langle s \rangle^2 - NkTln\left(cosh(B_{\mathrm{eff}}/kT)\right) \tag{59}$$

and minimizing $F$ wrt $\langle s \rangle$ we arrive at

$$\langle s \rangle = tanh(2cosh\left(B_{\mathrm{eff}}/kT\right)). \tag{60}$$

## Connecting to Landau Theory

Close to the phase transition we expect $\langle s\rangle$ to become small and eventually vanish. We can then expand $F$ in powers of $\langle s\rangle$ as

$$F = -NkTln(2) + NzJ\langle s\rangle^2 - NkT - BN\langle s\rangle + NkT\left(\frac{1}{2}\langle s\rangle^2 + \frac{1}{12}\langle s\rangle^4 + \dots\right), \quad (61)$$

and using $\langle M\rangle = N\langle s\rangle$ we can rewrite as

$$F = F_0 - B\langle M\rangle + \frac{1}{2}a\langle M\rangle^2 + \frac{1}{4}b\langle M\rangle^4 + \dots \quad (62)$$

## Connecting to Landau Theory

Let $\langle M \rangle = m$ and

$$F = F_0 + \frac{1}{2}am^2 + \frac{1}{4}bm^4 + \frac{1}{6}cm^6 \tag{63}$$

$F$ has a minimum at equilibrium $F'(m) = 0$ and $F''(m) > 0$

$$F'(m) = 0 = m(a + bm^2 + cm^4),$$

and if we assume that $m$ is real we have two solutions

$$m = 0,$$

or

$$m^2 = \frac{b}{2c}\left(-1 \pm \sqrt{1 - 4ac/b^2}\right)$$

## Second Order Phase Transition

Can describe both first and second-order phase transitions. Here we consider the second case. Assume $b > 0$ and $a \ll 1$ small since we want to study a perturbation around $m = 0$. We reach the critical point when $a = 0$.

$$m^2 = \frac{b}{2c}\left((-1 \pm \sqrt{1 - 4ac/b^2}\right) \approx -a/b$$

Assume that

$$a(T) = \alpha(T - T_C),$$

with $\alpha > 0$ and $T_C$ the critical temperature where the magnetization vanishes. If $a$ is negative we have two solutions

$$m = \pm\sqrt{-a/b} = \pm\sqrt{\frac{\alpha(T_C - T)}{b}}$$

$m$ evolves continuously to the critical temperature where $F = 0$ for $T \leq T_C$ (see separate graph).

## Entropy and Specific Heat

We can now compute the entropy

$$S = -\left(\frac{\partial F}{\partial T}\right)$$

For $T \geq T_C$ we have $m = 0$ and

$$S = -\left(\frac{\partial F_0}{\partial T}\right)$$

and for $T \leq T_C$

$$S = -\left(\frac{\partial F_0}{\partial T}\right) - \alpha^2 (T_C - T)/2b,$$

and we see that there is a smooth crossover at $T_C$.

# Entropy and Specific Heat

We can now compute the specific heat

$$C_V = T\left(\frac{\partial S}{\partial T}\right)$$

and $T_C$ we get a discontinuity of

$$\Delta C_V = -\alpha^2/2b,$$

signalling a second-order phase transition. Landau theory gives irrespective of dimension critical exponents

$$m \sim (T_C - T)^{\beta},$$

and

$$C_V \sim (T_C - T)^{\alpha},$$

with $\beta = 1/2$ and $\alpha = 1$. It predicts a phase transition for one dimension as well. For the Ising model there is no phase transition for $d = 1$. In two dimensions we have $\beta = 1/8$ and $\alpha = 0$.

## Analytic Results: two-dimensional Ising model

The analytic expression for the Ising model in two dimensions was obtained in 1944 by the Norwegian chemist Lars Onsager (Nobel prize in chemistry). The exact partition function for $N$ spins in two dimensions and with zero magnetic field $\mathcal{H}$ is given by

$$Z_N = \left[2cosh(\beta J)e^I\right]^N, \tag{64}$$

with

$$I = \frac{1}{2\pi}\int_0^{\pi} d\phi ln\left[\frac{1}{2}\left(1+(1-\kappa^2 sin^2\phi)^{1/2}\right)\right], \tag{65}$$

and

$$\kappa = 2sinh(2\beta J)/cosh^2(2\beta J). \tag{66}$$

## Analytic Results: two-dimensional Ising model

The resulting energy is given by

$$\langle E\rangle = -Jcoth(2\beta J)\left[1+\frac{2}{\pi}(2tanh^2(2\beta J)-1)K_1(q)\right], \tag{67}$$

with $q = 2sinh(2\beta J)/cosh^2(2\beta J)$ and the complete elliptic integral of the first kind

$$K_1(q) = \int_0^{\pi/2}\frac{d\phi}{\sqrt{1-q^2sin^2\phi}}. \tag{68}$$

## Analytic Results: two-dimensional Ising model

Differentiating once more with respect to temperature we obtain the specific heat given by

$$C_V = \frac{4k_B}{\pi}(\beta J coth(2\beta J))^2$$

$$\left\{K_1(q) - K_2(q) - (1 - tanh^2(2\beta J))\left[\frac{\pi}{2} + (2tanh^2(2\beta J) - 1)K_1(q)\right]\right\},$$

with

$$K_2(q) = \int_0^{\pi/2} d\phi\sqrt{1 - q^2 sin^2\phi}. \tag{69}$$

is the complete elliptic integral of the second kind. Near the critical temperature $T_C$ the specific heat behaves as

$$C_V \approx -\frac{2}{k_B\pi}\left(\frac{J}{T_C}\right)^2 ln\left|1 - \frac{T}{T_C}\right| + \mathrm{const.} \tag{70}$$

## Analytic Results: two-dimensional Ising model

In theories of critical phenomena one has that

$$C_V \sim \left|1 - \frac{T}{T_C}\right|^{-\alpha}, \tag{71}$$

and Onsager's result is a special case of this power law behavior. The limiting form of the function

$$lim_{\alpha\rightarrow 0}\frac{1}{\alpha}(Y^{-\alpha} - 1) = -lnY, \tag{72}$$

meaning that the analytic result is a special case of the power law singularity with $\alpha = 0$.

## Analytic Results: two-dimensional Ising model

One can also show that the mean magnetization per spin is

$$\left[1 - \frac{(1 - tanh^2(\beta J))^4}{16 tanh^4(\beta J)}\right]^{1/8}$$

for $T < T_C$ and 0 for $T > T_C$. The behavior is thus as $T \rightarrow T_C$ from below

$$M(T) \sim (T_C - T)^{1/8}$$

The susceptibility behaves as

$$\chi(T) \sim |T_C - T|^{-7/4}$$

## Correlation Length

Another quantity (given by the covariance) is the correlation function (defined in friday's lecture)

$$G_{ij} = \langle S_i S_j \rangle - \langle S_i \rangle \langle S_j \rangle. \tag{73}$$

and the correlation length

$$\xi^{-1} = -\lim_{r \to \infty} \frac{\partial}{\partial r} lnG(r), \tag{74}$$

with $r = |i - j|$.

## Scaling Results

Near $T_C$ we can characterize the behavior of many physical quantities by a power law behavior. As an example, the mean magnetization is given by

$$\langle \mathcal{M}(T) \rangle \sim (T - T_C)^{\beta} , \tag{75}$$

where $\beta$ is a so-called critical exponent. A similar relation applies to the heat capacity

$$C_V(T) \sim |T_C - T|^{-\alpha} , \tag{76}$$

the susceptibility

$$\chi(T) \sim |T_C - T|^{\gamma} . \tag{77}$$

and the correlation length

$$\xi(T) \sim |T_C - T|^{-\nu} . \tag{78}$$

$\alpha = 0$, $\beta = 1/8$, $\gamma = 7/4$ and $\nu = 1$. Later we will derive these coefficients from finite size scaling theories.

## Scaling Results

Through finite size scaling relations it is possible to relate the behavior at finite lattices with the results for an infinitely large lattice. The critical temperature scales then as

$$T_C(L) - T_C(L=\infty) \sim aL^{-1/\nu}, \tag{79}$$

$$\langle\mathcal{M}(T)\rangle \sim (T-T_C)^{\beta} \rightarrow L^{-\beta/\nu}, \tag{80}$$

$$C_V(T) \sim |T_C - T|^{-\gamma} \rightarrow L^{\gamma/\nu}, \tag{81}$$

and

$$\chi(T) \sim |T_C - T|^{-\alpha} \rightarrow L^{\alpha/\nu}. \tag{82}$$

We can compute the slope of the curves for $M$, $C_V$ and $\chi$ as function of lattice sites and extract the exponent $\nu$.

## Topics for Week 6, January 29- February 2

### Project 1

- Brief reminder from last week
- Markov processes and the Metropolis algorithm
- Discussion of project 1 and simulation of the Ising and Potts models.
- Time-correlation functions and the correlation time
- Discussions of the Wolff and Swendsen-Wang algos

## Brownian Motion and Markov Processes

A Markov process is a random walk with a selected probability for making a move. The new move is independent of the previous history of the system. The Markov process is used repeatedly in Monte Carlo simulations in order to generate new random states. The reason for choosing a Markov process is that when it is run for a long enough time starting with a random state, we will eventually reach the most likely state of the system. In thermodynamics, this means that after a certain number of Markov processes we reach an equilibrium distribution. This mimicks the way a real system reaches its most likely state at a given temperature of the surroundings.
To reach this distribution, the Markov process needs to obey two important conditions, that of **ergodicity** and **detailed balance**. These conditions impose then constraints on our algorithms for accepting or rejecting new random states. The Metropolis algorithm discussed here abides to both these constraints. The Metropolis algorithm is widely used in Monte Carlo simulations and the understanding of it rests within the interpretation of random walks and Markov processes.

## Brownian Motion and Markov Processes

In a random walk one defines a mathematical entity called a **walker**, whose attributes completely define the state of the system in question. The state of the system can refer to any physical quantities, from the vibrational state of a molecule specified by a set of quantum numbers, to the brands of coffee in your favourite supermarket.
The walker moves in an appropriate state space by a combination of deterministic and random displacements from its previous position.

This sequence of steps forms a **chain**.

## A simple Example

The obvious case is that of a random walker on a one-, or two- or three-dimensional lattice (dubbed coordinate space hereafter)
Consider a system whose energy is defined by the orientation of single spins. Consider the state $i$, with given energy $E_i$ represented by the following $N$ spins

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

We may be interested in the transition with one single spinflip to a new state $j$ with energy $E_j$

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

This change from one microstate $i$ (or spin configuration) to another microstate $j$ is the **configuration space** analogue to a random walk on a lattice. Instead of jumping from one place to another in space, we ’jump’ from one microstate to another.

## Brownian Motion and Markov Processes

We wish to study the time-development of a PDF after a given number of time steps. We define our PDF by the function $w(t)$. In addition we define a transition probability $W$. The time development of our PDF $w(t)$, after one time-step from $t = 0$ is given by

$$w_i(t=\epsilon) = W(j\rightarrow i)w_j(t=0).$$

This equation represents the discretized time-development of an original PDF. We can rewrite this as a

$$w_i(t=\epsilon) = W_{ij}w_j(t=0).$$

with the transition matrix $W$ for a random walk left or right (cannot stay in the same position) given by

$$W_{ij}(\epsilon) = W(il-jl,\epsilon) = \left\{\begin{array}{cc} \frac{1}{2} & |i-j| = 1\\ 0 & \mathrm{else} \end{array}\right.$$

We call $W_{ij}$ for the transition probability and we represent it as a matrix.

## Brownian Motion and Markov Processes

Both $W$ and $w$ represent probabilities and they have to be normalized, meaning that that at each time step we have

$$\sum_i w_i(t) = 1,$$

and

$$\sum_j W(j \rightarrow i) = 1.$$

Further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$. We can thus write the action of $W$ as

$$w_i(t+1) = \sum_j W_{ij} w_j(t),$$

or as vector-matrix relation

$$\hat{\mathbf{w}}(t+1) = \hat{\mathbf{W}}\hat{\mathbf{w}}(t),$$

and if we have that $||\hat{\mathbf{w}}(t+1) - \hat{\mathbf{w}}(t)|| \rightarrow 0$, we say that we have reached the most likely state of the system, the so-called steady state or equilibrium state. Another way of phrasing this is

$$\mathbf{w}(t=\infty) = \mathbf{W}\mathbf{w}(t=\infty).$$

## Brownian Motion and Markov Processes, a simple Example

Consider the simple $3 \times 3$ matrix $\hat{W}$

$$\hat{W} = \begin{pmatrix} 1/4 & 1/8 & 2/3 \\ 3/4 & 5/8 & 0 \\ 0 & 1/4 & 1/3 \end{pmatrix},$$

and we choose our initial state as

$$\hat{w}(t=0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The first iteration is

$$w_i(t=\epsilon) = W(j \rightarrow i) w_j(t=0),$$

resulting in

$$\hat{w}(t=\epsilon) = \begin{pmatrix} 1/4 \\ 3/4 \\ 0 \end{pmatrix}.$$

## Brownian Motion and Markov Processes, a simple Example

The next iteration results in

$$w_i(t = 2\epsilon) = W(j \rightarrow i) w_j(t = \epsilon),$$

resulting in

$$\hat{w}(t = 2\epsilon) = \left( \begin{array}{c} 5/23 \\ 21/32 \\ 6/32 \end{array} \right).$$

Note that the vector $\hat{w}$ is always normalized to 1. We find the steady state of the system by solving the linear set of equations

$$\mathbf{w}(t = \infty) = \mathbf{W}\mathbf{w}(t = \infty).$$

# Brownian Motion and Markov Processes, a simple Example

This linear set of equations reads

$$\begin{aligned} W_{11}w_1(t=\infty)+W_{12}w_2(t=\infty)+W_{13}w_3(t=\infty)= & \quad w_1(t=\infty) \\ W_{21}w_1(t=\infty)+W_{22}w_2(t=\infty)+W_{23}w_3(t=\infty)= & \quad w_2(t=\infty) \\ W_{31}w_1(t=\infty)+W_{32}w_2(t=\infty)+W_{33}w_3(t=\infty)= & \quad w_3(t=\infty) \end{aligned} \tag{83}$$

with the constraint that

$$\sum_i w_i(t=\infty)=1,$$

yielding as solution

$$\hat{w}(t=\infty)=\left(\begin{array}{c} 4/15 \\ 8/15 \\ 3/15 \end{array}\right).$$

## Brownian Motion and Markov Processes, a simple Example

Convergence of the simple example

| Iteration | $w_1$ | $w_2$ | $w_3$ |
|---|---|---|---|
| 0 | 1.00000 | 0.00000 | 0.00000 |
| 1 | 0.25000 | 0.75000 | 0.00000 |
| 2 | 0.15625 | 0.62625 | 0.18750 |
| 3 | 0.24609 | 0.52734 | 0.22656 |
| 4 | 0.27848 | 0.51416 | 0.20736 |
| 5 | 0.27213 | 0.53021 | 0.19766 |
| 6 | 0.26608 | 0.53548 | 0.19844 |
| 7 | 0.26575 | 0.53424 | 0.20002 |
| 8 | 0.26656 | 0.53321 | 0.20023 |
| 9 | 0.26678 | 0.53318 | 0.20005 |
| 10 | 0.26671 | 0.53332 | 0.19998 |
| 11 | 0.26666 | 0.53335 | 0.20000 |
| 12 | 0.26666 | 0.53334 | 0.20000 |
| 13 | 0.26667 | 0.53333 | 0.20000 |
| $\hat{w}(t=\infty)$ | 0.26667 | 0.53333 | 0.20000 |

Exercise: make a small program where you perform these iterations,but change the initial vector and study the convergence.

## Brownian Motion and Markov Processes, what is happening?

We have after $t$-steps

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^{\mathbf{t}}\hat{\mathbf{w}}(0),$$

with $\hat{\mathbf{w}}(0)$ the distribution at $t = 0$ and $\hat{\mathbf{W}}$ representing the transition probability matrix. We can always expand $\hat{\mathbf{w}}(0)$ in terms of the right eigenvectors $\hat{\mathbf{v}}$ of $\hat{\mathbf{W}}$ as

$$\hat{\mathbf{w}}(0) = \sum_i \alpha_i \hat{\mathbf{v}}_i,$$

resulting in

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^t\hat{\mathbf{w}}(0) = \hat{\mathbf{W}}^t \sum_i \alpha_i \hat{\mathbf{v}}_i = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i,$$

with $\lambda_i$ the $i^{\text{th}}$ eigenvalue corresponding to the eigenvector $\hat{\mathbf{v}}_i$.

## Brownian Motion and Markov Processes, what is happening?

If we assume that $\lambda_0$ is the largest eigenvector we see that in the limit $t \longrightarrow \infty$, $\hat{\mathbf{w}}(t)$ becomes proportional to the corresponding eigenvector $\hat{\mathbf{v}}_0$. This is our steady state or final distribution.
In our discussion below in connection with the entropy of a system and tomorrow's lecture on physics applications, we will relate these properties to correlation functions such as the time-correlation function.

That will allow us to define the so-called *equilibration time*,viz the time needed for the system to reach its most likely state. Form that state and on we can can compute contributions to various statistical variables.

## Brownian Motion and Markov Processes, what is happening?

We can relate this property to an observable like the mean magnetization of say a magnetic material. With the probabilty $\hat{\mathbf{w}}(t)$ we can write the mean magnetization as

$$\langle \mathcal{M}(t)\rangle = \sum_{\mu} \hat{\mathbf{w}}(t)_{\mu}\mathcal{M}_{\mu},$$

or as the scalar of a vector product

$$\langle \mathcal{M}(t)\rangle = \hat{\mathbf{w}}(t)\mathbf{m},$$

with $\mathbf{m}$ being the vector whose elements are the values of $\mathcal{M}_{\mu}$ in its various microstates $\mu$.
Recall our definition of an expectation value with a discrete PDF $p(x_i)$:

$$E[x^k] = \langle x^k\rangle = \frac{1}{N}\sum_{i=1}^{N} x_i^k p(x_i),$$

provided that the sums (or integrals) $\sum_{i=1}^{N} p(x_i)$ converge absolutely (viz , $\sum_{i=1}^{N} |p(x_i)|$ converges)

## Brownian Motion and Markov Processes, what is happening?

We rewrite the last relation as

$$\langle \mathcal{M}(t)\rangle = \hat{\mathbf{w}}(t)\mathbf{m} = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i \mathbf{m}_i.$$

If we define $m_i = \hat{\mathbf{v}}_i \mathbf{m}_i$ as the expectation value of $\mathcal{M}$ in the $i^{\mathrm{th}}$ eigenstate we can rewrite the last equation as

$$\langle \mathcal{M}(t)\rangle = \sum_i \lambda_i^t \alpha_i m_i.$$

Since we have that in the limit $t \rightarrow \infty$ the mean magnetization is dominated by the largest eigenvalue $\lambda_0$, we can rewrite the last equation as

$$\langle \mathcal{M}(t)\rangle = \langle \mathcal{M}(\infty)\rangle + \sum_{i \neq 0} \lambda_i^t \alpha_i m_i.$$

## Brownian Motion and Markov Processes, what is happening?

We define the quantity

$$\tau_i = -\frac{1}{log\lambda_i},$$

and rewrite the last expectation value as

$$\langle \mathcal{M}(t) \rangle = \langle \mathcal{M}(\infty) \rangle + \sum_{i \neq 0} \alpha_i m_i e^{-t/\tau_i}.$$

The quantities $\tau_i$ are the correlation times for the system. They control also the time-correlation functions to be discussed later.

The longest correlation time is obviously given by the second largest eigenvalue $\tau_1$, which normally defines the correlation time discussed above. For large times, this is the only correlation time that survives. If higher eigenvalues of the transition matrix are well separated from $\lambda_1$ and we simulate long enough, $\tau_1$ may well define the correlation time. In other cases we may not be able to extract a reliable result for $\tau_1$.

## Detailed Balance

An important condition we require that our Markov chain should satisfy is that of detailed balance. In statistical physics this condition ensures that it is e.g., the Boltzmann distribution which is generated when equilibrium is reached. The definition for being in equilibrium is that the rates at which a system makes a transition to or from a given state $i$ have to be equal, that is

$$\sum_i W(j \to i)w_j = \sum_i W(i \to j)w_i.$$

However, the condition that the rates should equal each other is in general not sufficient to guarantee that we, after many simulations, generate the correct distribution. We therefore introduce an additional condition, namely that of detailed balance

$$W(j \to i)w_j = W(i \to j)w_i.$$

At equilibrium detailed balance gives thus

$$\frac{W(j \to i)}{W(i \to j)} = \frac{w_i}{w_j}.$$

Proof:

$$\sum_i W(i \to j)w_i = \sum_i W(j \to i)w_j = w_j \sum_i W(j \to i) = w_j.$$

In a Markov process $w$ is known while $W$ is the unknown.

## Ergodicity

It should be possible for any Markov process to reach every possible state of the system from any starting point if the simulations is carried out for a long enough time. If any state in a distribution which has a probability different from zero and this state cannot be reached from any given starting point if we simulate long enough, then the system is not ergodic.

## The Boltzmann Distribution as Example

We introduce the Boltzmann distribution

$$w_i = \frac{\exp\left(-\beta(E_i)\right)}{Z},$$

which states that probability of finding the system in a state $i$ with energy $E_i$ at an inverse temperature $\beta = 1/k_BT$ is $w_i \propto \exp\left(-\beta(E_i)\right)$. The denominator $Z$ is a normalization constant which ensures that the sum of all probabilities is normalized to one. It is defined as the sum of probabilities over all microstates $j$ of the system

$$Z = \sum_j \exp\left(-\beta(E_i)\right).$$

From the partition function we can in principle generate all interesting quantities for a given system in equilibrium with its surroundings at a temperature $T$.

## Boltzmann Distribution as Example

With the probability distribution given by the Boltzmann distribution we are now in the position where we can generate expectation values for a given variable $A$ through the definition

$$\langle A\rangle = \sum_j A_j w_j = \frac{\sum_j A_j \exp\left(-\beta(E_j)\right)}{Z}.$$

In general, most systems have an infinity of microstates making thereby the computation of $Z$ practically impossible and a brute force Monte Carlo calculation over a given number of randomly selected microstates may therefore not yield those microstates which are important at equilibrium. To select the most important contributions we need to use the condition for detailed balance. Since this is just given by the ratios of probabilities, we never need to evaluate the partition function $Z$. For the Boltzmann distribution, detailed balance results in

$$\frac{w_i}{w_j} = \exp\left(-\beta(E_i - E_j)\right).$$

## Boltzmann Distribution as Example

Let us now specialize to a system whose energy is defined by the orientation of single spins. Consider the state $i$, with given energy $E_i$ represented by the following $N$ spins

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

We are interested in the transition with one single spinflip to a new state $j$ with energy $E_j$

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

We saw previously that this change from one microstate $i$ (or spin configuration) to another microstate $j$ is the configuration space analogue to a random walk on a lattice.

## Boltzmann Distribution as Example

However, the selection of states has to generate a final distribution which is the Boltzmann distribution. This is again the same we saw for a random walker, for the discrete case we had always a binomial distribution, whereas for the continuous case we had a normal distribution. The way we sample configurations should result in, when equilibrium is established, in the Boltzmann distribution. Else, our algorithm for selecting microstates has to be wrong.
Since we do not know the analytic form of the transition rate, we are free to model it as

$$W(i \rightarrow j) = g(i \rightarrow j)A(i \rightarrow j),$$

where $g$ is a selection probability while $A$ is the probability for accepting a move. It is also called the acceptance ratio.

## Boltzmann Distribution as Example

The selection probability should be same for all possible spin orientations, namely

$$g(i \rightarrow j) = \frac{1}{N}.$$

With detailed balance this gives

$$\frac{g(j \rightarrow i)A(j \rightarrow i)}{g(i \rightarrow j)A(i \rightarrow j)} = \exp\left(-\beta(E_i - E_j)\right),$$

but since the selection ratio is the same for both transitions, we have

$$\frac{A(j \rightarrow i)}{A(i \rightarrow j)} = \exp\left(-\beta(E_i - E_j)\right)$$

In general, we are looking for those spin orientations which correspond to the average energy at equilibrium.

## Boltzmann Distribution as Example

We are in this case interested in a new state $E_j$ whose energy is lower than $E_i$, viz., $\Delta E = E_j - E_i \leq 0$. A simple test would then be to accept only those microstates which lower the energy. Suppose we have ten microstates with energy $E_0 \leq E_1 \leq E_2 \leq E_3 \leq \cdots \leq E_9$. Our desired energy is $E_0$. At a given temperature $T$ we start our simulation by randomly choosing state $E_9$. Flipping spins we may then find a path from $E_9 \rightarrow E_8 \rightarrow E_7 \cdots \rightarrow E_1 \rightarrow E_0$. This would however lead to biased statistical averages since it would violate the ergodic hypothesis discussed above. This principle states that it should be possible for any Markov process to reach every possible state of the system from any starting point if the simulations is carried out for a long enough time.

## Boltzmann Distribution as Example

Any state in a Boltzmann distribution has a probability different from zero and if such a state cannot be reached from a given starting point, then the system is not ergodic. This means that another possible path to $E_0$ could be $E_9 \longrightarrow E_7 \longrightarrow E_8 \cdots \longrightarrow E_9 \longrightarrow E_5 \longrightarrow E_0$ and so forth. Even though such a path could have a negligible probability it is still a possibility, and if we simulate long enough it should be included in our computation of an expectation value.

Thus, we require that our algorithm should satisfy the principle of detailed balance and be ergodic. One possible way is the Metropolis algorithm.

## Metropolis Algorithm

The equation for detailed balance

$$\frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

is general and we could replace the Boltzmann distribution with other distributions as well. It specifies the ratio of pairs of acceptance probabilities, which leaves us with quite some room to manouvre.

- We give the largest of the two acceptance ratios the value 1 and adjust the other to satisfy the constraint.
- If $E_\mu < E_\nu$ then the larger of the two acceptance ratios is $A(\nu \rightarrow \mu)$ and we set to 1.
- Then we must have

$$A(\mu \rightarrow \nu) = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

if $E_\nu - E_\mu > 0$ and 1 otherwise. And that is the **Metropolis** algorithm.

## Implementation

- Establish an initial energy $E_b$
- Do a random change of this initial state by e.g., flipping an individual spin. This new state has energy $E_t$. Compute then $\Delta E = E_t - E_b$
- If $\Delta E \leq 0$ accept the new configuration.
- If $\Delta E > 0$, compute $w = e^{-(\beta \Delta E)}$.
- Compare $w$ with a random number $r$. If $r \leq w$ accept, else keep the old configuration.
- Compute the terms in the sums $\sum A_s P_s$.
- Repeat the above steps in order to have a large enough number of microstates
- For a given number of MC cycles, compute then expectation values.

## More general Definition

A Markov chain Monte Carlo method for the simulation of a distribution $p$ is any method producing an ergodic Markov chain of events $x$ whose stationary distribution is $p$.

- Generate an initial value $x^{(i)}$.
- Generate a trial value $y_t$ with probability $f(y_t|x^{(i)})$.
- Take a new value

$$x^{(i+1)} = \begin{cases} y_t & \text{with probability} = \rho(x^{(i)}, y_t) \\ x^{(i)} & \text{with probability} = 1 - \rho(x^{(i)}, y_t) \end{cases}$$

- We have defined

$$\rho(x, y) = \min\left\{ \frac{p(y)f(x|y)}{p(x)f(y|x)}, 1 \right\}.$$

  The distribution $f$ is often called the instrumental (we will relate it to the jumping of a walker) or proposal distribution while $\rho$ is the Metropolis-Hastings acceptance probability. When $f(y|x)$ is symmetric it is just called the Metropolis algo.

## Implementation

- Establish an initial state with some selected features to test.
- Do a random change of this initial state.
- Compute the Metropolis-Hastings acceptance probability $\rho$
- Compare $\rho$ with a random number $r$. If $r \leq \rho$ accept, else keep the old configuration.
- Compute the terms needed to obtain expectations values.
- Repeat the above steps in order to have as good statistics as possible.
- For a given number of MC cycles, compute then the final expectation values.

## Modelling the Ising Model

The code uses periodic boundary conditions with energy

$$E_i = -J \sum_{j=1}^{N} s_j s_{j+1},$$

In our case we have as the Monte Carlo sampling function the probability for finding the system in a state $s$ given by

$$P_s = \frac{e^{-(\beta E_s)}}{Z},$$

with energy $E_s$, $\beta = 1/kT$ and $Z$ is a normalization constant which defines the partition function in the canonical ensemble

$$Z(\beta) = \sum_s e^{-(\beta E_s)}$$

This is difficult to compute since we need all states. In a calculation of the Ising model in two dimensions, the number of configurations is given by $2^N$ with $N = L \times L$ the number of spins for a lattice of length $L$. Fortunately, the Metropolis algorithm considers only ratios between probabilities and we do not need to compute the partition function at all.

## Metropolis Algorithm

1. Establish an initial state with energy $E_b$ by positioning yourself at a random position in the lattice
2. Change the initial configuration by flipping e.g., one spin only. Compute the energy of this trial state $E_t$.
3. Calculate $\Delta E = E_t - E_b$. The number of values $\Delta E$ is limited to five for the Ising model in two dimensions, see the discussion below.
4. If $\Delta E \leq 0$ we accept the new configuration, meaning that the energy is lowered and we are hopefully moving towards the energy minimum at a given temperature. Go to step 7.
5. If $\Delta E > 0$, calculate $w = e^{-(\beta\Delta E)}$.
6. Compare $w$ with a random number $r$. If

$$r \leq w,$$

   then accept the new configuration, else we keep the old configuration and its values.

7. The next step is to update various expectations values.
8. The steps (2)-(7) are then repeated in order to obtain a sufficently good representation of states.

## Modelling the Ising Model

In the calculation of the energy difference from one spin configuration to the other, we will limit the change to the flipping of one spin only. For the Ising model in two dimensions it means that there will only be a limited set of values for $\Delta E$. Actually, there are only five possible values. To see this, select first a random spin position $x, y$ and assume that this spin and its nearest neighbors are all pointing up. The energy for this configuration is $E = -4J$. Now we flip this spin as shown below. The energy of the new configuration is $E = 4J$, yielding $\Delta E = 8J$.

$$E = -4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \uparrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad E = 4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \downarrow & \uparrow \\ & \uparrow & \end{array}$$

# Modelling the Ising Model

The four other possibilities are as follows

$$E = -2J \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \uparrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad E = 2J \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \downarrow & \uparrow \\ & \uparrow & \end{array}$$

with $\Delta E = 4J$,

$$E = 0 \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \uparrow & \uparrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad E = 0 \qquad \begin{array}{ccc} & \uparrow & \\ \downarrow & \downarrow & \uparrow \\ & \downarrow & \end{array}$$

with $\Delta E = 0$

## Modelling the Ising Model

$$E = 2J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \uparrow & \uparrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad E = -2J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \downarrow & \uparrow \\ & \downarrow & \end{array}$$

with $\Delta E = -4J$ and finally

$$E = 4J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \uparrow & \downarrow \\ & \downarrow & \end{array} \qquad \Longrightarrow \qquad E = -4J \qquad \begin{array}{ccc} & \downarrow & \\ \downarrow & \downarrow & \downarrow \\ & \downarrow & \end{array}$$

with $\Delta E = -8J$. This means in turn that we could construct an array which contains all values of $e^{\beta \Delta E}$ before doing the Metropolis sampling. Else, we would have to evaluate the exponential at each Monte Carlo sampling.

## The loop over $T$ in main

```
for ( double temp = initial_temp; temp <= final_temp; temp+=temp_ste
  //    initialise energy and magnetization
  E = M = 0.;
  // setup array for possible energy changes
  for( int de =-8; de <= 8; de++) w[de] = 0;
  for( int de =-8; de <= 8; de+=4) w[de+8] = exp(-de/temp);
  // initialise array for expectation values
  for( int i = 0; i < 5; i++) average[i] = 0.;
  initialize(n_spins, temp, spin_matrix, E, M);
  // start Monte Carlo computation
  for (int cycles = 1; cycles <= mcs; cycles++){
    Metropolis(n_spins, idum, spin_matrix, E, M, w);
    // update expectation values
    average[0] += E;    average[1] += E*E;
    average[2] += M;    average[3] += M*M; average[4] += fabs(M);
  }
  // print results
  output(n_spins, mcs, temp, average);
}
```

## The Initialise function

```
void initialize(int n_spins, double temp, int **spin_matrix,
double& E, double& M)
{
  // setup spin matrix and intial magnetization
  for(int y =0; y < n_spins; y++) {
    for (int x= 0; x < n_spins; x++){
      spin_matrix[y][x] = 1; // spin orientation for the ground state
      M +=  (double) spin_matrix[y][x];
    }
  }
  // setup initial energy
  for(int y =0; y < n_spins; y++) {
    for (int x= 0; x < n_spins; x++){
      E -=  (double) spin_matrix[y][x]*
(spin_matrix[periodic(y,n_spins,-1)][x] +
 spin_matrix[y][periodic(x,n_spins,-1)]);
    }
  }
}// end function initialise
```

## The periodic function

A compact way of dealing with periodic boundary conditions is given as follows:

```
// inline function for periodic boundary conditions
inline int periodic(int i, int limit, int add) {
  return (i+limit+add) % (limit);
```

with the following example from the function initialise

```
      E -=  (double) spin_matrix[y][x]*
(spin_matrix[periodic(y,n_spins,-1)][x] +
 spin_matrix[y][periodic(x,n_spins,-1)]);
```

## Alternative way for periodic boundary conditions

A more pedagogical way is given by the Fortran program

```
DO y = 1,lattice_y
    DO x = 1,lattice_x
       right = x+1 ; IF(x == lattice_x  ) right = 1
       left = x-1 ; IF(x == 1  ) left = lattice_x
       up = y+1 ; IF(y == lattice_y  ) up = 1
       down = y-1 ; IF(y == 1  ) down = lattice_y
       energy=energy - spin_matrix(x,y)*(spin_matrix(right,y)+&
            spin_matrix(left,y)+spin_matrix(x,up)+ &
            spin_matrix(x,down) )
       magnetization = magnetization + spin_matrix(x,y)
    ENDDO
 ENDDO
 energy = energy*0.5
```

## The Metropolis function

```
    // loop over all spins
    for(int y =0; y < n_spins; y++) {
      for (int x= 0; x < n_spins; x++){
        int ix = (int) (ran1(&idum)*(double)n_spins);   // RANDOM SPIN
        int iy = (int) (ran1(&idum)*(double)n_spins);  // RANDOM SPIN
        int deltaE =  2*spin_matrix[iy][ix]*
  (spin_matrix[iy][periodic(ix,n_spins,-1)]+
   spin_matrix[periodic(iy,n_spins,-1)][ix] +
   spin_matrix[iy][periodic(ix,n_spins,1)] +
   spin_matrix[periodic(iy,n_spins,1)][ix]);
        if ( ran1(&idum) <= w[deltaE+8] ) {
  spin_matrix[iy][ix] *= -1;  // flip one spin and accept new spin confi
          M += (double) 2*spin_matrix[iy][ix];
          E += (double) deltaE;
        }
      }
    }
```

## Expectation Values

```
double norm = 1/((double) (mcs));// divided by total number of cycle
double Eaverage = average[0]*norm;
double E2average = average[1]*norm;
double Maverage = average[2]*norm;
double M2average = average[3]*norm;
double Mabsaverage = average[4]*norm;
// all expectation values are per spin, divide by 1/n_spins/n_spins
double Evariance = (E2average- Eaverage*Eaverage)/n_spins/n_spins;
double Mvariance = (M2average - Mabsaverage*Mabsaverage)/n_spins/n_s
ofile << setiosflags(ios::showpoint | ios::uppercase);
ofile << setw(15) << setprecision(8) << temp;
ofile << setw(15) << setprecision(8) << Eaverage/n_spins/n_spins;
ofile << setw(15) << setprecision(8) << Evariance/temp/temp;
ofile << setw(15) << setprecision(8) << Maverage/n_spins/n_spins;
ofile << setw(15) << setprecision(8) << Mvariance/temp;
ofile << setw(15) << setprecision(8) << Mabsaverage/n_spins/n_spins
```

## Potts Model

The Potts model has been, in addition to the Ising model, widely used in studies of phase transitions in statistical physics. The so-called two-dimensional $q$-state Potts model has an energy given by

$$E = -J \sum_{<kl>}^{N} \delta_{s_l, s_k},$$

where the spin $s_k$ at lattice position $k$ can take the values $1, 2, \ldots, q$. The Kronecker delta function $\delta_{s_l, s_k}$ equals unity if the spins are equal and is zero otherwise. $N$ is the total number of spins.

## Potts Model

For $q = 2$ the Potts model corresponds to the Ising model. To see that we can rewrite the last equation as

$$E = -\frac{J}{2}\sum_{<kl>}^{N} 2(\delta_{s_l,s_k} - \frac{1}{2}) - \sum_{<kl>}^{N} \frac{J}{2}.$$

Now, $2(\delta_{s_l,s_k} - \frac{1}{2})$ is +1 when $s_l = s_k$ and $-1$ when they are different. This model is thus equivalent to the Ising model except a trivial difference in the energy minimum given by a an additional constant and a factor $J \to J/2$. One of the many applications of the Potts model is to helium absorbed on the surface of graphite.
For references on the Potts Models, see Barkema and Newman chapter 4.5 and Monroe at http://prola.aps.org/abstract/PRE/v66/i6/e066129 and Challa and Landau at `http://link.aps.org/abstract/PRB/v34/p1841`.

## Metropolis Algorithm for the Potts Model

1. Establish an initial state with energy $E_b$ by positioning yourself at a random position in the lattice
2. Change the initial configuration by flipping e.g., one spin only. Compute the energy of this trial state $E_t$.
3. Calculate $\Delta E = E_t - E_b$. The number of values $\Delta E$ is limited to five for the Ising model in two dimensions, see the discussion below.
4. If $\Delta E \leq 0$ we accept the new configuration, meaning that the energy is lowered and we are hopefully moving towards the energy minimum at a given temperature. Go to step 7.
5. If $\Delta E > 0$, calculate $w = e^{-(\beta \Delta E)}$.
6. Compare $w$ with a random number $r$. If
$$r \leq w,$$
then accept the new configuration, else we keep the old configuration and its values.
7. The next step is to update various expectations values.
8. The steps (2)-(7) are then repeated in order to obtain a sufficently good representation of states.

Metropolis for Potts model for $q \geq 5$ is inefficient! Use heat bath algo.

## Potts Model

Only four possible values for $\Delta E$

```
void Energy(double T,double *Boltzmann){

  Boltzmann[0] = exp(-J/T)  ;
  Boltzmann[1] = exp(-2*J/T);
  Boltzmann[2] = exp(-3*J/T);
  Boltzmann[3] = exp(-4*J/T);

}//Energy
```

## Potts Model

Must choose *q* randomly!

```
void Metropolis(int q,double *Boltzmann,int **Spin,long& seed,double&

  int  SpinFlip, LocalEnergy0, LocalEnergy, x, y, dE;

    for(int i = 0; i < N; i++){
      for(int j = 0; j < N; j++){
        x = (int) (ran1(&seed)*N);
        y = (int) (ran1(&seed)*N);
        LocalEnergy0 = 0;
        LocalEnergy = 0;
        dE = 0;
        if(Spin[x][y] == Spin[x][periodic(y,N,-1)])
          LocalEnergy0 --;
        if(Spin[x][y] == Spin[periodic(x,N,-1)][y])
          LocalEnergy0 --;
        if(Spin[x][y] == Spin[x][periodic(y,N,1)])
          LocalEnergy0 --;
        if(Spin[x][y] == Spin[periodic(x,N,1)][y])
          LocalEnergy0 --;
```

## Potts Model

```
do{
SpinFlip = (int)(ran1(&seed)*(q)+1);
}while(SpinFlip == Spin[x][y]);

if(SpinFlip == Spin[x][periodic(y,N,-1)])
  LocalEnergy --;
if(SpinFlip == Spin[periodic(x,N,-1)][y])
  LocalEnergy --;
if(SpinFlip == Spin[x][periodic(y,N,1)])
  LocalEnergy --;
if(SpinFlip == Spin[periodic(x,N,1)][y])
  LocalEnergy --;

dE =  LocalEnergy - LocalEnergy0;

if(dE<=0){
  Spin[x][y] = SpinFlip;
  E += J*dE;
}
else if(ran1(&seed)<Boltzmann[dE-1]){
  Spin[x][y] = SpinFlip;
  E += J*dE;
```

## Time Auto-correlation Function

Here we mention that one can show, using scaling relations, that at the critical temperature the correlation time $\tau$ relates to the lattice size $L$ as

$$\tau \sim L^{d+z},$$

with $d$ the dimensionality of the system. For the Metropolis algorithm based on a single spin-flip process, Nightingale and Blöte obtained $z = 2.1665 \pm 0.0012$. This is a rather high value, meaning that our algorithm is not the best choice when studying properties of the Ising model near $T_C$.

We can understand this behavior by studying the development of the two-dimensional

Ising model as function of temperature.

## Time Auto-correlation Function

Cooling the system down to the critical temperature we observe clusters pervading larger areas of the lattice. The reason for the large correlation time (and the parameter $z$) for the single-spin flip Metropolis algorithm is the development of these large domains or clusters with all spins pointing in one direction. It is quite difficult for the algorithm to flip over one of these large domains because it has to do it spin by spin, with each move having a high probability of being rejected due to the ferromagnetic interaction between spins. Since all spins point in the same direction, the chance of performing the flip

$$
E=-4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \uparrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad E=4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \downarrow & \uparrow \\ & \uparrow & \end{array}
$$

leads to an energy difference of $\Delta E = 8J$. Using the exact critical temperature $k_BT_C/J \approx 2.2.69$, we obtain a probability $\exp -(8/2.269) = 0.029429$ which is rather small. The increase in large correlation times due to increasing lattices can be diminished by using so-called cluster algorithms, such as that introduced by Ulli Wolff in 1989 and the Swendsen-Wang algorithm from 1987. The two-dimensional Ising model with the Wolff or Swendsen-Wang algorithms exhibits a much smaller correlation time, with the variable $z = 0.25 \pm 001$. Here, instead of flipping a single spin, one flips an entire cluster of spins pointing in the same direction.

## Time Auto-correlation Function

The so-called time-displacement autocorrelation $\phi(t)$ for the magnetization is given by

$$\phi(t) = \int dt' \left[\mathcal{M}(t') - \langle\mathcal{M}\rangle\right]\left[\mathcal{M}(t'+t) - \langle\mathcal{M}\rangle\right],$$

which can be rewritten as

$$\phi(t) = \int dt' \left[\mathcal{M}(t')\mathcal{M}(t'+t) - \langle\mathcal{M}\rangle^2\right],$$

where $\langle\mathcal{M}\rangle$ is the average value of the magnetization and $\mathcal{M}(t)$ its instantaneous value. We can discretize this function as follows, where we used our set of computed values $\mathcal{M}(t)$ for a set of discretized times (our Monte Carlo cycles corresponding to a sweep over the lattice)

$$\phi(t) = \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t}\mathcal{M}(t')\mathcal{M}(t'+t) - \frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t}\mathcal{M}(t')\times\frac{1}{t_{\max}-t}\sum_{t'=0}^{t_{\max}-t}\mathcal{M}(t'+t).$$

## Time Auto-correlation Function

One should be careful with times close to $t_{\mathrm{max}}$, the upper limit of the sums becomes small and we end up integrating over a rather small time interval. This means that the statistical error in $\phi(t)$ due to the random nature of the fluctuations in $\mathcal{M}(t)$ can become large.
One should therefore choose $t \ll t_{\mathrm{max}}$.
Note also that we could replace the magnetization with the mean energy, or any other expectation values of interest.

The time-correlation function for the magnetization gives a measure of the correlation between the magnetization at a time $t'$ and a time $t' + t$. If we multiply the magnetizations at these two different times, we will get a positive contribution if the magnetizations are fluctuating in the same direction, or a negative value if they fluctuate in the opposite direction. If we then integrate over time, or use the discretized version of, the time correlation function $\phi(t)$ should take a non-zero value if the fluctuations are correlated, else it should gradually go to zero. For times a long way apart the magnetizations are most likely uncorrelated and $\phi(t)$ should be zero.

## Time Auto-correlation Function

We can derive the correlation time by observing that our Metropolis algorithm is based on a random walk in the space of all possible spin configurations. Our probability distribution function $\hat{\mathbf{w}}(t)$ after a given number of time steps $t$ could be written as

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^{\mathbf{t}}\hat{\mathbf{w}}(0),$$

with $\hat{\mathbf{w}}(0)$ the distribution at $t = 0$ and $\hat{\mathbf{W}}$ representing the transition probability matrix. We can always expand $\hat{\mathbf{w}}(0)$ in terms of the right eigenvectors of $\hat{\mathbf{v}}$ of $\hat{\mathbf{W}}$ as

$$\hat{\mathbf{w}}(0) = \sum_i \alpha_i \hat{\mathbf{v}}_i,$$

resulting in

$$\hat{\mathbf{w}}(t) = \hat{\mathbf{W}}^t\hat{\mathbf{w}}(0) = \hat{\mathbf{W}}^t \sum_i \alpha_i \hat{\mathbf{v}}_i = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i,$$

with $\lambda_i$ the $i^{\mathrm{th}}$ eigenvalue corresponding to the eigenvector $\hat{\mathbf{v}}_i$.

## Time Auto-correlation Function

If we assume that $\lambda_0$ is the largest eigenvector we see that in the limit $t \rightarrow \infty$, $\hat{\mathbf{w}}(t)$ becomes proportional to the corresponding eigenvector $\hat{\mathbf{v}}_0$. This is our steady state or final distribution.
We can relate this property to an observable like the mean magnetization. With the probabilty $\hat{\mathbf{w}}(t)$ (which in our case is the Boltzmann distribution) we can write the mean magnetization as

$$\langle \mathcal{M}(t) \rangle = \sum_{\mu} \hat{\mathbf{w}}(t)_{\mu} \mathcal{M}_{\mu},$$

or as the scalar of a vector product

$$\langle \mathcal{M}(t) \rangle = \hat{\mathbf{w}}(t)\mathbf{m},$$

with $\mathbf{m}$ being the vector whose elements are the values of $\mathcal{M}_{\mu}$ in its various microstates $\mu$.

## Time Auto-correlation Function

We rewrite this relation as

$$\langle \mathcal{M}(t) \rangle = \hat{\mathbf{w}}(t)\mathbf{m} = \sum_i \lambda_i^t \alpha_i \hat{\mathbf{v}}_i \mathbf{m}_i.$$

If we define $m_i = \hat{\mathbf{v}}_i \mathbf{m}_i$ as the expectation value of $\mathcal{M}$ in the $i^{\mathrm{th}}$ eigenstate we can rewrite the last equation as

$$\langle \mathcal{M}(t) \rangle = \sum_i \lambda_i^t \alpha_i m_i.$$

Since we have that in the limit $t \rightarrow \infty$ the mean magnetization is dominated by the the largest eigenvalue $\lambda_0$, we can rewrite the last equation as

$$\langle \mathcal{M}(t) \rangle = \langle \mathcal{M}(\infty) \rangle + \sum_{i \neq 0} \lambda_i^t \alpha_i m_i.$$

We define the quantity

$$\tau_i = -\frac{1}{log \lambda_i},$$

and rewrite the last expectation value as

$$\langle \mathcal{M}(t) \rangle = \langle \mathcal{M}(\infty) \rangle + \sum_{i \neq 0} \alpha_i m_i e^{-t/\tau_i}.$$

## Time Auto-correlation Function

The quantities $\tau_i$ are the correlation times for the system. They control also the auto-correlation function discussed above. The longest correlation time is obviously given by the second largest eigenvalue $\tau_1$, which normally defines the correlation time discussed above. For large times, this is the only correlation time that survives. If higher eigenvalues of the transition matrix are well separated from $\lambda_1$ and we simulate long enough, $\tau_1$ may well define the correlation time. In other cases we may not be able to extract a reliable result for $\tau_1$. Coming back to the time correlation function $\phi(t)$ we can present a more general definition in terms of the mean magnetizations $\langle\mathcal{M}(t)\rangle$. Recalling that the mean value is equal to $\langle\mathcal{M}(\infty)\rangle$ we arrive at the expectation values

$$\phi(t) = \langle\mathcal{M}(0) - \mathcal{M}(\infty)\rangle\langle\mathcal{M}(t) - \mathcal{M}(\infty)\rangle,$$

resulting in

$$\phi(t) = \sum_{i,j\neq 0} m_i\alpha_i m_j\alpha_j e^{-t/\tau_i},$$

which is appropriate for all times.

## Topics for Week 7, February 5- 9

- Brief reminder from last week
- How to compute the time-correlation function and the correlation time
- Wolff and Swendsen-Wang algorithms
- Heat-Bath algorithm and Potts Model
- MPI instructions
- Histogram method and statistical analysis of data.

## Time Auto-correlation Function, Point b) in Project 1

The so-called time-displacement autocorrelation $\phi(t)$ for the magnetization is given by

$$\phi(t) = \int dt' \left[\mathcal{M}(t') - \langle \mathcal{M} \rangle\right] \left[\mathcal{M}(t'+t) - \langle \mathcal{M} \rangle\right],$$

which can be rewritten as

$$\phi(t) = \int dt' \left[\mathcal{M}(t')\mathcal{M}(t'+t) - \langle \mathcal{M} \rangle^2\right],$$

where $\langle \mathcal{M} \rangle$ is the average value of the magnetization and $\mathcal{M}(t)$ its instantaneous value. We can discretize this function as follows, where we used our set of computed values $\mathcal{M}(t)$ for a set of discretized times (our Monte Carlo cycles corresponding to a sweep over the lattice)

$$\phi(t) = \frac{1}{t_{\max} - t} \sum_{t'=0}^{t_{\max}-t} \mathcal{M}(t')\mathcal{M}(t'+t) - \frac{1}{t_{\max} - t} \sum_{t'=0}^{t_{\max}-t} \mathcal{M}(t') \times \frac{1}{t_{\max} - t} \sum_{t'=0}^{t_{\max}-t} \mathcal{M}(t'+t).$$

## Time Auto-correlation Function

One should be careful with times close to $t_{\mathrm{max}}$, the upper limit of the sums becomes small and we end up integrating over a rather small time interval. This means that the statistical error in $\phi(t)$ due to the random nature of the fluctuations in $\mathcal{M}(t)$ can become large. Note also that we could replace the magnetization with the mean energy, or any other expectation values of interest.

The time-correlation function for the magnetization gives a measure of the correlation between the magnetization at a time $t'$ and a time $t' + t$. If we multiply the magnetizations at these two different times, we will get a positive contribution if the magnetizations are fluctuating in the same direction, or a negative value if they fluctuate in the opposite direction. If we then integrate over time, or use the discretized version the time correlation function $\phi(t)$ should take a non-zero value if the fluctuations are correlated, else it should gradually go to zero. For times a long way apart the magnetizations are most likely uncorrelated and $\phi(t)$ should be zero.

## Correlation Time, Point f) in Project 1

The Ising model does not have dynamics built into it: there is no kinetic energy term associated with the spins $S_i$. The Metropolis Monte Carlo method generates successive configurations of spins, but this does not represent the real time evolution of a system of spins. In a Metropolis simulation, the successive spin configurations also exhibit a type of critical slowing down near the phase transition temperature $T_C(L)$ of the finite lattice. This is not the same as relaxation in a real system. However, it is useful to measure a relaxation time for the Metropolis ”dynamics” because it helps to determine how many steps to skip in order to generate statistically independent configurations. Recall that one Monte Carlo step per spin is taken conventionally to be $N$ Metropolis steps. If the correlation time is of the order of a single Monte Carlo step, then every configuration can be used in measuring averages. But if the correlation time is longer, then approximately Monte Carlo steps should be discarded between every data point.

## Correlation Time

If the correlation function decays exponentially

$$\phi(t) \sim \exp{(-t/\tau)}$$

then the exponential correlation time can be computed as the average

$$\tau_{\mathrm{exp}} = -\langle \frac{t}{log|\frac{\phi(t)}{\phi(0)}|} \rangle.$$

If the decay is exponential, then

$$\int_0^{\infty} dt \phi(t) = \int_0^{\infty} dt \phi(0) \exp{(-t/\tau)} = \tau\phi(0),$$

which suggests another measure of correlation

$$\tau_{\mathrm{int}} = \sum_k \frac{\phi(k)}{\phi(0)},$$

called the integrated correlation time.

## Better Algorithm needed

Monte Carlo simulations close to a phase transition are affected by critical slowing down. In the 2-D Ising system, the correlation length becomes very large, and the correlation time, which measures the number of steps between independent Monte Carlo configurations behaves like

$$\tau \sim \xi^z,$$

with $z \approx 2.1$ for the Metropolis algorithm. The exponent $z$ is called the dynamic critical exponent, The maximum possible value for $\xi$ in a finite system of $N = L \times L$ spins is $\xi \sim L$, because $\xi$ cannot be larger than the lattice size! This implies that $\tau \sim L^z \approx N$. This makes simulations difficult because the Metropolis algorithm time scales like $N$, so the time to generate independent Metropolis configurations scales like

$$N\tau \sim N^2 = L^4.$$

If the lattice size

$$L \rightarrow \sqrt{10}L \approx 3.2L$$

, the simulation time increases by a factor of 100.

## Better Algorithm needed

There is a simple physical argument which helps understand why $z = 2$, The Metropolis algorithm is a local algorithm, i.e., one spin is tested and flipped at a time. Near $T_C$ the system develops large domains of correlated spins which are difficult to break up. So the most likely change in configuration is the movement of a whole domain of spins. But one Metropolis sweep of the lattice can move a domain at most by approximately one lattice spacing in each time step. This motion is stochastic, i.e., like a random walk. The distance traveled in a random walk scales like $\sqrt{\mathrm{time}}$, so to move a domain a distance of order $\xi$ takes $\tau \sim \xi^2$ Monte Carlo steps. This argument suggests that the way to speed up a Monte Carlo simulation near $T_C$ is to use a non-local algorithm.

## Swendsen-Wang Algorithm, Ising Model

The essential idea of this algorithm suggested by R.H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987), is to identify clusters of like spins and treat each cluster as a giant spin to be flipped according to a random criterion. It is necessary that the algorithm obey the detailed balance condition. Swendsen and Wang found the following algorithm based on ideas from percolation theory.

## Swendsen-Wang Algorithm, Ising Model

Freeze/delete bonds: The $2-D$ square lattice, periodic boundary conditions, has $N = L \times L$ spins and $2N$ bonds between spins. Construct a bond lattice as follows:

- If the bond connects opposite spins, then delete it, i.e., temporarily uncouple the two spins. Note that opposite spins have a higher bond energy $+J$ if $J > 0$ and thus a higher effective temperature. So if $J$ is large we are effectively ”melting” the bond.
- If the bond connects like spins (both up or both down), then delete the bond with probability $e^{-2J/(k_B T)}$ , i.e., generate a random deviate $r$ and delete the bond if $r < e^{-2J/(k_B T)}$ . Note that a like-spin pair has bond energy $-J$ : so the change in energy in flipping one spin of the pair, i.e., in going from like to unlike spins is $E = 2J$ . Bonds which survive this test are ”frozen”. The probability of this happening is $1 - e^{-2J/(kBT)}$ . If $T = 0$ all like-spin bonds get frozen, while at $T = \infty$ the freezing probability is zero and all the bonds melt.

Note that constructing the bond lattice takes time of $O(N)$ because there are $2N$ bonds.

# Swendsen-Wang Algorithm, Ising Model

- After the bond lattice has been set up, the spins are decomposed into clusters. A cluster is simply a domain of spins connected to one another by frozen bonds. The lattice obviously decomposes into clusters in a unique way, and the decomposition is a deterministic problem. Cluster decomposition is potentially time consuming. A naive algorithm can take time of $O(N^2)$, so it is essential to use a decomposition algorithm that scales linearly with lattice size like Metropolis!
- Spin Up date: So far, constructing the bond lattice and identifying clusters has not changed any of the spins. The spins in each cluster are now ”frozen” and the bonds between different clusters have been deleted. Each cluster is now updated by assigning a random new value $\pm 1$ to all of the spins simultaneously, i.e., generate a random deviate $r$ and flip all spins in that cluster if $r < 0.5$. Note that $T$ does not play a role in this flipping decision. The spin update step scales like the number of clusters which is $< N$ . Swendsen and Wang showed that $z \approx 0.35$ for this algorithm in the $2 - D$ Ising model. Assuming that each Swendsen-Wang step scales like $N$ , the running time for the simulation scales like
$$N\tau \sim N\xi^{0.35} \sim NL^{0.35} \sim N^{1.175}$$
which is much better than $O(N^2)$ with Metropolis.

## Wolff Algorithm, Ising Model

Two years after Swendsen and Wang published their algorithm, U. Wolff, Phys. Rev. Lett. **62**, 361 (1989) published an even more efficient algorithm based on constructing and flipping one single cluster at a time Freeze/delete bonds: The $2-D$ square lattice, periodic boundary conditions, has $N = L \times L$ spins and $2N$ bonds between spins. Construct a bond lattice as follows:

- Choose a random spin in the lattice
- Look at the nearest neighbors of that spin. If they point in the same direction as the seed spin, add them to the cluster with a probability $P_{\mathrm{add}} = 1 - e^{-2J/(kBT)}$ as in the Swendsen-Wang algo.
- For each spin that is added, examine each ot its neighbors. If they point in the same direction, add each with the probability $P_{\mathrm{add}}$. Repeat till no more neighbors to consider for inclusions.
- Flip the cluster.

## Wolff Algorithm, Detailed Balance

Consider two states of the total lattice, $i$ a and $j$. They differ from one another by the flipping of a single cluster of similarly oriented spins. The crucial thing to notice is the way the spins are oriented around the edge of the cluster. The bonds between these spins and the ones in the cluster have to be broken when the cluster is flipped. This means that the bonds which are not broken in going from $i$ to $j$ must be broken when we flip back again from $j$ to $i$
When we move from $i$ to $j$ we break $m$ bonds in order to flip the cluster. These bonds represent pairs of similarly oriented spins which are not added to the cluster. The probability of not adding a spin is $1 - P_{\mathrm{add}}$. If we break $m$ bonds we have a selection probability $(1 - P_{\mathrm{add}})^m$.

If there are $n$ bonds to break in the reverse move, then the selection probability is

$(1 - P_{\mathrm{add}})^n$.

## Wolff Algorithm, Detailed Balance

An important condition we require that our Markov chain should satisfy is that of detailed balance. At equilibrium detailed balance gives thus

$$\frac{W(j\rightarrow i)}{W(i\rightarrow j)} = \frac{w_i}{w_j}.$$

We model the transition probability as $W(j\rightarrow i) = g(j\rightarrow i)A(j\rightarrow i)$, where $g$ is the selection probability and $A$ is the acceptance probability. Our cluster algo gives

$$\frac{g(j\rightarrow i)A(j\rightarrow i)}{g(i\rightarrow j)A(i\rightarrow j)} = \frac{w_i}{w_j},$$

which with the Boltzmann distribution results in

$$(1-P_{\mathrm{add}})^{m-n}\frac{A(j\rightarrow i)}{A(i\rightarrow j)} = e^{-\beta(E_i-E_j)}.$$

## Wolff Algorithm, Detailed Balance

The change between the two states depends on the number of broken bonds. For each broken bond of the *m* ones the change is (Ising model, you must figure out what this means for the Potts model) $+2J$. For each broken bond for the reverse process (total *n*) we an energy change $-2J$. That means that we get

$$(1 - P_{\mathrm{add}})^{m-n}\frac{A(j \to i)}{A(i \to j)} = e^{-\beta 2J(m-n)},$$

which we rewrite as

$$\frac{A(j \to i)}{A(i \to j)} = \left(e^{\beta 2J}(1 - P_{\mathrm{add}})\right)^{n-m},$$

and using the fact that $P_{\mathrm{add}} = 1 - e^{-2J/(kBT)}$ we obtain a ratio between the acceptance probabilities as

$$\frac{A(j \to i)}{A(i \to j)} = 1$$

The cluster is always flipped. We make the acceptance ratios for both forward and backward moves equal unity. Note that we need not go through the whole lattice to flip a cluster, this should be contrasted with the Swendsen-Wang also. Wolff showed that $z \approx 0.25$ for this algorithm.

## Wolff Algorithm, Ising Code, useful for Point g) in Project 1

```
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <list>
#include "rng.h"

using namespace std;

double J = +1;                // ferromagnetic coupling
int Lx, Ly;                   // number of spins in x and y
int N;                        // number of spins
int **s;                      // the spins
double T;                     // temperature
double H = 0;                 // magnetic field
int steps;                    // number of Monte Carlo steps
```

## Wolff Algorithm

```
void initialize ( ) {
    s = new int* [Lx];
    for (int i = 0; i < Lx; i++)
        s[i] = new int [Ly];
    for (int i = 0; i < Lx; i++)
        for (int j = 0; j < Ly; j++)
            s[i][j] = qadran() < 0.5 ? +1 : -1;   // hot start
    steps = 0;
}
```

## Wolff Algorithm

This function initialises the cluster with its given probability.

```
bool **cluster;                          // cluster[i][j] = true if i,j bel
double addProbability;                   // 1 - e^(-2J/kT)

void initializeClusterVariables() {

    // allocate 2-D array for spin cluster labels
    cluster = new bool* [Lx];
    for (int i = 0; i < Lx; i++)
        cluster[i] = new bool [Ly];

    // compute the probability to add a like spin to the cluster
    addProbability = 1 - exp(-2*J/T);
}
```

The array to mark whether a spin belongs to the cluster or not is given by the variable cluster.

## Wolff Algorithm

At each Monte Carlo step a single cluster is grown around a randomly chosen seed spin and all the spins of this cluster are flipped.

```
// declare functions to implement Wolff algorithm
void growCluster(int i, int j, int clusterSpin);
void tryAdd(int i, int j, int clusterSpin);

void oneMonteCarloStep() {

    // no cluster defined so clear the cluster array
    for (int i = 0; i < Lx; i++)
    for (int j = 0; j < Lx; j++)
        cluster[i][j] = false;

    // choose a random spin and grow a cluster
    int i = int(qadran() * Lx);
    int j = int(qadran() * Ly);
    growCluster(i, j, s[i][j]);

    ++steps;
}
```

## Wolff Algorithm

This function grows a Wolff cluster and simultaneously flips all spins in the cluster. We do it in two steps

```
void growCluster(int i, int j, int clusterSpin) {

    // mark the spin as belonging to the cluster and flip it
    cluster[i][j] = true;
    s[i][j] = -s[i][j];

    // find the indices of the 4 neighbors
    // assuming periodic boundary conditions
    int iPrev = i == 0    ? Lx-1 : i-1;
    int iNext = i == Lx-1 ? 0    : i+1;
    int jPrev = j == 0    ? Ly-1 : j-1;
    int jNext = j == Ly-1 ? 0    : j+1;
}
```

## Wolff Algorithm

```
// if the neighbor spin does not belong to the
// cluster, then try to add it to the cluster
if (!cluster[iPrev][j])
    tryAdd(iPrev, j, clusterSpin);
if (!cluster[iNext][j])
    tryAdd(iNext, j, clusterSpin);
if (!cluster[i][jPrev])
    tryAdd(i, jPrev, clusterSpin);
if (!cluster[i][jNext])
    tryAdd(i, jNext, clusterSpin);
```

## Wolff Algorithm

```
void tryAdd(int i, int j, int clusterSpin) {
    if (s[i][j] == clusterSpin)
        if (qadran() < addProbability)
            growCluster(i, j, clusterSpin);
}
```

## Wolff Algorithm

```
// variables to measure chi and its error estimate
double chi;                 // current susceptibility per spin
double chiSum;              // accumulate chi values
double chiSqdSum;           // accumulate chiˆ2 values
int nChi;                   // number of values accumulated

// variables to measure autocorrelation time
int nSave = 10;             // number of values to save
double cChiSum;             // accumulate
list<double> chiSave;       // the saved values
double *cChi;               // correlation sums
int nCorr;                  // number of values accumulated

// variables to estimate fluctuations by blocking
int stepsPerBlock = 1000;   // suggested in Wolff paper
double chiBlock;            // used to calculate block average
double chiBlockSum;         // accumulate block <chi> values
double chiBlockSqdSum;      // accumulate block <chi>ˆ2 values
int stepInBlock;            // number of steps in current block
int blocks;                 // number of blocks
```

## Wolff Algorithm

```
void initializeObservables() {
    chiSum = chiSqdSum = 0;
    nChi = 0;
    chiBlock = chiBlockSum = chiBlockSqdSum = 0;
    stepInBlock = blocks = 0;
    cChiSum = 0;
    cChi = new double [nSave + 1];
    for (int i = 0; i <= nSave; i++)
        cChi[i] = 0;
    nCorr = 0;
}
```

## Wolff Algorithm

```
void measureObservables() {

    // observables are derived from the magnetic moment
    int M = 0;
    for (int i = 0; i < Lx; i++)
    for (int j = 0; j < Ly; j++)
        M += s[i][j];
    chi = M * double(M) / double(N);
    // accumulate values
    chiSum += chi;
    chiSqdSum += chi * chi;
    ++nChi;
    // accumulate correlation values
    if (chiSave.size() == nSave) {
        cChiSum += chi;
        cChi[0] += chi * chi;
        ++nCorr;
        list<double>::const_iterator iter = chiSave.begin();
        for (int i = 1; i <= nSave; i++)
            cChi[i] += *iter++ * chi;
        chiSave.pop_back();     // remove oldest saved chi value
    }
    chiSave.push_front(chi);    // add current chi value
```

## Wolff Algorithm

```
// accumulate block values
chiBlock += chi;
++stepInBlock;
if (stepInBlock == stepsPerBlock) {
    chiBlock /= stepInBlock;
    chiBlockSum += chiBlock;
    chiBlockSqdSum += chiBlock * chiBlock;
    ++blocks;
    stepInBlock = 0;
    chiBlock = 0;
}
```

## Wolff Algorithm

```
// averages of observables
double chiAve;                   // average susceptibility per spin
double chiError;                 // Monte Carlo error estimate
double chiStdDev;                // Standard deviation error from blocking
double tauChi;                   // autocorrelation time
double tauEffective;             // effective autocorrelation time
```

## Wolff Algorithm

```
void computeAverages()  {
    // average susceptibility per spin
    chiAve = chiSum / nChi;
    // Monte Carlo error estimate
    chiError = chiSqdSum / nChi;
    chiError = sqrt(chiError - chiAve * chiAve);
    chiError /= sqrt(double(nChi));
    // exponential correlation time
    tauChi = 0;
    double cAve = cChiSum / nCorr;
    double c0 = cChi[0] / nCorr - cAve * cAve;
    for (int i = 1; i <= nSave; i++) {
         double c = (cChi[i] / nCorr - cAve * cAve) / c0;
         if (c > 0.01) {
             tauChi += -i/log(c);
         } else {
             tauChi /= (i - 1);
             break;
         }
         if (i == nSave)
             tauChi /= nSave;
    }
    // standard deviation from blocking
    double chiBlockAve = chiBlockSum / blocks;
```

## Wolff Algorithm, Main function

```
int main() {

    cout << " Two-dimensional Ising Model - Wolff Cluster Algorithm\n"
         << " -----------------------------------------------------\n"
         << " Enter number of spins L in each direction: ";
    cin >> Lx;
    Ly = Lx;
    N = Lx * Ly;
    cout << " Enter temperature T: ";
    cin >> T;
    cout << " Enter number of Monte Carlo steps: ";
    int MCSteps;
    cin >> MCSteps;

    initialize();
    initializeClusterVariables();
    int thermSteps = MCSteps / 5;
    cout << " Performing " << thermSteps
         << " thermalization steps ..." << flush;
    for (int i = 0; i < thermSteps; i++)
        oneMonteCarloStep();
```

## Wolff Algorithm, Main function

```
    cout << " done\n Performing production steps ..." << flush;
    initializeObservables();
    for (int i = 0; i < MCSteps; i++) {
        oneMonteCarloStep();
        measureObservables();
    }
    cout << " done" << endl;
    computeAverages();
    cout << "\n       Average chi per spin = " << chiAve
         << "\n Monte Carlo error estimate = " << chiError
         << "\n   Autocorrelation time tau = " << tauChi
         << "\n   Std. Dev. using blocking = " << chiStdDev

         << "\n              Effective tau = " << tauEffective << endl
}
```

# Repeat: Metropolis Algorithm

The equation for detailed balance

$$\frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

results in the following algo

- We give the largest of the two acceptance ratios the value 1 and adjust the other to satisfy the constraint.
- If $E_\mu < E_\nu$ then the larger of the two acceptance ratios is $A(\nu \rightarrow \mu)$ and we set to 1.
- Then we must have

$$A(\mu \rightarrow \nu) = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

if $E_\nu - E_\mu > 0$ and 1 otherwise. And that is the **Metropolis** algorithm.

## Potts Model and Metropolis

In the calculation of the energy difference from one spin configuration to the other, we have for the $q = 2$ Potts two possible values only. When change one of the values such as flipping a spin we start with an energy $E = -4J$. Now we flip this spin as shown below. The energy of the new configuration is $E = 0J$, yielding $\Delta E = 4J$.

$$E = -4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \uparrow & \uparrow \\ & \uparrow & \end{array} \qquad \Longrightarrow \qquad E = 4J \qquad \begin{array}{ccc} & \uparrow & \\ \uparrow & \downarrow & \uparrow \\ & \uparrow & \end{array}$$

## Potts Model and Heat-Bath Algorithm, last Point of Project

However, when $q$ becomes large the standard Metropolis algorithm becomes inefficient. Assume that $q = 100$. At high $T$ the acceptance probability is close to 1 and our algorithm is efficient.
When we cool down the system $T \rightarrow T_C$, more and more 'spins' will take the same value and we build up cluster/domains with equally valued 'spins'. If the spins is aligned with its neigbours it has lower energy and thereby larger $e^{-\beta E}$ weight.
The problem comes when $q$ is large. If our value is one of the other 96 values, we need on average $100/4 = 25$ steps to find a desired state. Long time to find state with lower energy.
If we start at low temperatures, there is an extra cost to excite, leading to smaller acceptance probability. Could then have almost 96 out 100 moves rejected.

We need a better algo again.

## Potts Model and Heat-Bath Algorithm

- Choose a random spin $S_i$ in the lattice
- Regardless of its value, choose a new value $S_i$ in proportion to the Boltzmann weight for the different values, the values are drawn from a so-called heat-bath. In other words, we give the spin a value $n$ between 1 and $q$ ($n \in [1, q]$) with a probability
$$p_n = \frac{e^{\beta E_n}}{\sum_{m=1}^{q} e^{\beta E_n}},$$
with $E_n$ the energy of the system with spin $S_i = n$
- The probabilities of add up to one. The probability of making the transition from a state with $S_i = n$ to one in which $S_i = m$ is then $p_m$ and for going back $p_n$. Note that these probabilities do not depend on the initial state, only on the final one, unlike the Metropolis algo.

## Potts Model and Heat-Bath Algorithm

The equation for detailed balance

$$\frac{P(n \to m)}{P(m \to n)} = \frac{p_m}{p_n} = \exp\left(-\beta(E_m - E_n)\right)$$

The algorithm is much more efficient than the standard Metropolis for large values of $q$ since it will choose the states with the highest Boltzmann weight most often and can find them in one move instead of wandering among large numbers of unfavourable states.
For $q = 2$ it is less efficient than the Metropolis algorithm. For the Ising model it gives and acceptance ratio of

$$A = \frac{\exp-(1/2\beta\Delta E)}{\exp-(1/2\beta\Delta E) + \exp(1/2\beta\Delta E)}$$

which is smaller than the standard $\exp-(\beta\Delta E)$.

## Bootstrap and Jackknife Methods, formalities

A good reference is M. C. K. Yang and David H. Robinson, Understanding and Learning Statistics by Computer, (World Scientific, Singapore, 1986).
Simple example: suppose we want to estimate the mass of an elementary particle as predicted in a numerical simulation. The mass is obtained by fitting an exponential to a simulation data set as follows:

$$y(t) = a\exp(-mt)$$

where the data are given as a table of $y$ values for integer values of $t$, as

$$\{y(0), y(1), y(2), \dots, y(L)\}.$$

Actually the simulation spits out a list of such values in one single measurement, runs for a while, and spits out another list, and so on. So our data set looks like

$$\{y_i(0), y_i(1), y_i(2), \dots, y_i(L)\},$$

where $i = 1, \dots, N$ labels a list of measurements.

## Bootstrap and Jackknife Methods

We might think all we have to do is to take the raw data and construct means $\bar{y}(t)$ and standard errors $\sigma(t)$ at each time $t$ and then do a standard least chi square fit. We would get the best values for the parameters $a$ and $m$ and we would get the errors from the error matrix. But we have a problem. The standard chi square fit assumes that the fluctuations in the data points are statistically independent. It turns out that with the numerical simulations (also often a problem with experimental data as well) the fluctuations in the data are correlated. That is, if $y(0)$ fluctuates upwards, chances are better that $y(1)$ also fluctuates upwards. So we can't use the standard formula for chi square. Now it is possible to modify the formula for chi square to take proper account of the correlations. But the analysis becomes much more involved, so one would like to develop more confidence in the resulting error in the mass parameter.

## Bootstrap and Jackknife Methods, formalities

Starting from a sample of $N$ measurements, the jackknife begins by throwing out the first measurement, leaving a jackknife data set of $N-1$ "resampled" values. The statistical analysis is done on the reduced sample, giving a measured value of a parameter, say $m_{J1}$. Then a new resampling is done, this time throwing out the second measurement, and a new measured value of the parameter is obtained, say $m_{J2}$. The process is repeated for each set $i$ in the sample, resulting in a set of parameter values $\{m_{Ji}, i = 1, \dots, N\}$. The standard error is given by the formula

$$\sigma^2_{\mathrm{Jmean}} = (N-1)\sum_{i=1}^{N}(m_{Ji} - m)^2/N$$

where $m$ is the result of fitting the full sample.

## Bootstrap and Jackknife Methods, formalities

The jackknife method is also capable of giving an estimate of sampling bias. We may have a situation in which a parameter estimate tends to come out on the high side (or low side) of its true value if a data sample is too small. Thus the estimate $m$ derived from a fit to $N$ data points may be higher (or lower) than the true value. When this happens, we might expect that removing a measurement, as we do in the jackknife, would enhance the bias. We measure this effect by comparing the mean of the jackknife values $m_{Ji}$, call it $m_{J.}$ with the result $m$ of fitting the full data set. If there is a difference, we can correct for the bias using

$$\tilde{m} = m - (N-1)(m_{J.} - m)$$

## Bootstrap and Jackknife Methods, formalities

To see how the jackknife works, let us consider the much simpler problem of computing the mean and standard deviation of the mean of a random sample $\{x_i\}$. The conventional approach gives

$$\begin{aligned}
\bar{x} &= \sum_{j=1}^{N} x_i/N \\
\sigma^2_{\mathrm{mean}} &= \sum_{j=1}^{N} (x_j - \bar{x})^2/[N(N-1)]
\end{aligned}$$

The jackknife approach computes the jackknife sample means

$$x_{Ji} = \sum_{j \neq i} x_i/(N-1)$$

for $i = 1, \dots, N$. Then we compute the jackknife error in the mean, which is given by

$$\sigma^2_{\mathrm{Jmean}} = (N-1)\sum_{i=1}^{N} (x_{Ji} - \bar{x})^2/N$$

## Bootstrap and Jackknife Methods, formalities

Compare the placement of the factors of $N$ and $N-1$ here with the expression for $\sigma_{\rm mean}$. The reason for the difference is that the jackknife sample means are distributed $N-1$ times closer to the mean than the original values $x_i$, so we need a correction factor of $(N-1)^2$. In fact for this simple example, it is easy to show that

$$x_{Ji} - \bar{x} = (\bar{x} - x_i)/(N-1).$$

Consequently we can show trivially that

$$\sigma_{\rm Jmean} = \sigma_{\rm mean}$$

so the jackknife procedure hasn't gained us anything in this simple case. But our example of determining the mass of an elementary particle is not so simple. The error estimate is found from Eq ([*]). This error estimate is not likely to be the same as the error obtained from a full correlated chi square analysis. However, we expect that in the limit of an infinitely large sample, both estimates should agree.

So if we get two error estimates and they don't agree, which should we believe? A conservative approach would take the larger of the two.

## Topics for Week 8, February 12- 16

- Brief reminder from last week
- Histogram method and statistical analysis of data.
- Finite size scaling and Monte Carlo renormalization group

## Histogram Method

During the simulation we can build up an estimate of the complete density $P_\beta(O)$, and clearly it would be beneficial to utilize this information. This insight is the key to *histogram methods*. In 1989 Ferrenberg and Swendsen published a method to combine results obtained at different couplings. The method was highly efficient, and Ferrenberg-Swendsen reweighting has become an essential tool for MC practitioners. The use of rawdata from several couplings allow for reweighting to a much broader range of couplings than ordinary single histogram methods. When doing a MC simulation with the Metropolis algorithm the probability to be in a state $\psi$ with energy $\epsilon_\psi$ is proportional to

$$w(\epsilon_\psi)e^{-\beta\epsilon_\psi}.$$

## Histogram Method

If we record a histogram of energies from a simulation at coupling $\beta$; we get a histogram $h_\beta(\epsilon)$ which is proportional to $w(\epsilon)e^{-\beta\epsilon}$. Multiplying this histogram with $e^{\beta\epsilon}$ we get something which is proportional to $w(\epsilon)$, i.e.

$$\hat{w}_\beta(\epsilon) = e^{\xi_\beta} e^{\beta\epsilon} h_\beta(\epsilon)$$

is an estimator for $w(\epsilon)$. Here $e^{\xi_\beta}$ is a dimensionless constant of proportionality to be determined. The density of states has an index $\beta$ to indicate that the histogram was recorded at this coupling, but it does not have any intrinsic temperature dependence. In principle this equation can be used to estimate $w(\epsilon)$ regardless of temperature, however practically only a small energy range around $\langle E\rangle(T)$ will be sampled with a sufficiently high frequency.

## Histogram Method

Although the equation is useless as an immediate estimator for $w(\epsilon)$, it provides a basis for *combining* results from different couplings to an estimator $\hat{w}(\epsilon)$ which can be applied over the complete energy range. Given $N$ different histograms $h_i(E)$ recorded at the couplings $\beta_1 > \beta_2 > \cdots > \beta_N$, we can combine them as

$$\hat{w}(\epsilon) = w_0 \sum_{i=1}^{N} e^{\xi_i} h_i(\epsilon) W_i(\epsilon) e^{\beta_i \epsilon},$$

$$W_i(\epsilon) = \frac{h_i(\epsilon)}{\sum_{i=1}^{N} h_i(\epsilon)},$$

to obtain an estimator which is usable over the complete $\epsilon$ range $[\min_\epsilon h_i(\epsilon), \max_\epsilon h(\epsilon)]$. $W_i(\epsilon)$ is a weight function, which denotes the weight ascribed to histogram $i$ in the estimation of $w(\epsilon)$. The constants $e^{\xi_i}$ are determined by joining the various histograms.

## Histogram Method

The algorithm we have applied to determine $\xi_i$ is to set $\xi_1$ to an arbitrary value, and then compute $\xi_{i>1}$ by minimising

$$\chi^2 = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{\epsilon} h_i(\epsilon) h_j(\epsilon) \underbrace{\left(\xi_i + \beta_i \epsilon + \ln h_i(\epsilon) - \xi_j - \beta_j \epsilon - \ln h_j(\epsilon)\right)^2}_{\ln \hat{w}_{\beta_i}(\epsilon) - \ln \hat{w}_{\beta_j}(\epsilon)}.$$

## Histogram Method

The central principle is to minimise the pairwise difference between all the $\hat{g}(\epsilon)$ estimates.
Minimising $\chi^2$ with respect to $\xi_i$ gives $N-1$ linear equations which can be solved by e.g. LU decomposition.
When the coefficients $\xi_i$ have been determined we have all the coefficients $\xi_{i>1}$ expressed in terms of $\xi_1$. For discrete models with a finite ground state degeneracy $w_0$ we can determine $\xi_1$ by requiring $w(\epsilon_0) = w_0$, or alternatively if the *total number of states* is known, this can be used to normalize $w(\epsilon)$.

To determine $w(\epsilon)$ is in principle quite straightforward, but in practice it is important to be careful to avoid numeric underflow or overflow in intermediate steps, in particular the implementation must ensure that only $\ln w(\epsilon)$ is needed in actual computations.

## Histogram Method

Knowledge of $w(\epsilon)$ is in principle equivalent to knowledge of the partition function $Z(\beta)$, hence all the properties of a system are contained in $w(\epsilon)$, however $w(\epsilon)$ does not have a very prominent role in modern statistical mechanics. We will therefor express some important results based on $w(\epsilon)$.
The definition of temperature in statistical mechanics is given by

$$\beta = \frac{\partial \ln w(\epsilon)}{\partial \epsilon}.$$

## Histogram Method

From this we find that the fundamental requirement $C_V(T) \geq 0$ is equivalent to $\partial_\epsilon^2 \ln w(\epsilon) \leq 0$. The limiting value $\partial_\epsilon \ln w(\epsilon) = \mathcal{C}$ is the signature of a phase transition. A finite $\epsilon$ range with $\partial_\epsilon \ln w(\epsilon) = \mathcal{C}$ means that the temperature is unchanged for this $\epsilon$ range, i.e. an indication of a first order transition (actually, this is slightly more complicated). When the width of the of linear part of $\ln w(\epsilon)$ diminishes the first order transition is weakened; until $\partial_\epsilon^2 \ln w(\epsilon) = 0$ in a isolated point only, this is the manifestation of a critical point. If we differentiate with respect to $T$ we find the function $C_V(\epsilon)$

$$C_V(\epsilon) = \frac{\partial \epsilon}{\partial T} = \frac{-\left(\partial_\epsilon \ln w(\epsilon)\right)^2}{\partial_\epsilon^2 \ln w(\epsilon)}.$$

## Histogram Method

From this we see that the critical properties, and in particular the critical exponent $\alpha$, must be related to how $\partial_\epsilon^2 \ln w(\epsilon)$ approaches zero. To infer $\alpha$ directly from the behaviour of $w(\epsilon)$ close to $\epsilon_c$ is difficult, but if we make the size dependence of $w(\epsilon)$ explicit we can use finite size scaling.
At the (pseudo)critical point in a finite system, $C_V$ scales as $L^{d+\alpha/\nu}$. The factor $(\partial_\epsilon \ln w(\epsilon))^2$ in the heat capacity is just equal to $\beta_c^2$, hence the critical properties must come from the second derivative

$$\left|\partial_\epsilon^2 \ln w(\epsilon, L)\right| L^d \propto L^{-\alpha/\nu}.$$

In general $\partial_\epsilon \ln w(\epsilon, L)$ will also have finite size effects, however for this only the *deviation* from the thermodynamic value will show critical scaling.

## Histogram Method

For microcanonical systems the externally specified variable is $\epsilon$, and not $T$, and critical scaling is governed by the difference $|\epsilon - \epsilon_C|$.
When we have $w(\epsilon)$ we can easily calculate $F(T)$ and $P(\epsilon, T)$

$$F(T) = -T \ln \sum_{\epsilon} w(\epsilon) e^{-\beta\epsilon},$$

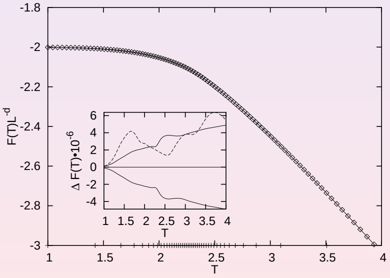
$$P(\epsilon, T) = \frac{w(\epsilon) e^{-\beta\epsilon}}{\sum_{\epsilon} w(\epsilon) e^{-\beta\epsilon}}.$$

From $P(\epsilon, T)$ we can easily calculate the internal energy, and all moments thereof. If we in addition to $\epsilon$ sample other operators like the magnetisation, we can use $P(\epsilon, T)$ to find thermal averages of arbitrary operators,

$$\langle O \rangle_T = \sum_{\epsilon} \langle O \rangle_{\epsilon} P(\epsilon, T).$$

Here $\langle O \rangle_{\epsilon}$ is the mean of $\hat{O}$ for a given value of $\epsilon$.

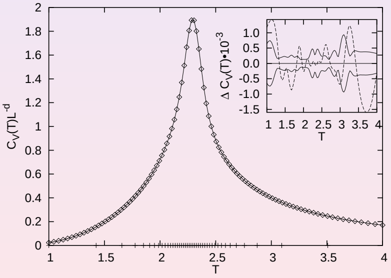
## Histogram Method, Ising Model



### Blocks: Ten

Lattice 32 $\times$ 32 system with PBC,and verified that within statistical error $F(T)$ agrees with the exact values. The small ticks on $T$ axis indicate $T$ values were simulations have been performed. In the inset the dashed curve shows the relative error of the estimated values, and the solid lines are $\pm$ an estimated statistical error.

## Histogram Method, Ising Model



**Blocks: Ten**

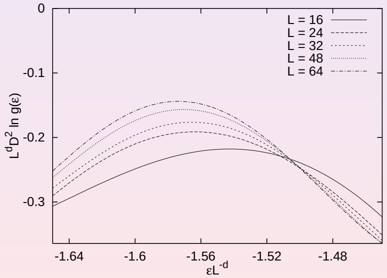
Lattice $32 \times 32$ system with PBC, and verified that within statistical error $C_V(T)$ agrees with the exact values. The small ticks on $T$ axis indicate $T$ values were simulations have been performed. In the inset the dashed curve shows the relative error of the estimated values, and the solid lines are $\pm$ an estimated statistical error.

## Histogram Method

All critical properties must be present in $w(\epsilon)$. In this section we will discuss the critical properties of the $Q = 3$ and $Q = 10$ Potts model. The first model has a continuous phase transition with $\alpha = 1/3$ and $\nu = 5/6$, i.e. $\alpha/\nu = 0.4$, the second model has a first order transition.

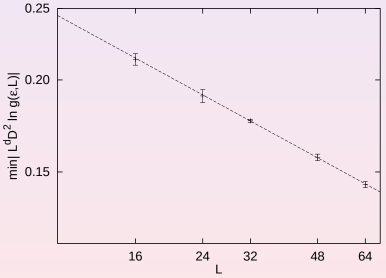
First we consider the $Q = 3$ model, for this model the goal is to determine the ratio $\alpha/\nu$ from $w(\epsilon, L)$. This can be done by considering how $\partial_\epsilon^2 \ln w(\epsilon, L)$ vanishes when approaching the critical energy $\epsilon_c$.

## Histogram Method, Potts Model $q = 3$.



We show $L^d \partial_\epsilon^2 \ln w(\epsilon, L)$ for different system sizes, and we can see that peak approaches zero with increasing system size. In the limit $L \to \infty$ this vanishes as $L^{-\alpha/\nu}$.

# Histogram Method, Potts Model



Plott of $\min|L^d \partial_\epsilon^2 \ln w(\epsilon, L)|$, i.e. the magnitude of the *peak* value for the curves in the previous figure, as a function of $L$. This shows finite size scaling of $L^d \min|\partial_\epsilon^2 w(\epsilon, L)|$ in a log-log plot, the dashed line has slope $-\alpha/\nu \approx -0.29$.

## Histogram Method

The dashed line in the previous figure has slope of $-\alpha/\nu \approx -0.29$; this is a significant deviation from the exact value $\alpha/\nu = 0.40$, however we feel that these results are sufficient to demonstrate that the critical properties, and in particular the exponents $\alpha$ and $\nu$ are contained in $w(\epsilon)$. There is clearly significant finite size effects in $\partial_\epsilon \ln w(\epsilon, L)$ also; including the factor $\partial_\epsilon \ln w(\epsilon)$ gives the "improved" value $\alpha/\nu \approx 0.35$, however this can not contribute in the $L \to \infty$ limit and we have therefore not included this factor.

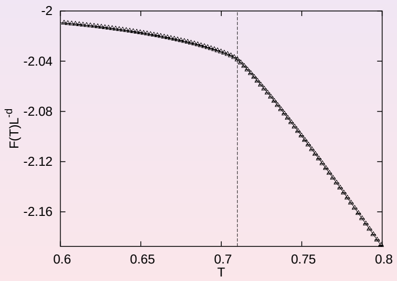
## Histogram Method

Finally the last figure is based on the second derivative of a sampled quantity; hence it will clearly be difficult to determine with high precision. In conclusion it is definitely *possible* to infer the ratio $\alpha/\nu$ from the properties of $w(\epsilon, L)$, but it is certainly not the most suitable way for high precision measurements. Finally we mention that the remaining critical exponents *cannot* be obtained from $w(\epsilon)$, their value is based on the explicit choice of fields to represent the critical state.

## Histogram Method

Altough all thermodynamic information about a phase transition is contained in $F(T)$, it is only for a first order transition, where $\partial_T F(T)$ is discontinous at $T_c$, that the phase transition stands out in $F(T)$. The next figure shows $F(T)$ for the strongly transition in the two dimensional $Q = 10$ Potts model; a discontinuity in $\partial_T F(T)$ at $T \approx 0.71$ is easily spotted.

## Histogram Method, Potts model $q = 10$



The free energy $F(T)$ for the $q = 10$ Potts model which has a strong first order transition. Although there is inevitably some finite size rounding, the cusp in this figure is quite clear.

## Topics for Week 10, February 26 - March 2

**Finite size scaling and Monte Carlo Renormalization Group**

- Brief survey of Monte Carlo renormalization group and the renormalization group idea
- Discussion of finite size scaling.
- Discussions of project 1

## Topics for Week 11, March 5-9

### Finite size scaling and Monte Carlo Renormalization Group

- Brief survey of Monte Carlo renormalization group and the renormalization group idea
- Discussions of project 1
- Introduction of new topic, variational quantum Monte Carlo
- Metropolis algorithm for quantal systems
- Simple systems, hydrogen and helium atoms

## Finite Size Scaling

We have defined

$$\tau = \frac{T - T_C}{T_C},$$

and assumed that the various thermodynamical quantities of interest scale as (for $\tau \to 0^-$)

$$M \sim |\tau|^{\beta},$$

$$C_V \sim |\tau|^{-\alpha},$$

$$\xi \sim |\tau|^{-\nu}$$

$$\chi \sim |\tau|^{-\gamma} \sim \xi^{\gamma/\nu}$$

and

$$B \sim M^{\delta} \sim |\tau|^{\beta\delta}$$

## Finite Size Scaling, Relations

For the Ising model in two dimensions we have $\beta = 1/8$, $\gamma = 7/4$, $\delta = 15$, $\nu = 1$ and $\alpha = 0$. We showed from simple dimensional analysis that we could obtain the following relations between critical exponents ($d$ is the dimensionality of the system)

$$\beta = \frac{\nu d}{\delta + 1} \qquad \text{Widom},$$

$$\gamma = \nu d \frac{\delta - 1}{\delta + 1} \qquad \text{Widom},$$

$$2 - \alpha = \nu d \qquad \text{Josephson},$$

$$\alpha + 2\beta + \gamma = 2 \qquad \text{Rushbrooke},$$

which are examples of some selected dimensonalities.

## Finite Size Scaling

The scaling hypothesis is used to derive relations between critical exponents from the fundamental definitions and relations in statistical physics.
It is an assumption based on the fact that thermodynamic functions are homogenous close to the critical temperature or more generally a given critical point.
A function $f(r)$ is homogeneous if for all values of a parameter $\lambda$ we have

$$f(\lambda r) = g(\lambda)f(r)$$

where the function $g(\lambda)$ is unspecified.

Such a function has the interesting feature that if we know the value at $r = r_0$ (one point) and we know $g(\lambda)$ then we know the function $f$ everywhere. *A simple change of scale* relates thus $f(r = r_0)$ to $f(r)$. In our particular case this means that if have the simulation results for one particular lattice, we can, if we know $g(\lambda)$, find the result for a larger lattice or even in the thermodynamic limit.

## Multivariable Relations

Can extend the homogeneous behavior to more variables

$$f(\lambda x, \lambda y) = \lambda^p f(x, y)$$

or

$$f(\lambda^a x, \lambda^b y) = \lambda f(x, y).$$

Look at Gibbs' free energy

$$G(T, B) \rightarrow G(\tau, B),$$

and scale it as

$$G(\lambda^{a_\tau} \tau, \lambda^{a_B} B) = \lambda G(\tau, B).$$

Can do the same with Helmoholtz' free energy

$$F(\lambda^{a_\tau} \tau, \lambda^{a_M} M) = \lambda F(\tau, M).$$

## Finite Size Scaling

Taking the derivative of $G$ wrt $B$ yields the magnetic moment $M$

$$\lambda^{a_B} M(\lambda^{a_\tau}\tau, \lambda^{a_B}B) = \lambda M(\tau, B).$$

Can derive two critical exponents, $\beta$ when $B = 0$ and $\tau \rightarrow 0^-$ and $\delta$ $\tau = 0$ and $B \rightarrow 0$. Collecting results yields Widom's relation

$$\gamma = \beta(\delta + 1)$$

We can repeat the same by taking derivatives wrt to $\tau$ and can then obtain Rushbrooke's inequality

$$\alpha + 2\beta + \gamma = 2.$$

## Widom's Relation

We start with

$$\lambda^{a_B} M(\lambda^{a_\tau}\tau, \lambda^{a_B} B) = \lambda M(\tau, B).$$

and set $B = 0$ which results in

$$\lambda^{a_B - 1} M(\lambda^{a_\tau}\tau, 0) = M(\tau, 0).$$

It is valid for all $\lambda$ which means it must also hold for $\lambda = (-1/\tau)^{1/a_\tau}$ resulting in

$$M(\tau, 0) = (-\tau)^{(1-a_B)/a_\tau} M(-1, 0)$$

and taking the limit $\tau \to 0^-$ we have

$$M(\tau, 0) \sim (-\tau)^{\beta}$$

and collecting we get

$$\beta = \frac{1 - a_B}{a_\tau}$$

## Finite Size Scaling

Repeat this exercise for $B \to 0$ and $\tau = 0$. We obtain

$$\lambda^{a_B - 1} M(0, \lambda^{a_B} B) = M(0, B).$$

and setting $\lambda = B^{-1/a_B}$ we have

$$\lambda^{a_B - 1} M(0, \lambda^{a_B} B) = M(0, B).$$

Using then

$$M(0, B) \sim B^{1/\delta},$$

we obtain

$$\delta = \frac{a_B}{1 - a_B}$$

resulting in

$$a_B = \frac{\delta}{\delta + 1}$$

and

$$a_\tau = \frac{1}{\beta(\delta + 1)}$$

## Finite Size Scaling

We can repeat this exercise by taking the second derivative of $G$ and obtain the susceptibility $\chi$. Studying it $B = 0$ and using again $\lambda = (-1/\tau)^{1/a_\tau}$ we have

$$\lambda^{2a_B}\chi(\lambda^{a_\tau}\tau, \lambda^{a_B}B) = \lambda\chi(\tau, B),$$

resulting in

$$\chi(\tau, 0) = (-\tau)^{(2a_B-1)/a_\tau}\chi(-1, 0),$$

and using that

$$\chi(\tau, 0) \sim (-\tau)^{-\gamma},$$

we arrive at

$$\gamma = \frac{2a_B - 1}{a_\tau}$$

or

$$\gamma = \beta(\delta - 1),$$

which is Widom's relation we derived from dimensional analysis.

## Reminder: Monte Carlo Keywords

Consider it is a numerical experiment

- Be able to generate random variables following a given probability distribution function (PDF). The starting point for any calculation is the derivation of random numbers based on the uniform distribution.
- Sampling rule for accepting a move, important algo Metropolis-Hastings
- Compute standard deviation and other expectation values
- Techniques for improving errors

## Metropolis-Hastings Algorithm

The equation for detailed balance with the acceptance probability $A$ is

$$\frac{A(\mu \rightarrow \nu)}{A(\nu \rightarrow \mu)} = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

is general and we could replace the Boltzmann distribution with other distributions as well. It specifies the ratio of pairs of acceptance probabilities, which leaves us with quite some room to manouvre.

- We give the largest of the two acceptance ratios the value 1 and adjust the other to satisfy the constraint.
- If $E_\mu < E_\nu$ then the larger of the two acceptance ratios is $A(\nu \rightarrow \mu)$ and we set to 1.
- Then we must have

$$A(\mu \rightarrow \nu) = \exp\left(-\beta(E_\nu - E_\mu)\right)$$

if $E_\nu - E_\mu > 0$ and 1 otherwise. And that is the **Metropolis-Hastings** algorithm.

## More general Definition

A Markov chain Monte Carlo method for the simulation of a distribution $p$ is any method producing an ergodic Markov chain of events $x$ whose stationary distribution is $p$.

- Generate an initial value $x^{(i)}$.
- Generate a trial value $y_t$ with probability $f(y_t|x^{(i)})$.
- Take a new value

$$x^{(i+1)} = \begin{cases} y_t & \text{with probability} = \rho(x^{(i)}, y_t) \\ x^{(i)} & \text{with probability} = 1 - \rho(x^{(i)}, y_t) \end{cases}$$

- We have defined

$$\rho(x, y) = \min\left\{\frac{p(y)f(x|y)}{p(x)f(y|x)}, 1\right\}.$$

  The distribution $f$ is often called the instrumental (we will relate it to the jumping of a walker) or proposal distribution while $\rho$ is the Metropolis-Hastings acceptance probability.

## Implementation

- Establish an initial state with some selected features to test.
- Do a random change of this initial state.
- Compute the Metropolis-Hastings acceptance probability $\rho$
- Compare $\rho$ with a random number $r$. If $r \leq \rho$ accept, else keep the old configuration.
- Compute the terms needed to obtain expectations values.
- Repeat the above steps in order to have as good statistics as possible.
- For a given number of MC cycles, compute then the final expectation values.

## Quantum Monte Carlo

Most quantum mechanical problems of interest in e.g., atomic, molecular, nuclear and solid state physics consist of a large number of interacting electrons and ions or nucleons. The total number of particles $N$ is usually sufficiently large that an exact solution cannot be found. Typically, the expectation value for a chosen hamiltonian for a system of $N$ particles is

$$\langle H \rangle =$$

$$\frac{\int d\mathbf{R}_1 d\mathbf{R}_2 \dots d\mathbf{R}_N \Psi^*(\mathbf{R_1}, \mathbf{R}_2, \dots, \mathbf{R}_N) H(\mathbf{R_1}, \mathbf{R}_2, \dots, \mathbf{R}_N) \Psi(\mathbf{R_1}, \mathbf{R}_2, \dots, \mathbf{R}_N)}{\int d\mathbf{R}_1 d\mathbf{R}_2 \dots d\mathbf{R}_N \Psi^*(\mathbf{R_1}, \mathbf{R}_2, \dots, \mathbf{R}_N) \Psi(\mathbf{R_1}, \mathbf{R}_2, \dots, \mathbf{R}_N)},$$

an in general intractable problem. an in general intractable problem.

This integral is actually the starting point in a Variational Monte Carlo calculation.

**Gaussian quadrature: Forget it!** given 10 particles and 10 mesh points for each degree of freedom and an ideal 1 Tflops machine (all operations take the same time), how long will it ta ke to compute the above integral? Lifetime of the universe $T \approx 4.7 \times 10^{17}$s.

## Quantum Monte Carlo

As an example from the nuclear many-body problem, we have Schrödinger's equation as a differential equation

$$\hat{H}\Psi(\mathbf{r}_1,..,\mathbf{r}_A,\alpha_1,..,\alpha_A) = E\Psi(\mathbf{r}_1,..,\mathbf{r}_A,\alpha_1,..,\alpha_A)$$

where

$$\mathbf{r}_1,..,\mathbf{r}_A,$$

are the coordinates and

$$\alpha_1,..,\alpha_A,$$

are sets of relevant quantum numbers such as spin and isospin for a system of $A$ nucleons ($A = N + Z$, $N$ being the number of neutrons and $Z$ the number of protons).

## Quantum Monte Carlo

There are

$$2^A \times \left( \begin{array}{c} A \\ Z \end{array} \right)$$

coupled second-order differential equations in $3A$ dimensions.
For a nucleus like $^{10}$Be this number is **215040**. This is a truely challenging many-body problem.

Methods like partial differential equations can at most be used for 2-3 particles.

## Quantum Many-particle(body) Methods

1. Monte-Carlo methods
2. Renormalization group (RG) methods, in particular density matrix RG
3. Large-scale diagonalization (Iterative methods, Lanczo's method, dimensionalities $10^{10}$ states)
4. Coupled cluster theory, favoured method in quantum chemistry, molecular and atomic physics. Applications to ab initio calculations in nuclear physics as well for large nuclei.
5. Perturbative many-body methods
6. Green's function methods
7. Density functional theory/Mean-field theory.....

The physics of the system hints at which many-body methods to use. For systems with strong correlations among the constituents, item 5 and 7 are ruled out.

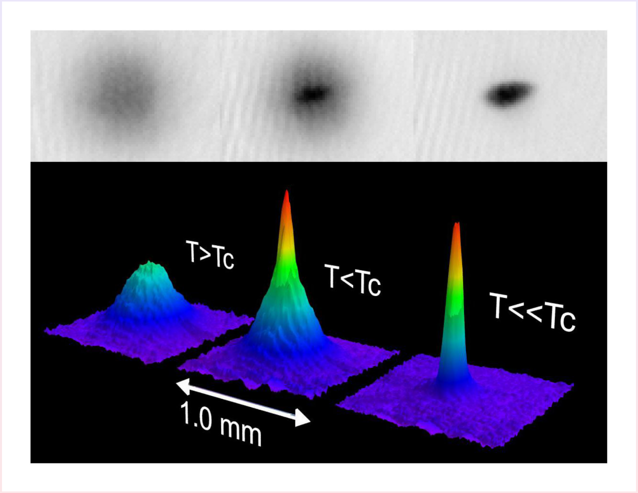
## Pros and Cons of Monte Carlo

- Is physically intuitive.
- Allows one to study systems with many degrees of freedom. Diffusion Monte Carlo (DMC) and Green's function Monte Carlo (GFMC) yield in principle the exact solution to Schrödinger's equation.
- Variational Monte Carlo (VMC) is easy to implement but needs a reliable trial wave function, can be difficult to obtain.
- DMC/GFMC for fermions (spin with half-integer values, electrons, baryons, neutrinos, quarks) has a sign problem. Nature prefers an anti-symmetric wave function. PDF in this case given distribution of random walkers ($p \geq 0$).
- The solution has a statistical error, which can be large. Hard to compete with light systems in quantum chemistry (less than 100 electrons).
- There is a limit for how large systems one can study, DMC needs a huge number of random walkers in order to achieve stable results.
- Obtain only the lowest-lying states with a given symmetry. Difficult to get excited states.

## Where and why do we use Monte Carlo Methods in Quantum Physics

- Quantum systems with many particles at finite temperature: Path Integral Monte Carlo with applications to dense matter and quantum liquids (phase transitions from normal fluid to superfluid). Strong correlations.
- Bose-Einstein condensation of dilute gases, method transition from non-linear PDE to Diffusion Monte Carlo as density increases.
- Light atoms, molecules and nuclei. In quantum chemistry, atomic and molecular physics however precision not good enough. Applications in nuclear physics for systems with less than 12 nucleons.
- Lattice Quantum-Chromo Dynamics. Impossible to solve without MC calculations.
- Simulations of systems in solid state physics, from semiconductors to spin systems. Many electrons active and possibly strong correlations.

## Bose-Einstein Condensation of atoms, thousands of Atoms in one State, Project 2

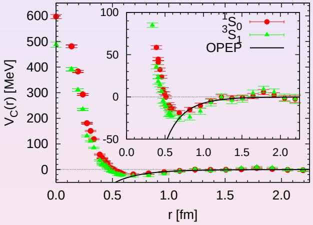


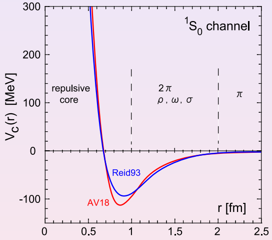
## Lattice QCD

### Brief Description

Analytic or perturbative solutions in QCD are hard or impossible due to the highly nonlinear nature of the strong force. The formulation of QCD on a discrete rather than continuous space-time naturally introduces a momentum cut off at the order 1/a, which regularizes the theory. As a result lattice QCD is mathematically well-defined. Most importantly, lattice QCD provides the framework for investigation of non-perturbative phenomena such as confinement and quark-gluon plasma formation, which are intractable by means of analytic field theories.

## Lattice QCD, Ishii *et al*, nucl-th/0611096





The nucleon-nucleon interaction, Phenomenology vs Lattice calculations.

## Quantum Monte Carlo and Schrödinger's equation

For one-body problems (one dimension)

$$-\frac{\hbar^2}{2m}\nabla^2\Psi(x,t) + V(x,t)\Psi(x,t) = \imath\hbar\frac{\partial\Psi(x,t)}{\partial t},$$

Quantum mechanical probablity distribution function (PDF)

$$P(x,t) = \Psi(x,t)^*\Psi(x,t)$$

$$P(x,t)dx = \Psi(x,t)^*\Psi(x,t)dx$$

Interpretation: probability of finding the system in a region between $x$ and $x + dx$. Always real

$$\Psi(x,t) = R(x,t) + \imath I(x,t)$$

yielding

$$\Psi(x,t)^*\Psi(x,t) = (R - \imath I)(R + \imath I) = R^2 + I^2$$

Variational Monte Carlo uses only $P(x,t)$!!

## Quantum Monte Carlo and Schrödinger's equation

### Petit digression

Choose $\tau = it/\hbar$.
The time-dependent (1-dim) Schrödinger equation becomes then

$$\frac{\partial \Psi(x,\tau)}{\partial \tau} = \frac{\hbar^2}{2m}\frac{\partial^2 \Psi(x,\tau)}{\partial x^2} - V(x,\tau)\Psi(x,\tau).$$

With $V = 0$ we have a diffusion equation in complex time with diffusion constant

$$D = \frac{\hbar^2}{2m}.$$

Used in diffusion Monte Carlo calculations. The wave function is given by the distribution of random walkers. Leads to problems if the wave function is anti-symmetric.

## Quantum Monte Carlo and Schrödinger's equation

Conditions which $\Psi$ has to satisfy:

1. Normalization

$$\int_{-\infty}^{\infty} P(x,t)dx = \int_{-\infty}^{\infty} \Psi(x,t)^* \Psi(x,t)dx = 1$$

   meaning that

$$\int_{-\infty}^{\infty} \Psi(x,t)^* \Psi(x,t)dx < \infty$$

2. $\Psi(x,t)$ and $\partial\Psi(x,t)/\partial x$ must be finite
3. $\Psi(x,t)$ and $\partial\Psi(x,t)/\partial x$ must be continuous.
4. $\Psi(x,t)$ and $\partial\Psi(x,t)/\partial x$ must be single valued

Square integrable functions.

## First Postulate

Any physical quantity $A(\vec{r},\vec{p})$ which depends on position $\vec{r}$ and momentum $\vec{p}$ has a corresponding quantum mechanical operator by replacing $\vec{p}$ $-i\hbar\vec{\bigtriangledown}$, yielding the quantum mechanical operator

$$\widehat{\mathbf{A}} = A(\vec{r}, -i\hbar\vec{\bigtriangledown}).$$

| Quantity | Classical definition | QM operator |
|---|---|---|
| Position | $\vec{r}$ | $\widehat{\tilde{\mathbf{r}}} = \vec{r}$ |
| Momentum | $\vec{p}$ | $\widehat{\tilde{\mathbf{p}}} = -i\hbar\vec{\bigtriangledown}$ |
| Orbital momentum | $\vec{L} = \vec{r} \times \vec{p}$ | $\widehat{\tilde{\mathbf{L}}} = \vec{r} \times (-i\hbar\vec{\bigtriangledown})$ |
| Kinetic energy | $T = (\vec{p})^2/2m$ | $\widehat{\mathbf{T}} = -(\hbar^2/2m)(\vec{\bigtriangledown})^2$ |
| Total energy | $H = (p^2/2m) + V(\vec{r})$ | $\widehat{\mathbf{H}} = -(\hbar^2/2m)(\vec{\bigtriangledown})^2 + V(\vec{r})$ |

## Second Postulate

*The only possible outcome of an ideal measurement of the physical quantity A are the eigenvalues of the corresponding quantum mechanical operator* $\widehat{\mathbf{A}}$.

$$\widehat{\mathbf{A}}\psi_\nu = a_\nu \psi_\nu,$$

resulting in the eigenvalues $a_1, a_2, a_3, \cdots$ as the only outcomes of a measurement.

The corresponding eigenstates $\psi_1, \psi_2, \psi_3 \cdots$ contain all relevant information about the system.

## Third Postulate

Assume $\Phi$ is a linear combination of the eigenfunctions $\psi_\nu$ for $\widehat{\mathbf{A}}$,

$$\Phi = c_1\psi_1 + c_2\psi_2 + \cdots = \sum_\nu c_\nu\psi_\nu.$$

The eigenfunctions are orthogonal and we get

$$c_\nu = \int (\Phi)^*\psi_\nu d\tau.$$

From this we can formulate the third postulate:

*When the eigenfunction is $\Phi$, the probability of obtaining the value $a_\nu$ as the outcome of a measurement of the physical quantity A is given by $|c_\nu|^2$ and $\psi_\nu$ is an eigenfunction of $\widehat{\mathbf{A}}$ with eigenvalue $a_\nu$.*

## Third Postulate

As a consequence one can show that:
*when a quantal system is in the state $\Phi$, the mean value or expectation value of a physical quantity $A(\vec{r},\vec{p})$ is given by*

$$\langle A\rangle = \int (\Phi)^{*}\widehat{\mathbf{A}}(\vec{r}, -i\hbar\vec{\bigtriangledown})\Phi d\tau.$$

We have assumed that $\Phi$ has been normalized, viz., $\int(\Phi)^{*}\Phi d\tau = 1$. Else

$$\langle A\rangle = \frac{\int(\Phi)^{*}\widehat{\mathbf{A}}\Phi d\tau}{\int(\Phi)^{*}\Phi d\tau}.$$

## Fourth Postulate

The time development of of a quantal system is given by

$$i\hbar\frac{\partial\Psi}{\partial t} = \widehat{\mathbf{H}}\Psi,$$

with $\widehat{\mathbf{H}}$ the quantal Hamiltonian operator for the system.

## Quantum Monte Carlo

Given a hamiltonian $H$ and a trial wave function $\Psi_T$, the variational principle states that the expectation value of $\langle H \rangle$, defined through

$$E[H] = \langle H \rangle = \frac{\int d\mathbf{R}\Psi_T^*(\mathbf{R})H(\mathbf{R})\Psi_T(\mathbf{R})}{\int d\mathbf{R}\Psi_T^*(\mathbf{R})\Psi_T(\mathbf{R})},$$

is an upper bound to the ground state energy $E_0$ of the hamiltonian $H$, that is

$$E_0 \leq \langle H \rangle.$$

In general, the integrals involved in the calculation of various expectation values are multi-dimensional ones. Traditional integration methods such as the Gauss-Legendre will not be adequate for say the computation of the energy of a many-body system.

## Quantum Monte Carlo

The trial wave function can be expanded in the eigenstates of the hamiltonian since they form a complete set, viz.,

$$\Psi_T(\mathbf{R}) = \sum_i a_i \Psi_i(\mathbf{R}),$$

and assuming the set of eigenfunctions to be normalized one obtains

$$\frac{\sum_{nm} a_m^* a_n \int d\mathbf{R} \Psi_m^*(\mathbf{R}) H(\mathbf{R}) \Psi_n(\mathbf{R})}{\sum_{nm} a_m^* a_n \int d\mathbf{R} \Psi_m^*(\mathbf{R}) \Psi_n(\mathbf{R})} = \frac{\sum_n a_n^2 E_n}{\sum_n a_n^2} \geq E_0,$$

where we used that $H(\mathbf{R})\Psi_n(\mathbf{R}) = E_n\Psi_n(\mathbf{R})$. In general, the integrals involved in the calculation of various expectation values are multi-dimensional ones. The variational principle yields the lowest state of a given symmetry.

## Quantum Monte Carlo

In most cases, a wave function has only small values in large parts of configuration space, and a straightforward procedure which uses homogenously distributed random points in configuration space will most likely lead to poor results. This may suggest that some kind of importance sampling combined with e.g., the Metropolis algorithm may be a more efficient way of obtaining the ground state energy. The hope is then that those regions of configurations space where the wave function assumes appreciable values are sampled more efficiently.

The tedious part in a VMC calculation is the search for the variational minimum. A good knowledge of the system is required in order to carry out reasonable VMC calculations. This is not always the case, and often VMC calculations serve rather as the starting point for so-called diffusion Monte Carlo calculations (DMC). DMC is a way of solving exactly the many-body Schrödinger equation by means of a stochastic procedure. A good guess on the binding energy and its wave function is however necessary. A carefully performed VMC calculation can aid in this context.

## Quantum Monte Carlo

- Construct first a trial wave function $\psi_T^\alpha(\mathbf{R})$, for a many-body system consisting of $N$ particles located at positions $\mathbf{R} = (\mathbf{R_1}, \dots, \mathbf{R_N})$. The trial wave function depends on $\alpha$ variational parameters $\alpha = (\alpha_{\mathbf{1}}, \dots, \alpha_{\mathbf{N}})$.
- Then we evaluate the expectation value of the hamiltonian $H$

$$E[H] = \langle H \rangle = \frac{\int d\mathbf{R}\Psi^*_{T_\alpha}(\mathbf{R})H(\mathbf{R})\Psi_{T_\alpha}(\mathbf{R})}{\int d\mathbf{R}\Psi^*_{T_\alpha}(\mathbf{R})\Psi_{T_\alpha}(\mathbf{R})}.$$

- Thereafter we vary $\alpha$ according to some minimization algorithm and return to the first step.

# Quantum Monte Carlo

Choose a trial wave function $\psi_T(\mathbf{R})$.

$$P(\mathbf{R}) = \frac{|\psi_T(\mathbf{R})|^2}{\int |\psi_T(\mathbf{R})|^2 d\mathbf{R}}.$$

This is our new probability distribution function (PDF). The approximation to the expectation value of the Hamiltonian is now

$$E[H] \approx \frac{\int d\mathbf{R}\Psi_T^*(\mathbf{R})H(\mathbf{R})\Psi_T(\mathbf{R})}{\int d\mathbf{R}\Psi_T^*(\mathbf{R})\Psi_T(\mathbf{R})}.$$

Define a new quantity

$$E_L(\mathbf{R}) = \frac{1}{\psi_T(\mathbf{R})}H\psi_T(\mathbf{R}),$$

called the local energy, which, together with our trial PDF yields

$$E[H] = \langle H \rangle \approx \int P(\mathbf{R})E_L(\mathbf{R})d\mathbf{R} \approx \frac{1}{N}\sum_{i=1}^{N} P(\mathbf{R_i})E_L(\mathbf{R_i})$$

with $N$ being the number of Monte Carlo samples.

## Quantum Monte Carlo

Algo:

- Initialisation: Fix the number of Monte Carlo steps. Choose an initial $\mathbf{R}$ and variational parameters $\alpha$ and calculate $\left|\psi_T^\alpha(\mathbf{R})\right|^2$.
- Initialise the energy and the variance and start the Monte Carlo calculation (thermalize)
    1. Calculate a trial position $\mathbf{R}_p = \mathbf{R} + r * step$ where $r$ is a random variable $r \in [0, 1]$.
    2. Metropolis algorithm to accept or reject this move
    $$w = P(\mathbf{R}_p)/P(\mathbf{R}).$$
    3. If the step is accepted, then we set $\mathbf{R} = \mathbf{R}_p$. Update averages
- Finish and compute final averages.

Observe that the jumping in space is governed by the variable *step*. Called brute-force sampling. Need importance sampling.

## Quantum Monte Carlo

The radial Schrödinger equation for the hydrogen atom can be written as

$$-\frac{\hbar^2}{2m}\frac{\partial^2 u(r)}{\partial r^2} - \left(\frac{ke^2}{r} - \frac{\hbar^2 l(l+1)}{2mr^2}\right)u(r) = Eu(r),$$

or with dimensionless variables

$$-\frac{1}{2}\frac{\partial^2 u(\rho)}{\partial \rho^2} - \frac{u(\rho)}{\rho} + \frac{l(l+1)}{2\rho^2}u(\rho) - \lambda u(\rho) = 0,$$

with the hamiltonian

$$H = -\frac{1}{2}\frac{\partial^2}{\partial \rho^2} - \frac{1}{\rho} + \frac{l(l+1)}{2\rho^2}.$$

Use variational parameter $\alpha$ in the trial wave function

$$u_T^{\alpha}(\rho) = \alpha\rho e^{-\alpha\rho}.$$

## Quantum Monte Carlo

Inserting this wave function into the expression for the local energy $E_L$ gives

$$E_L(\rho) = -\frac{1}{\rho} - \frac{\alpha}{2}\left(\alpha - \frac{2}{\rho}\right).$$

| $\alpha$ | $\langle H\rangle$ | $\sigma^2$ | $\sigma/\sqrt{N}$ |
|---:|---:|---:|---:|
| 7.00000E-01 | -4.57759E-01 | 4.51201E-02 | 6.71715E-04 |
| 8.00000E-01 | -4.81461E-01 | 3.05736E-02 | 5.52934E-04 |
| 9.00000E-01 | -4.95899E-01 | 8.20497E-03 | 2.86443E-04 |
| 1.00000E-00 | -5.00000E-01 | 0.00000E+00 | 0.00000E+00 |
| 1.10000E+00 | -4.93738E-01 | 1.16989E-02 | 3.42036E-04 |
| 1.20000E+00 | -4.75563E-01 | 8.85899E-02 | 9.41222E-04 |
| 1.30000E+00 | -4.54341E-01 | 1.45171E-01 | 1.20487E-03 |

## Quantum Monte Carlo

We note that at $\alpha = 1$ we obtain the exact result, and the variance is zero, as it should. The reason is that we then have the exact wave function, and the action of the hamiltionan on the wave function

$$H\psi = \mathrm{constant} \times \psi,$$

yields just a constant. The integral which defines various expectation values involving moments of the hamiltonian becomes then

$$\langle H^n \rangle = \frac{\int d\mathbf{R}\Psi^*_T(\mathbf{R})H^n(\mathbf{R})\Psi_T(\mathbf{R})}{\int d\mathbf{R}\Psi^*_T(\mathbf{R})\Psi_T(\mathbf{R})} = \mathrm{constant} \times \frac{\int d\mathbf{R}\Psi^*_T(\mathbf{R})\Psi_T(\mathbf{R})}{\int d\mathbf{R}\Psi^*_T(\mathbf{R})\Psi_T(\mathbf{R})} = \mathrm{constant}.$$

**This gives an important information: the exact wave function leads to zero variance!** Variation is then performed by minimizing both the energy and the variance.

## Quantum Monte Carlo

The helium atom consists of two electrons and a nucleus with charge $Z = 2$. The contribution to the potential energy due to the attraction from the nucleus is

$$-\frac{2ke^2}{r_1} - \frac{2ke^2}{r_2},$$

and if we add the repulsion arising from the two interacting electrons, we obtain the potential energy

$$V(r_1, r_2) = -\frac{2ke^2}{r_1} - \frac{2ke^2}{r_2} + \frac{ke^2}{r_{12}},$$

with the electrons separated at a distance $r_{12} = |\mathbf{r}_1 - \mathbf{r}_2|$.

## Quantum Monte Carlo

The hamiltonian becomes then

$$\widehat{\mathbf{H}} = -\frac{\hbar^2\nabla_1^2}{2m} - \frac{\hbar^2\nabla_2^2}{2m} - \frac{2ke^2}{r_1} - \frac{2ke^2}{r_2} + \frac{ke^2}{r_{12}},$$

and Schrödingers equation reads

$$\widehat{\mathbf{H}}\psi = E\psi.$$

All observables are evaluated with respect to the probability distribution

$$P(\mathbf{R}) = \frac{|\psi_T(\mathbf{R})|^2}{\int |\psi_T(\mathbf{R})|^2 \, d\mathbf{R}}.$$

generated by the trial wave function. The trial wave function must approximate an exact eigenstate in order that accurate results are to be obtained. Improved trial wave functions also improve the importance sampling, reducing the cost of obtaining a certain statistical accuracy.

## Quantum Monte Carlo

Choice of trial wave function for Helium: Assume $r_1 \rightarrow 0$.

$$E_L(\mathbf{R}) = \frac{1}{\psi_T(\mathbf{R})} H\psi_T(\mathbf{R}) = \frac{1}{\psi_T(\mathbf{R})}\left(-\frac{1}{2}\nabla_1^2 - \frac{Z}{r_1}\right)\psi_T(\mathbf{R}) + \text{finite terms}.$$

$$E_L(R) = \frac{1}{\mathcal{R}_T(r_1)}\left(-\frac{1}{2}\frac{d^2}{dr_1^2} - \frac{1}{r_1}\frac{d}{dr_1} - \frac{Z}{r_1}\right)\mathcal{R}_T(r_1) + \text{finite terms}$$

For small values of $r_1$, the terms which dominate are

$$\lim_{r_1\rightarrow 0} E_L(R) = \frac{1}{\mathcal{R}_T(r_1)}\left(-\frac{1}{r_1}\frac{d}{dr_1} - \frac{Z}{r_1}\right)\mathcal{R}_T(r_1),$$

since the second derivative does not diverge due to the finiteness of $\Psi$ at the origin.

## Quantum Monte Carlo

This results in

$$\frac{1}{\mathcal{R}_T(r_1)}\frac{d\mathcal{R}_T(r_1)}{dr_1} = -Z,$$

and

$$\mathcal{R}_T(r_1) \propto e^{-Zr_1}.$$

A similar condition applies to electron 2 as well. For orbital momenta $l > 0$ we have

$$\frac{1}{\mathcal{R}_T(r)}\frac{d\mathcal{R}_T(r)}{dr} = -\frac{Z}{l+1}.$$

Similalry, studying the case $r_{12} \rightarrow 0$ we can write a possible trial wave function as

$$\psi_T(\mathbf{R}) = e^{-\alpha(r_1+r_2)}e^{r_{12}/2}.$$

The last equation can be generalized to

$$\psi_T(\mathbf{R}) = \phi(\mathbf{r}_1)\phi(\mathbf{r}_2)\dots\phi(\mathbf{r}_N)\prod_{i<j} f(r_{ij}),$$

for a system with $N$ electrons or particles.

## Topics for Week 12, March 12-16

**Quantum Monte Carlo Methods**

- Reminder from last week
- Variational Monte Carlo
- Discussions of project 2
- Importance sampling

## Project 2

Here we limit ourselves to a mere technical discussion of the physics of Bose-Einstein condensation (BEC). That is we focus on the technical aspects we need to deal with in connection with this project. We will discuss in more detail later the physics behind BEC.

A key feature of the trapped alkali and atomic hydrogen systems is that they are dilute. The characteristic dimensions of a typical trap for $^{87}$Rb is $a_{h0} = (\hbar/m\omega_\perp)^{\frac{1}{2}} = 1 - 2 \times 10^4$ Å . The interaction between $^{87}$Rb atoms can be well represented by its s-wave scattering length, $a_{Rb}$. This scattering length lies in the range $85 < a_{Rb} < 140a_0$ where $a_0 = 0.5292$ Å is the Bohr radius. The definite value $a_{Rb} = 100a_0$ is usually selected and for calculations the definite ratio of atom size to trap size $a_{Rb}/a_{h0} = 4.33 \times 10^{-3}$ is usually chosen. A typical $^{87}$Rb atom density in the trap is $n \simeq 10^{12} - 10^{14}$ atoms/cm$^3$ giving an inter-atom spacing $\ell \simeq 10^4$ Å. Thus the effective atom size is small compared to both the trap size and the inter-atom spacing, the condition for diluteness (i.e., $na^3_{Rb} \simeq 10^{-6}$ where $n = N/V$ is the number density).

# Project 2

The aim of this project is to use Variational Monte Carlo (VMC) and Diffusion Monte Carlo (DMC) methods and evaluate the ground state energy of a trapped, hard sphere Bose gas for different numbers of particles with a specific trial wave function.
This wave function is used to study the sensitivity of condensate and non-condensate properties to the hard sphere radius and the number of particles. The trap we will use is a spherical (S) or an elliptical (E) harmonic trap in three dimensions given by

$$V_{ext}(\mathbf{r}) = \begin{cases} \frac{1}{2}m\omega_{ho}^2 r^2 & (S) \\ \frac{1}{2}m[\omega_{ho}^2(x^2+y^2)+\omega_z^2 z^2] & (E) \end{cases} \tag{84}$$

## Project 2

The two-body interaction is

$$H = \sum_{i}^{N} \left( \frac{-\hbar^2}{2m} \nabla_i^2 + V_{ext}(\mathbf{r}_i) \right) + \sum_{i<j}^{N} V_{int}(\mathbf{r}_i, \mathbf{r}_j), \tag{85}$$

Here $\omega_{ho}^2$ defines the trap potential strength. In the case of the elliptical trap, $V_{ext}(x, y, z)$, $\omega_{ho} = \omega_{\perp}$ is the trap frequency in the perpendicular or $xy$ plane and $\omega_z$ the frequency in the $z$ direction. The mean square vibrational amplitude of a single boson at $T = 0K$ in the trap (84) is $< x^2 >= (\hbar/2m\omega_{ho})$ so that $a_{ho} \equiv (\hbar/m\omega_{ho})^{\frac{1}{2}}$ defines the characteristic length of the trap. The ratio of the frequencies is denoted $\lambda = \omega_z/\omega_{\perp}$ leading to a ratio of the trap lengths $(a_{\perp}/a_z) = (\omega_z/\omega_{\perp})^{\frac{1}{2}} = \sqrt{\lambda}$.

## Project 2

We represent the inter boson interaction by a pairwise, hard core potential

$$V_{int}(|\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} \infty & |\mathbf{r}_i - \mathbf{r}_j| \leq a \\ 0 & |\mathbf{r}_i - \mathbf{r}_j| > a \end{cases} \tag{86}$$

where $a$ is the hard core diameter of the bosons. Clearly, $V_{int}(|\mathbf{r}_i - \mathbf{r}_j|)$ is zero if the bosons are separated by a distance $|\mathbf{r}_i - \mathbf{r}_j|$ greater than $a$ but infinite if they attempt to come within a distance $|\mathbf{r}_i - \mathbf{r}_j| \leq a$.

## Project 2

Our trial wave function for the ground state with $N$ atoms is given by

$$\Psi_T(\mathbf{R}) = \Psi_T(\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_N, \alpha, \beta) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{i<j} f(a, |\mathbf{r}_i - \mathbf{r}_j|), \tag{87}$$

where $\alpha$ and $\beta$ are variational parameters. The single-particle wave function is proportional to the harmonic oscillator function for the ground state, i.e.,

$$g(\alpha, \beta, \mathbf{r}_i) = \exp\left[-\alpha(x_i^2 + y_i^2 + \beta z_i^2)\right]. \tag{88}$$

For spherical traps we have $\beta = 1$ and for non-interacting bosons ($a = 0$) we have $\alpha = 1/2a_{ho}^2$. The correlation wave function is

$$f(a, |\mathbf{r}_i - \mathbf{r}_j|) = \begin{cases} 0 & |\mathbf{r}_i - \mathbf{r}_j| \le a \\ (1 - \frac{a}{|\mathbf{r}_i - \mathbf{r}_j|}) & |\mathbf{r}_i - \mathbf{r}_j| > a. \end{cases} \tag{89}$$

## Project 2

The first problem is to find analytic expressions for the local energy

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} H\Psi_T(\mathbf{R}), \tag{90}$$

for the above trial wave function of Eq. (87). You will also have to compute the analytic expression for the drift force to be used in the importance sampling (1d)

$$F = \frac{2\nabla\Psi_T}{\Psi_T}. \tag{91}$$

# How to find the analytic expression?

Our trial wave function is

$$\Psi_T(\mathbf{R}) = \Psi_T(\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_N, \alpha, \beta) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{i<j} f(a, |\mathbf{r}_i - \mathbf{r}_j|),$$

The tricky part is to find an analytic expressions for the derivative of the trial wave function

$$\frac{1}{\Psi_T(\mathbf{R})} \sum_i^N \nabla_i^2 \Psi_T(\mathbf{R}),$$

for the above trial wave function of Eq. (87). We need also to compute the drift force

$$F = \frac{2\nabla\Psi_T}{\Psi_T}.$$

## How to find the analytic expression?

We rewrite

$$\Psi_T(\mathbf{R}) = \Psi_T(\mathbf{r}_1, \mathbf{r}_2, \dots \mathbf{r}_N, \alpha, \beta) = \prod_i g(\alpha, \beta, \mathbf{r}_i) \prod_{i<j} f(a, |\mathbf{r}_i - \mathbf{r}_j|),$$

as

$$\Psi_T(\mathbf{R}) = \prod_i g(\alpha, \beta, \mathbf{r}_i) e^{\sum_{i<j} u(r_{ij})}$$

where we have defined $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ and

$$f(r_{ij}) = e^{\sum_{i<j} u(r_{ij})},$$

and in our case

$$g(\alpha, \beta, \mathbf{r}_i) = e^{-\alpha(x_i^2+y_i^2+z_i^2)} = \phi(\mathbf{r}_i).$$

## How to find the analytic expression?

The first derivative becomes

$$\nabla_k \Psi_T(\mathbf{R}) = \nabla_k \phi(\mathbf{r}_k) \left[\prod_{i\neq k} \phi(\mathbf{r}_i)\right] e^{\sum_{i<j} u(r_{ij})} + \prod_i \phi(\mathbf{r}_i) e^{\sum_{i<j} u(r_{ij})} \sum_{j\neq k} \nabla_k u(r_{ij})$$

For the second derivative, see development on blackboard (we assume that $u$ is a function of $r_{ij}$ only.)

## How to find the analytic expression?

The final expression is

$$\frac{1}{\Psi_T(\mathbf{R})}\nabla_k^2\Psi_T(\mathbf{R}) = \frac{\nabla_k^2\phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)} + \frac{\nabla_k\phi(\mathbf{r}_k)}{\phi(\mathbf{r}_k)}\left(\sum_{j\neq k}\frac{\mathbf{r}_k}{r_k}u'(r_{ij})\right) +$$

$$\sum_{ij\neq k}\frac{(\mathbf{r}_k-\mathbf{r}_i)(\mathbf{r}_k-\mathbf{r}_j)}{r_{ki}r_{kj}}u'(r_{ki})u'(r_{kj}) + \sum_{j\neq k}\left(u''(r_{kj}) + \frac{2}{r_{kj}}u'(r_{kj})\right)$$

You need to get the analytic expression for this using the harmonic oscillator wave functions and the correlation term defined in the project.

## Project 2

Write a Variational Monte Carlo program which uses standard Metropolis sampling and compute the ground state energy of a spherical harmonic oscillator ($\beta = 1$) with no interaction. Use natural units and make an analysis of your calculations using both the analytic expression for the local energy and a numerical calculation of the kinetic energy using numerical derivation. Compare the CPU time difference. You should also parallelize your code. The only variational parameter is $\alpha$. Perform these calculations for $N = 10$, 100 and 500 atoms. Compare your results with the exact answer. (what is the exact answer?)

## Project 2

We turn now to the elliptic trap with a hard core interaction. We fix $a/a_{ho} = 0.0043$. Introduce lengths in units of $a_{ho}$, $r \rightarrow r/a_{ho}$ and energy in units of $\hbar\omega_{ho}$. Show then that the original Hamiltonian can be rewritten as

$$H = \sum_{i=1}^{N} \frac{1}{2}\left(-\nabla_i^2 + x_i^2 + y_i^2 + \gamma^2 z_i^2\right) + \sum_{i<j} V_{int}(|\mathbf{r}_i - \mathbf{r}_j|). \qquad (92)$$

What is the expression for $\gamma$? Choose the initial value for $\beta = \gamma = 2.82843$ and set up a VMC program which computes the ground state energy using the trial wave function of Eq. (87). using only $\alpha$ as variational parameter. Use standard Metropolis sampling and vary the parameter $\alpha$ in order to find a minimum. Perform the calculations for $N = 10, 100$ and $N = 500$ and compare your results to those from the ideal case in the previous exercise.

## Project 2

We repeat exercise 1c), but now we replace the brute force Metropolis algorithm with importance sampling based on the Fokker-Planck and the Langevin equations. Discuss your results and comment on eventual differences between importance sampling and brute force sampling.
In the statistical analysis of your results you should use for example the blocking technique from Project 1.

If the parallel cluster setup works (hopefully!), you should also parallelize your code in much the same fashion as you did with project 1.

## Importance Sampling

We need to replace the brute force Metropolis algorithm with a walk in coordinate space biased by the trial wave function. This approach is based on the Fokker-Planck equation and the Langevin equation for generating a trajectory in coordinate space.
For a diffusion process characterized by a time-dependent probability density $P(x,t)$ in one dimension the Fokker-Planck equation reads (for one particle/walker)

$$\frac{\partial P}{\partial t} = D\frac{\partial P}{\partial x}\left(\frac{\partial P}{\partial x} - F\right)P(x,t),$$

where $F$ is a drift term and $D$ is the diffusion coefficient. The drift term is

$$F = 2\frac{1}{\Psi_T}\nabla\Psi_T$$

where $\Psi_T$ is our trial wave function. The new positions in coordinate space are given as the solutions of the Langevin equation using Euler's method, namely, we go from the Langevin equation

$$\frac{\partial x(t)}{\partial t} = DF(x(t)) + \eta,$$

with $\eta$ a random variable, yielding a new position

$$y = x + DF(x)\Delta t + \xi,$$

where $\xi$ is gaussian random variable and $\Delta t$ is a chosen time step.

## Importance Sampling

The Fokker-Planck equation yields a transition probability given by the Green's function

$$G(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-(y-x-D\Delta t F(x))^2/4D\Delta t\right)$$

which in turn means that our brute force Metropolis algorithm

$$A(y,x) = \min(1, q(y,x))),$$

with $q(y,x) = |\Psi_T(y)|^2/|\Psi_T(x)|^2$ is now replaced by

$$q(y,x) = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2}$$

## Topics for Week 13, March 26-30

**Finite size scaling and Monte Carlo Renormalization Group**

- Brief reminder from last week
- Discussion of project 2
- Presentation of diffusion Monte Carlo
- Summary of part 1.

## Digression: Diffusion from Markov Chain

A Markov process allows in principle for a microscopic description of Brownian motion. As with the random walk studied in the previous section, we consider a particle which moves along the $x$-axis in the form of a series of jumps with step length $\Delta x = l$. Time and space are discretized and the subsequent moves are statistically indenpendent, i.e., the new move depends only on the previous step and not on the results from earlier trials. We start at a position $x = jl = j\Delta x$ and move to a new position $x = i\Delta x$ during a step $\Delta t = \epsilon$, where $i \geq 0$ and $j \geq 0$ are integers. The original probability distribution function (PDF) of the particles is given by $w_i(t = 0)$ where $i$ refers to a specific position on a grid, with $i = 0$ representing $x = 0$. The function $w_i(t = 0)$ is now the discretized version of $w(x, t)$. We can regard the discretized PDF as a vector.

## Digression: Diffusion from Markov Chain

For the Markov process we have a transition probability from a position $x = jl$ to a position $x = il$ given by

$$W_{ij}(\epsilon) = W(il - jl, \epsilon) = \begin{cases} \frac{1}{2} & |i-j| = 1 \\ 0 & \text{else} \end{cases}$$

We call $W_{ij}$ for the transition probability and we can represent it, see below, as a matrix. Our new PDF $w_i(t = \epsilon)$ is now related to the PDF at $t = 0$ through the relation

$$w_i(t = \epsilon) = W(j \rightarrow i) w_j(t = 0).$$

## Digression: Diffusion from Markov Chain

This equation represents the discretized time-development of an original PDF. Since both $W$ and $w$ represent probabilities, they have to be normalized, i.e., we require that at each time step we have

$$\sum_i w_i(t) = 1,$$

and

$$\sum_j W(j \rightarrow i) = 1.$$

The further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$. Note that the probability for remaining at the same place is in general not necessarily equal zero. In our Markov process we allow only for jumps to the left or to the right.

## Digression: Diffusion from Markov Chain

The time development of our initial PDF can now be represented through the action of the transition probability matrix applied $n$ times. At a time $t_n = n\epsilon$ our initial distribution has developed into

$$w_i(t_n) = \sum_j W_{ij}(t_n) w_j(0),$$

and defining

$$W(il - jl, n\epsilon) = (W^n(\epsilon))_{ij}$$

we obtain

$$w_i(n\epsilon) = \sum_j (W^n(\epsilon))_{ij} w_j(0),$$

or in matrix form

$$w(\hat{n}\epsilon) = \hat{W}^n(\epsilon)\hat{w}(0).$$

## Digression: Diffusion from Markov Chain

The matrix $\hat{W}$ can be written in terms of two matrices

$$\hat{W} = \frac{1}{2}\left(\hat{L} + \hat{R}\right),$$

where $\hat{L}$ and $\hat{R}$ represent the transition probabilities for a jump to the left or the right, respectively. For a $4 \times 4$ case we could write these matrices as

$$\hat{R} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

and

$$\hat{L} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

## Digression: Diffusion from Markov Chain

However, in principle these are infinite dimensional matrices since the number of time steps are very large or infinite. For the infinite case we can write these matrices $R_{ij} = \delta_{i,(j+1)}$ and $L_{ij} = \delta_{(i+1),j}$, implying that

$$\hat{L}\hat{R} = \hat{R}\hat{L} = 1,$$

and

$$\hat{L} = \hat{R}^{-1}$$

To see that $\hat{L}\hat{R} = \hat{R}\hat{L} = 1$, perform e.g., the matrix multiplication

$$\hat{L}\hat{R} = \sum_k \hat{L}_{ik}\hat{R}_{kj} = \sum_k \delta_{(i+1),k}\delta_{k,(j+1)} = \delta_{i+1,j+1} = \delta_{i,j},$$

and only the diagonal matrix elements are different from zero.

## Digression: Diffusion from Markov Chain

For the first time step we have thus

$$\hat{W} = \frac{1}{2}\left(\hat{L} + \hat{R}\right),$$

and using the properties in Eqs. (271) and (271) we have after two time steps

$$\hat{W}^2(2\epsilon) = \frac{1}{4}\left(\hat{L}^2 + \hat{R}^2 + 2\hat{R}\hat{L}\right),$$

and similarly after three time steps

$$\hat{W}^3(3\epsilon) = \frac{1}{8}\left(\hat{L}^3 + \hat{R}^3 + 3\hat{R}\hat{L}^2 + 3\hat{R}^2\hat{L}\right).$$

## Digression: Diffusion from Markov Chain

Using the binomial formula

$$\sum_{k=0}^{n} \left( \begin{array}{c} n \\ k \end{array} \right) \hat{a}^k \hat{b}^{n-k} = (a+b)^n,$$

we have that the transition matrix after $n$ time steps can be written as

$$\hat{W}^n(n\epsilon)) = \frac{1}{2^n} \sum_{k=0}^{n} \left( \begin{array}{c} n \\ k \end{array} \right) \hat{R}^k \hat{L}^{n-k},$$

or

$$\hat{W}^n(n\epsilon)) = \frac{1}{2^n} \sum_{k=0}^{n} \left( \begin{array}{c} n \\ k \end{array} \right) \hat{L}^{n-2k} = \frac{1}{2^n} \sum_{k=0}^{n} \left( \begin{array}{c} n \\ k \end{array} \right) \hat{R}^{2k-n},$$

## Diffusion from Markov Chain

and using $R^m_{ij} = \delta_{i,(j+m)}$ and $L^m_{ij} = \delta_{(i+m),j}$ we arrive at

$$W(il-jl, n\epsilon) = \left\{ \begin{array}{cc} \frac{1}{2^n}\left(\begin{array}{c} n \\ \frac{1}{2}(n+i-j) \end{array}\right) & |i-j| \le n \\ 0 & \text{else} \end{array} \right. ,$$

and $n+i-j$ has to be an even number.

## Diffusion from Markov Chain

We note that the transition matrix for a Markov process has three important properties:

- It depends only on the difference in space $i-j$, it is thus homogenous in space.
- It is also isotropic in space since it is unchanged when we go from $(i,j)$ to $(-i,-j)$.
- It is homogenous in time since it depends only the difference between the initial time and final time.

If we place the walker at $x=0$ at $t=0$ we can represent the initial PDF with $w_i(0)=\delta_{i,0}$. Using Eq. (269) we have

$$w_i(n\epsilon)=\sum_j (W^n(\epsilon))_{ij}w_j(0)=\sum_j \frac{1}{2^n}\left(\begin{array}{c} n \\ \frac{1}{2}(n+i-j)\end{array}\right)\delta_{j,0},$$

resulting in

$$w_i(n\epsilon)=\frac{1}{2^n}\left(\begin{array}{c} n \\ \frac{1}{2}(n+i)\end{array}\right) \qquad |i|\leq n$$

## Diffusion from Markov Chain

Using the recursion relation for the binomials

$$\left( \begin{array}{c} n+1 \\ \frac{1}{2}(n+1+i)) \end{array} \right) = \left( \begin{array}{c} n \\ \frac{1}{2}(n+i+1) \end{array} \right) + \left( \begin{array}{c} n \\ \frac{1}{2}(n+i)-1 \end{array} \right)$$

we obtain, defining $x = il$, $t = n\epsilon$ and setting

$$w(x,t) = w(il, n\epsilon) = w_i(n\epsilon),$$

$$w(x, t+\epsilon) = \frac{1}{2}w(x+l, t) + \frac{1}{2}w(x-l, t),$$

and adding and subtracting $w(x,t)$ and multiplying both sides with $l^2/\epsilon$ we have

## Diffusion from Markov Chain

$$\frac{w(x,t+\epsilon)-w(x,t)}{\epsilon}=\frac{l^2}{2\epsilon}\frac{w(x+l,t)-2w(x,t)+w(x-l,t)}{l^2},$$

and identifying $D=l^2/2\epsilon$ and letting $l=\Delta x$ and $\epsilon=\Delta t$ we see that this is nothing but the discretized version of the diffusion equation. Taking the limits $\Delta x\rightarrow 0$ and $\Delta t\rightarrow 0$ we recover

$$\frac{\partial w(x,t)}{\partial t}=D\frac{\partial^2 w(x,t)}{\partial x^2},$$

the diffusion equation.

## Continuous Equation

Hitherto we have considered discretized versions of all equations. Our initial probability distribution function was then given by

$$w_i(0) = \delta_{i,0},$$

and its time-development after a given time step $\Delta t = \epsilon$ is

$$w_i(t) = \sum_j W(j \rightarrow i) w_j(t = 0).$$

The continuous analog to $w_i(0)$ is

$$w(\mathbf{x}) \rightarrow \delta(\mathbf{x}),$$

where we now have generalized the one-dimensional position $x$ to a generic-dimensional vector $\mathbf{x}$. The Kroenecker $\delta$ function is replaced by the $\delta$ distribution function $\delta(\mathbf{x})$ at $t = 0$.

The transition from a state $j$ to a state $i$ is now replaced by a transition to a state with position $\mathbf{y}$ from a state with position $\mathbf{x}$.

## Continuous Equation

The discrete sum of transition probabilities can then be replaced by an integral and we obtain the new distribution at a time $t + \Delta t$ as

$$w(\mathbf{y}, t + \Delta t) = \int W(\mathbf{y}, \mathbf{x}, \Delta t) w(\mathbf{x}, t) d\mathbf{x},$$

and after $m$ time steps we have

$$w(\mathbf{y}, t + m\Delta t) = \int W(\mathbf{y}, \mathbf{x}, m\Delta t) w(\mathbf{x}, t) d\mathbf{x}.$$

When equilibrium is reached we have

$$w(\mathbf{y}) = \int W(\mathbf{y}, \mathbf{x}, t) w(\mathbf{x}) d\mathbf{x}.$$

## Continuous Equation

We can solve the equation for $w(\mathbf{y},t)$ by making a Fourier transform to momentum space. The PDF $w(\mathbf{x},t)$ is related to its Fourier transform $\tilde{w}(\mathbf{k},t)$ through

$$w(\mathbf{x},t) = \int_{-\infty}^{\infty} d\mathbf{k} \exp{(i\mathbf{kx})}\tilde{w}(\mathbf{k},t),$$

and using the definition of the $\delta$-function

$$\delta(\mathbf{x}) = \frac{1}{2\pi}\int_{-\infty}^{\infty} d\mathbf{k} \exp{(i\mathbf{kx})},$$

we see that

$$\tilde{w}(\mathbf{k},0) = 1/2\pi.$$

## Continuous Equation

We can then use the Fourier-transformed diffusion equation

$$\frac{\partial \tilde{w}(\mathbf{k},t)}{\partial t} = -D\mathbf{k}^2\tilde{w}(\mathbf{k},t),$$

with the obvious solution

$$\tilde{w}(\mathbf{k},t) = \tilde{w}(\mathbf{k},0)\exp\left[-(D\mathbf{k}^2t)\right) = \frac{1}{2\pi}\exp\left[-(D\mathbf{k}^2t)\right].$$

We then obtain

$$w(\mathbf{x},t) = \int_{-\infty}^{\infty} d\mathbf{k}\exp\left[i\mathbf{k}\mathbf{x}\right]\frac{1}{2\pi}\exp\left[-(D\mathbf{k}^2t)\right] = \frac{1}{\sqrt{4\pi Dt}}\exp\left[-(\mathbf{x}^2/4Dt)\right],$$

with the normalization condition

$$\int_{-\infty}^{\infty} w(\mathbf{x},t)d\mathbf{x} = 1.$$

## Continuous Equation

It is rather easy to verify by insertion that the above is a solution of the diffusion equation. The solution represents the probability of finding our random walker at position $\mathbf{x}$ at time $t$ if the initial distribution was placed at $\mathbf{x} = 0$ at $t = 0$.
There is another interesting feature worth observing. The discrete transition probability $W$ itself is given by a binomial distribution. The results from the central limit theorem, see yesterday's lecture, state that the transition probability in the limit $n \rightarrow \infty$ converges to the normal distribution. It is then possible to show that

$$W(il - jl, n\epsilon) \rightarrow W(\mathbf{y}, \mathbf{x}, \Delta t) = \frac{1}{\sqrt{4\pi D\Delta t}} \exp\left[-((\mathbf{y} - \mathbf{x})^2/4D\Delta t)\right],$$

and that it satisfies the normalization condition and is itself a solution to the diffusion equation.

## Variational MC and Importance Sampling

The Fokker-Planck equation yields a transition probability given by the Green's function

$$G(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-(y-x-D\Delta t F(x))^2/4D\Delta t\right) \tag{93}$$

which in turn means that our brute force Metropolis algorithm

$$A(y,x) = \min(1, q(y,x))), \tag{94}$$

with $q(y,x) = |\Psi_T(y)|^2/|\Psi_T(x)|^2$ is now replaced by

$$q(y,x) = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2} \tag{95}$$

Below we discuss a simple implementation of this algorithm for a simple particle in a one-dimensional harmonic oscillator potential.

## Project 2, vmc-fp.cpp, 1-dim HO

```
int main() {
    int MCSteps;
    cin >> MCSteps;
    initialize();
    // perform 20% of MCSteps as thermalization steps
    // and adjust time step size so acceptance ratio ~90%
    int thermSteps = int(0.2 * MCSteps);
    int adjustInterval = int(0.1 * thermSteps) + 1;
    nAccept = 0;
    cout << " Performing " << thermSteps << " thermalization steps ...
         << flush;
    for (int i = 0; i < thermSteps; i++) {
        oneMonteCarloStep();
        if ((i+1) % adjustInterval == 0) {
            tStep *= nAccept / (0.9 * N * adjustInterval);
            nAccept = 0;
        }
    }
```

## Project 2 programs/vmc-fp.cpp

```
    cout << "\n Adjusted time step size = " << tStep << endl;

    // production steps
    zeroAccumulators();
    nAccept = 0;
    cout << " Performing " << MCSteps << " production steps ..." << fl
    for (int i = 0; i < MCSteps; i++)
        oneMonteCarloStep();
    // compute and print energy
    double eAve = eSum / double(N) / MCSteps;
    double eVar = eSqdSum / double(N) / MCSteps - eAve * eAve;
    double error = sqrt(eVar) / sqrt(double(N) * MCSteps);
    cout << "\n <Energy> = " << eAve << " +/- " << error
         << "\n Variance = " << eVar << endl;
}
void oneMonteCarloStep() {
    // perform N Metropolis steps
    for (int n = 0; n < N; n++) {
        MetropolisStep(n);
    }
}
```

## VMC and Importance Sampling

```
void MetropolisStep(int n) {
    // make a trial move
    double x = ::x[n];              // :: chooses the global x
    double Fx = - 4 * alpha * x;
    double y = x + gasdev(seed) * sqrt(tStep) + Fx * tStep / 2;
    // compute ratio for Metropolis test
    double rhoRatio = exp( - 2 * alpha * (y * y - x * x));
    double oldExp = y - x - Fx * tStep / 2;
    double Fy = - 4 * alpha * y;
    double newExp = x - y - Fy * tStep / 2;
    double GRatio = exp( -(newExp * newExp - oldExp * oldExp) / (2 * t
    double w = rhoRatio * GRatio;
    // Metropolis test
    if (w > ran2(seed)) {
        ::x[n] = x = y;
        ++nAccept;
    }
    // accumulate energy and wave function
    double e = eLocal(x);
    eSum += e;
    eSqdSum += e * e;
}
```

## VMC and Importance Sampling

```
void initialize() {
    x = new double [N];
    for (int i = 0; i < N; i++)
        x[i] = qadran() - 0.5;
    tStep = 0.1;
}
void zeroAccumulators() {
    eSum = eSqdSum = 0;
}
double eLocal(double x) {

    // compute the local energy
    return alpha + x * x * (0.5 - 2 * alpha * alpha);
}
```

## Diffusion Monte Carlo

Choose $\tau = it/\hbar$. The time-dependent Schrödinger equation becomes then

$$\frac{\partial \Psi(x,\tau)}{\partial \tau} = \frac{\hbar^2}{2m}\frac{\partial^2 \Psi(x,\tau)}{\partial x^2}$$

Diffusion constant

$$D = \frac{\hbar^2}{2m}$$

Can solve this equation with a random walk algorithm for the above diffusion equation. What happens with an interaction term?

$$\frac{\partial \Psi(x,\tau)}{\partial \tau} = \frac{\hbar^2}{2m}\frac{\partial^2 \Psi(x,\tau)}{\partial x^2} - V(x)\Psi(x,\tau)$$

## Diffusion Monte Carlo

Without the kinetic energy term we have

$$\frac{\partial\Psi(x,\tau)}{\partial\tau} = -V(x)\Psi(x,\tau)$$

which is the same as a decay or growth process (depending on the sign of $V$). We can obtain the solution to this first-order differential equation by replacing it by a random decay or growth process. We can thus interpret the full SE as a combination of diffusion and branching processes. For the latter, the number of random walkers at a point $x$ depends on the sign of $V(x)$.

## Diffusion Monte Carlo

A crucial aspect (which leads to the Monte Carlo sign problem for Fermions) is that the probability distribution is no longer

$$P(x,\tau) = \Psi^*(x,\tau)\Psi(x,\tau)dx$$

but

$$P(x,\tau) = \Psi(x,\tau)dx$$

$\Psi$ must be nonnegative! It is related to distribution of walkers.
The general solution to SE

$$\Psi(x,\tau) = \sum_n c_n\phi_n(x)e^{-E_n\tau}$$

For sufficiently large $\tau$ the dominant term becomes the eigenvalue with lowest energy

$$\Psi(x,\tau\rightarrow\infty) = c_0\phi_0(x)e^{-E_0\tau}$$

## Diffusion Monte Carlo

Note

- Spatial dependence of $\Psi(x, \tau \to \infty)$ proportional to $\phi_0$
- The population of walkers will however decay to zero unless $E_0 = 0$!
- Can avoid this problem by introducing an arbitrary reference energy $V_{\mathrm{ref}}$, which is adjusted so that an approximate steady state distribution of random walkers is obtained.

We obtain then

$$\frac{\partial \Psi(x,\tau)}{\partial \tau} = \frac{\hbar^2}{2m}\frac{\partial^2 \Psi(x,\tau)}{\partial x^2} - \left[V(x) - V_{\mathrm{ref}}\right]\Psi(x,\tau),$$

and

$$\Psi(x,\tau) \approx c_0\phi_0(x)e^{-(E_0 - V_{\mathrm{ref}})\tau}$$

## Diffusion Monte Carlo

The DMC method is based on rewriting the SE in imaginary time, by defining $\tau = it$. The imaginary time SE is then

$$\frac{\partial \psi}{\partial \tau} = -\widehat{\mathbf{H}}\psi.$$

The wave function $\psi$ is again expanded in eigenstates of the Hamiltonian

$$\psi = \sum_{i}^{\infty} c_i \phi_i,$$

where

$$\widehat{\mathbf{H}}\phi_i = \epsilon_i \phi_i,$$

$\epsilon_i$ being an eigenstate of $\widehat{\mathbf{H}}$. A formal solution of the imaginary time Schrödinger equation is

$$\psi(\tau_1 + \delta\tau) = e^{-\widehat{\mathbf{H}}\delta\tau}\psi(\tau_1).$$

# Diffusion Monte Carlo

The DMC equation reads

$$-\frac{\partial\psi(\mathbf{R},\tau)}{\partial\tau} = \left[\sum_i^N -\frac{1}{2}\nabla_i^2\psi(\mathbf{R},\tau)\right] + (V(\mathbf{R}) - E_T)\psi(\mathbf{R}).$$

This equation is a diffusion equation where the wave function $\psi$ may be interpreted as the density of diffusing particles (or “walkers”), and the term $V(\mathbf{R}) - E_T$ is a rate term describing a potential-dependent increase or decrease in the particle density. The above equation may be transformed into a form suitable for Monte Carlo methods, but this leads to a very inefficient algorithm. The potential $V(\mathbf{R})$ is unbounded in e.g., atomic systems and hence the rate term $V(\mathbf{R}) - E_T$ can diverge. Large fluctuations in the particle density then result and give impractically large statistical errors.

## Diffusion Monte Carlo

These fluctuations may be substantially reduced by the incorporation of importance sampling in the algorithm. Importance sampling is essential for DMC methods, if the simulation is to be efficient. A trial or guiding wave function $\psi_T(\mathbf{R})$, which closely approximates the ground state wave function is introduced.
For the trial wave function and energy, one typically uses the results from a as best as possible VMC calculation. A new distribution is defined as

$$f(\mathbf{R},\tau) = \psi_T(\mathbf{R})\psi(\mathbf{R},\tau),$$

which is also a solution of the SE when $\psi(\mathbf{R},\tau)$ is a solution. Modified to

$$\frac{\partial f(\mathbf{R},\tau)}{\partial \tau} = \frac{1}{2}\nabla\left[\nabla - F(\mathbf{R})\right] f(\mathbf{R},\tau) + (E_L(\mathbf{R}) - E_T)f(\mathbf{R},\tau).$$

## Diffusion Monte Carlo

In this equation we have introduced the so-called force-term $F$, given by

$$F(\mathbf{R}) = \frac{2\nabla\psi_T(\mathbf{R})}{\psi_T(\mathbf{R})},$$

and is commonly referred to as the “quantum force”. The local energy $E_L$ is defined as previously

$$E_L\mathbf{R}) = -\frac{1}{\psi_T(\mathbf{R})}\frac{\nabla^2\psi_T(\mathbf{R})}{2} + V(\mathbf{R}),$$

and is computed, as in the VMC method, with respect to the trial wave function.

## Diffusion Monte Carlo

We can give the following interpretation. The right hand side of the importance sampled DMC equation consists, from left to right, of diffusion, drift and rate terms. The problematic potential dependent rate term of the non-importance sampled method is replaced by a term dependent on the difference between the local energy of the guiding wave function and the trial energy. The trial energy is initially chosen to be the VMC energy of the trial wave function, and is updated as the simulation progresses. Use of an optimised trial function minimises the difference between the local and trial energies, and hence minimises fluctuations in the distribution $f$ .

## DMC

Our previous Green's function, (the diffusion part only)

$$G_{\mathrm{Diff}}(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(-(y-x-D\Delta tF(x))^2/4D\Delta t\right) \tag{96}$$

is replaced by a diffusion piece and a branching part

$$G_{\mathrm{B}}(y,x,\Delta t) = \exp\left(-\left[\frac{1}{2}(E_L(y)+E_L(x))-E_T\right]\Delta t\right) \tag{97}$$

yielding

$$G_{\mathrm{DMC}}(y,x,\Delta t) \approx G_{\mathrm{Diff}}(y,x,\Delta t)G_{\mathrm{B}}(y,x,\Delta t) \tag{98}$$

with $E_L$ being the local energy and $E_T$ our trial energy. The Metropolis algorithm is still

$$A(y,x) = \min(1, q(y,x))), \tag{99}$$

with

$$q(y,x) = \frac{G_{\mathrm{DMC}}(x,y,\Delta t)|\Psi_T(y)|^2}{G_{\mathrm{DMC}}(y,x,\Delta t)|\Psi_T(x)|^2} \tag{100}$$

Below we discuss a simple implementation of this algorithm for a simple three-dimensional harmonic oscillator potential.

## DMC programs, 3-dim HO

```
int main() {

    cout << " Diffusion Monte Carlo for the 3-D Harmonic Oscillator\n"
         << " -----------------------------------------------------\n"
    cout << " Enter desired target number of walkers: ";
    cin >> N_T;
    cout << " Enter time step dt: ";
    cin >> dt;
    cout << " Enter total number of time steps: ";
    int timeSteps;
    cin >> timeSteps;

    initialize();

    // do 20% of timeSteps as thermalization steps
    int thermSteps = int(0.2 * timeSteps);
    for (int i = 0; i < thermSteps; i++)
        oneTimeStep();

    // production steps
    zeroAccumulators();
    for (int i = 0; i < timeSteps; i++) {
        oneTimeStep();
    }
```

## DMC programs, 3-dim HO

```
    // compute averages
    double EAve = ESum / timeSteps;
    double EVar = ESqdSum / timeSteps - EAve * EAve;
    cout << " <E> = " << EAve << " +/- " << sqrt(EVar / timeSteps) <<
    cout << " <Eˆ2> - <E>ˆ2 = " << EVar << endl;
    double psiNorm = 0, psiExactNorm = 0;
    double dr = rMax / NPSI;
    for (int i = 0; i < NPSI; i++) {
        double r = i * dr;
        psiNorm += r * r * psi[i] * psi[i];
        psiExactNorm += r * r * exp(- r * r);
    }
    psiNorm = sqrt(psiNorm);
    psiExactNorm = sqrt(psiExactNorm);
    ofstream file("psi.data");
    for (int i = 0; i < NPSI; i++) {
        double r = i * dr;
        file << r << '\t' << r * r * psi[i] / psiNorm << '\t'
             << r * r * exp(- r * r / 2) / psiExactNorm << '\n';
    }
    file.close();
}
```

# DMC programs, 3-dim HO

```
// Diffusion Monte Carlo program for the 3-D harmonic oscillator

#include <cmath>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include "rng.h"

using namespace std;

int seed = -987654321;          // for ran2 and gasdev
const int DIM = 3;              // dimensionality of space

double V(double *r) {           // harmonic oscillator in DIM dimension
    double rSqd = 0;
    for (int d = 0; d < DIM; d++)
        rSqd += r[d] * r[d];
    return 0.5 * rSqd;
}
```

## DMC programs, 3-dim HO

```
double dt;                      // Delta_t set by user
double E_T;                     // target energy

// random walkers
int N;                          // current number of walkers
int N_T;                        // desired target number of walkers
double **r;                     // x,y,z positions of walkers
bool *alive;                    // is this walker alive?

// observables
double ESum;                    // accumulator for energy
double ESqdSum;                 // accumulator for variance
double rMax = 4;                // max value of r to measure psi
const int NPSI = 100;           // number of bins for wave function
double psi[NPSI];               // wave function histogram
```

## DMC programs, 3-dim HO

```
void ensureCapacity(int index) {

    static int maxN = 0;        // remember the size of the array

    if (index < maxN)           // no need to expand array
        return;                 // do nothing

    int oldMaxN = maxN;         // remember the old capacity
    if (maxN > 0)
        maxN *= 2;              // double capacity
    else
        maxN = 1;
    if (index > maxN - 1)       // if this is not sufficient
        maxN = index + 1;       // increase it so it is sufficient
```

# DMC programs, 3-dim HO

```
    // allocate new storage
    double **rNew = new double* [maxN];
    bool *newAlive = new bool [maxN];
    for (int n = 0; n < maxN; n++) {
        rNew[n] = new double [DIM];
        if (n < oldMaxN) {     // copy old values into new arrays
            for (int d = 0; d < DIM; d++)
                rNew[n][d] = r[n][d];
            newAlive[n] = alive[n];
            delete [] r[n];    // release old memory
        }
    }
    delete [] r;               // release old memory
    r = rNew;                  // point r to the new memory
    delete [] alive;
    alive = newAlive;
}
```

## DMC programs, 3-dim HO

```
void zeroAccumulators() {
    ESum = ESqdSum = 0;
    for (int i = 0; i < NPSI; i++)
        psi[i] = 0;
}

void initialize() {
    N = N_T;                          // set N to target number specified by
    for (int n = 0; n < N; n++) {
        ensureCapacity(n);
        for (int d = 0; d < DIM; d++)
            r[n][d] = ran2(seed) - 0.5;
        alive[n] = true;
    }
    zeroAccumulators();
    E_T = 0;                          // initial guess for the ground state e
}
```

## DMC programs, 3-dim HO

```
void oneMonteCarloStep(int n) {
    // Diffusive step
    for (int d = 0; d < DIM; d++)
        r[n][d] += gasdev(seed) * sqrt(dt);
    // Branching step
    double q = exp(- dt * (V(r[n]) - E_T));
    int survivors = int(q);
    if (q - survivors > ran2(seed))
        ++survivors;
    // append survivors-1 copies of the walker to the end of the array
    for (int i = 0; i < survivors - 1; i++) {
        ensureCapacity(N);
        for (int d = 0; d < DIM; d++)
            r[N][d] = r[n][d];
        alive[N] = true;
        ++N;
    }
    // if survivors is zero, then kill the walker
    if (survivors == 0)
        alive[n] = false;
}
```

# DMC programs, 3-dim HO

```
void oneTimeStep() {
    // DMC step for each walker
    int N_0 = N;
    for (int n = 0; n < N_0; n++)
        oneMonteCarloStep(n);
    // remove all dead walkers from the arrays
    int newN = 0;
    for (int n = 0; n < N; n++)
    if (alive[n]) {
        if (n != newN) {
            for (int d = 0; d < DIM; d++)
                r[newN][d] = r[n][d];
            alive[newN] = true;
        }
        ++newN;
    }
```

## DMC programs, 3-dim HO

```
    N = newN;
    // adjust E_T
    E_T += log(N_T / double(N)) / 10;
    // measure energy, wave function
    ESum += E_T;
    ESqdSum += E_T * E_T;
    for (int n = 0; n < N; n++) {
        double rSqd = 0;
        for (int d = 0; d < DIM; d++)
            rSqd = r[n][d] * r[n][d];
        int i = int(sqrt(rSqd) / rMax * NPSI);
        if (i < NPSI)
            psi[i] += 1;
    }
}
```

## How to obtain the energy?

Note the statement

```
    // adjust E_T
    E_T += log(N_T / double(N)) / 10;
```

What does it mean? After many time steps, the remaining time dependence (complex) for $\Psi$ is proportional to $e^{-(E_0-E_T)\Delta t}$. To obtain $E_0$ we need to follow the growth of the population of random walkers. If we let the current poulation of walkers be $N(t)$, it should be proportional to the function $f(x,t)$, viz

$$N(t) = \int f(x,t)dx. \tag{101}$$

The change in population is

$$N(t+\Delta t) = e^{-(E_0-E_T)\Delta t}N(t). \tag{102}$$

## How to obtain the energy?

For large enough numbers of Monte Carlo cycles, we can estimate the ground state energy from the so-called growth energy $E_g$ which can be obtained from the populations of walkers at different times, obtaining

$$E_g = E_T + \frac{1}{t_2 - t_1} ln(\frac{N(t_1)}{N(t_w)}), \tag{103}$$

yielding an energy $E_0$ which is the average of $< E_g >$. This is not the best choice since the population of walkers can vary strongly from time to time, resulting in large deviations of the mean. Test this for the enclosed program. A better estimater for the energy is the local energy $E_L$.