ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DPHYS
Department of Physics
Institute for Theoretical Physics

# Quantum Monte Carlo World Line Algorithms

Matthias Troyer, ETH Zürich

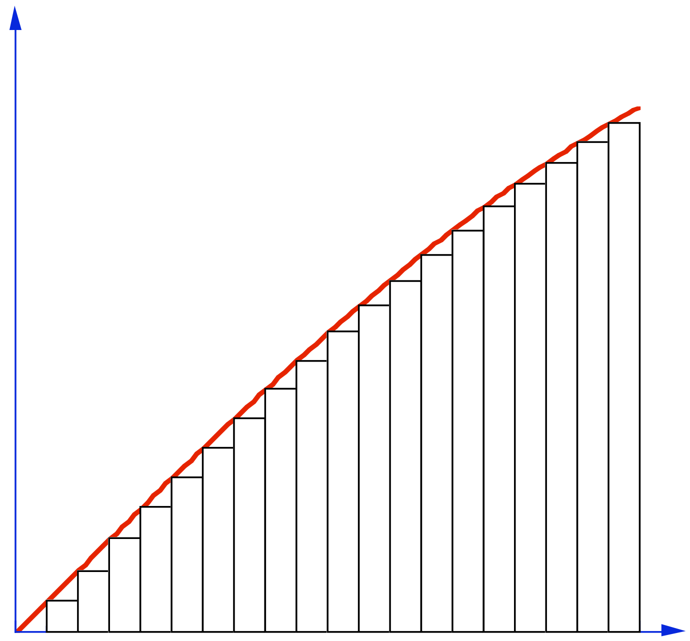
FNSNF
Swiss National Science Foundation
DARPA
erc

# 1. Monte Carlo Integration

# Integrating a function

- Convert the integral to a discrete sum

$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{i=1}^{N} f\left(a + i\frac{b-a}{N}\right) + O(1/N)$$



- Higher order integrators:
  - Trapezoidal rule:

$$\int_a^b f(x)dx = \frac{b-a}{N}\left(\frac{1}{2}f(a) + \sum_{i=1}^{N-1} f\left(a + i\frac{b-a}{N}\right) + \frac{1}{2}f(b)\right) + O(1/N^2)$$

  - Simpson rule:

$$\int_a^b f(x)dx = \frac{b-a}{3N}\left(f(a) + \sum_{i=1}^{N-1}(3-(-1)^i) f\left(a + i\frac{b-a}{N}\right) + f(b)\right) + O(1/N^4)$$



Saturday, January 21, 12

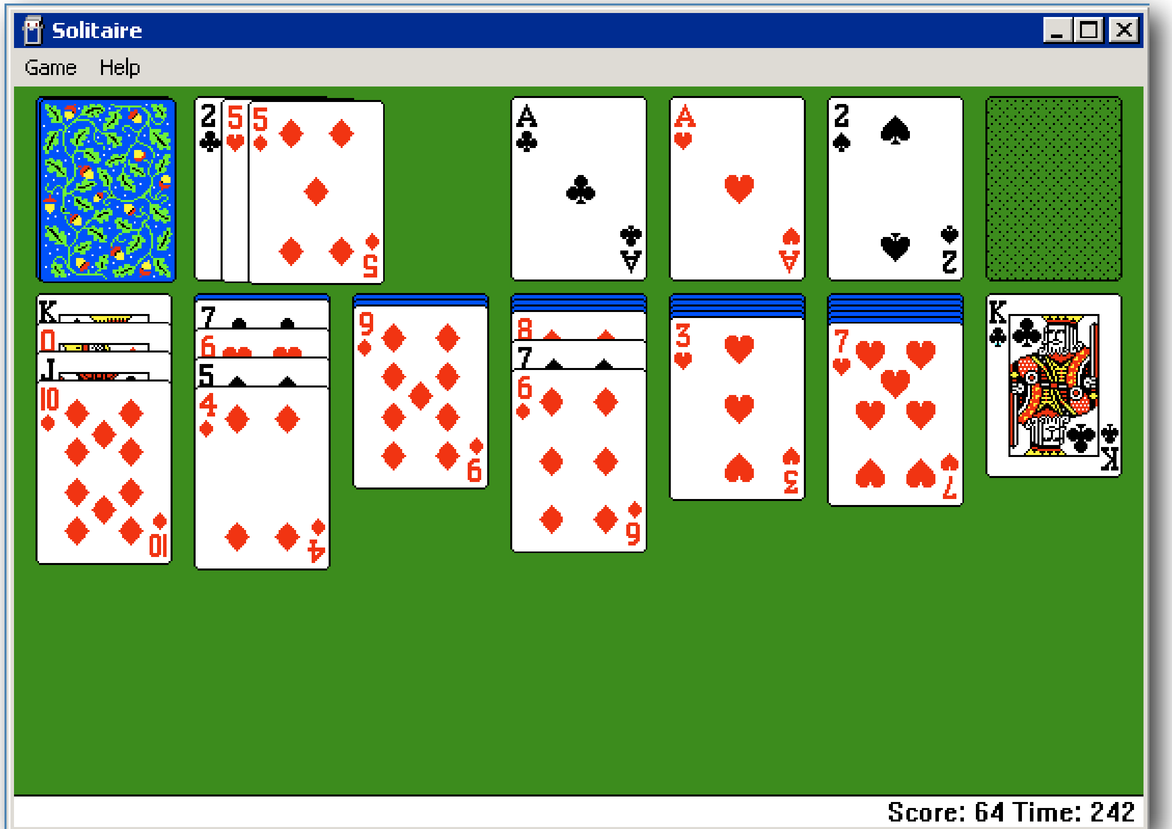
# High dimensional integrals

- Simpson rule with $M$ points per dimension
  - one dimension the error is O( $M^{-4}$ )
  - $d$ dimensions we need $N = M^d$ points

    the error is order O( $M^{-4}$ ) = O( $N^{-4/d}$ )
- An order - $n$ scheme in 1 dimension is order - $n/d$ $d$ in $d$ dimensions!
- In a statistical mechanics model with $N$ particles we have 6$N$-dimensional integrals (3$N$ positions and 3$N$ momenta).
- Integration becomes extremely inefficient!

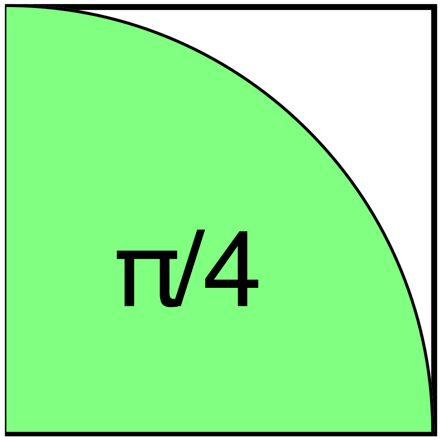
# Ulam: the Monte Carlo Method

- What is the probability to win in Solitaire?
  - Ulam's answer: play it 100 times, count the number of wins and you have a pretty good estimate




# Throwing stones into a pond

- How can we calculate π by throwing stones?
- Take a square surrounding the area we want to measure:




- Choose *M* pairs of random numbers ( *x, y* ) and count how many points ( *x, y* ) lie in the interesting area

# Monte Carlo integration

- Consider an integral

$$\langle f \rangle = \int_\Omega f(\vec{x})d\vec{x} \Big/ \int_\Omega d\vec{x}$$

- Instead of evaluating it at equally spaced points evaluate it at *M* points $x_i$ chosen randomly in Ω:

$$\langle f \rangle \approx \frac{1}{M}\sum_{i=1}^{M} f(\vec{x}_i)$$

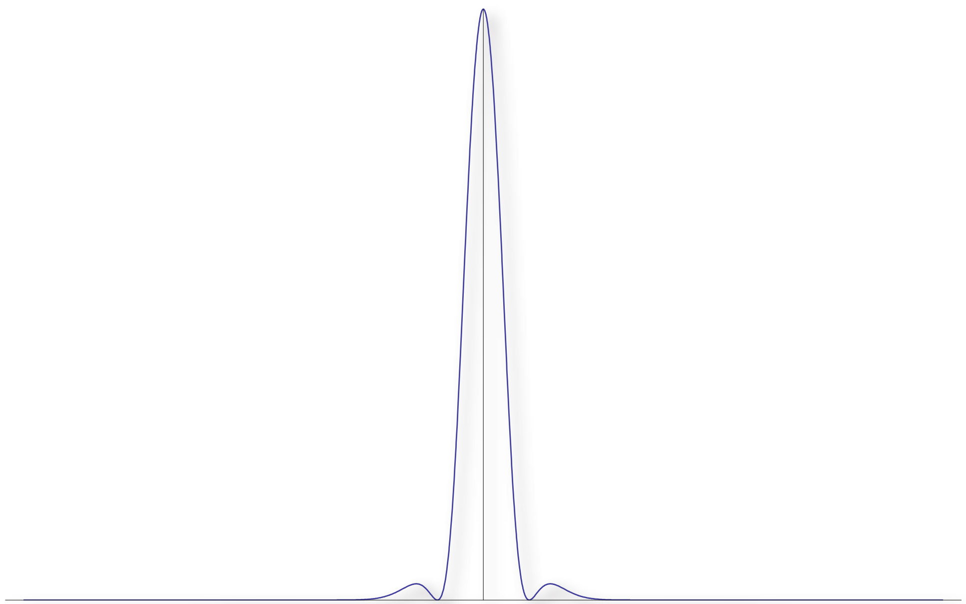
- The error is statistical:

$$\Delta = \sqrt{\frac{\mathrm{Var}\, f}{M}} \propto M^{-1/2}$$

$$\mathrm{Var}\, f = \langle f^2 \rangle - \langle f \rangle^2$$

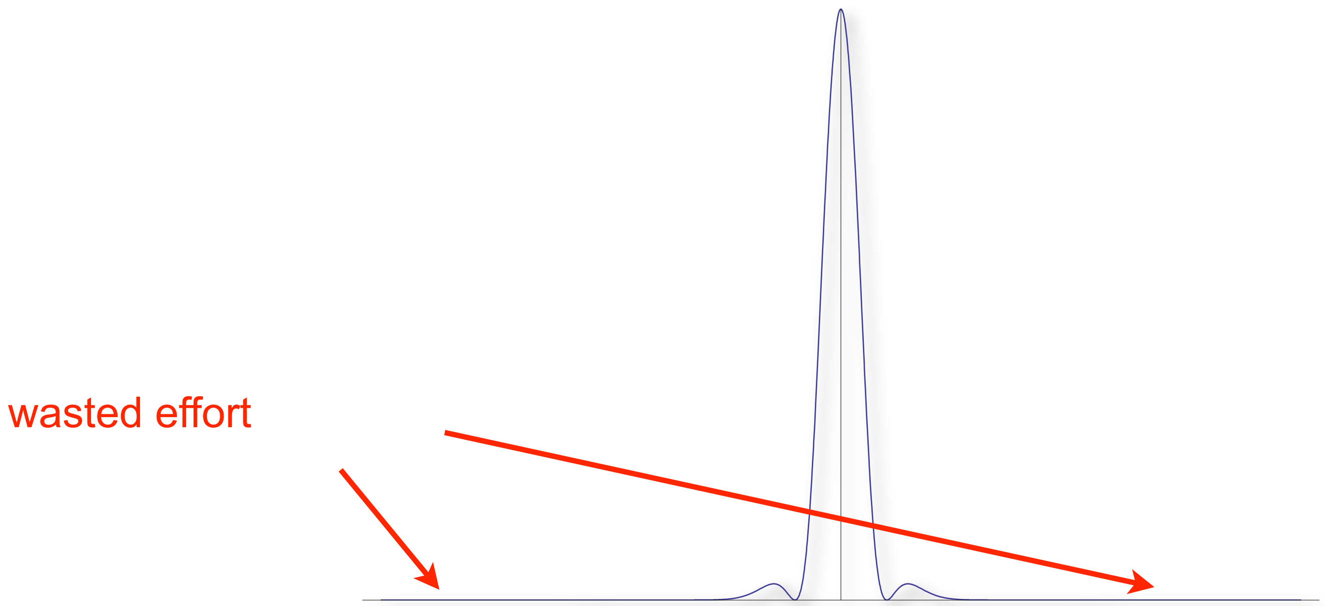
- In *d*>8 dimensions Monte Carlo is better than Simpson!

# Sharply peaked functions



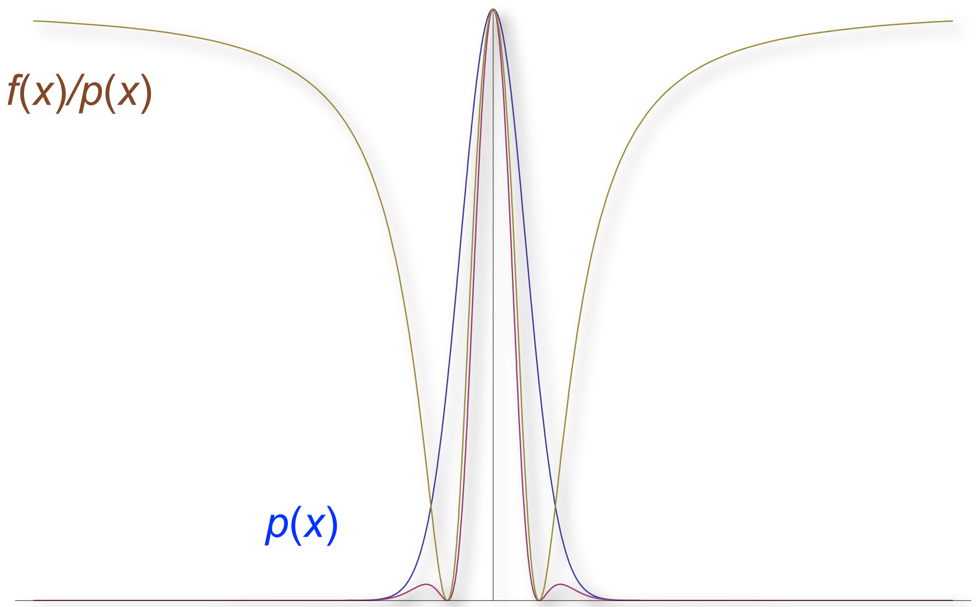
- In many cases a function is large only in a tiny region
- Lots of time wasted in regions where the function is small
- The sampling error is large since the variance is large

# Sharply peaked functions




- In many cases a function is large only in a tiny region
- Lots of time wasted in regions where the function is small
- The sampling error is large since the variance is large

# Importance sampling




- Choose points not uniformly but with probability $p(x)$:

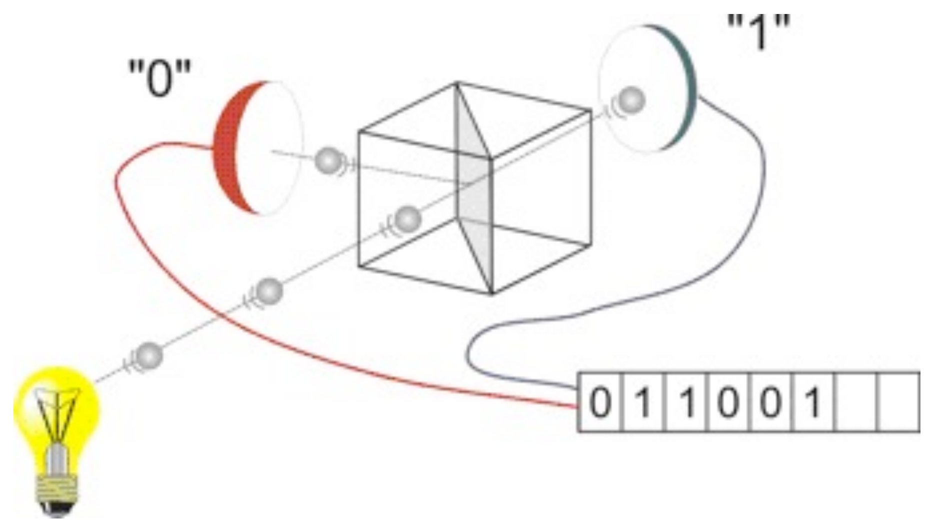
$$\langle f \rangle = \left\langle \frac{f}{p} \right\rangle_p := \int_\Omega \frac{f(\vec{x})}{p(\vec{x})} p(\vec{x}) d\vec{x} \Big/ \int_\Omega d\vec{x}$$

- The error is now determined by Var $f/p$
- Find $p$ similar to $f$ and such that $p$-distributed random numbers are easily available

# 2. Generating Random Numbers

# Random numbers

- Real random numbers are hard to obtain
  - classical chaos (atmospheric noise)
  - quantum mechanics
- Commercial products: quantum random number generators
  - based on photons and semi-transparent mirror
  - 4 Mbit/s from a USB device, too slow for most MC simulations


http://www.idquantique.com/

# Pseudo Random numbers

# Pseudo Random numbers

- Are generated by an algorithm

# Pseudo Random numbers

- Are generated by an algorithm
- Not random at all, but completely deterministic

# Pseudo Random numbers

- Are generated by an algorithm
- Not random at all, but completely deterministic
- Look nearly random however when algorithm is not known and may be good enough for our purposes

# Pseudo Random numbers

- Are generated by an algorithm
- Not random at all, but completely deterministic
- Look nearly random however when algorithm is not known and may be good enough for our purposes
- Never trust pseudo random numbers however!

# Linear congruential generators

- are of the simple form $x_{n+1}=f(x_n)$
- A reasonably good choice is the GGL generator

$$x_{n+1} = (ax_n + c) \bmod m$$

  with $a = 16807$, $c = 0$, $m = 2^{31}-1$
- quality depends sensitively on $a,c,m$

- Periodicity is a problem with such 32-bit generators
  - The sequence repeats identically after $2^{31}-1$ iterations
  - With 500 million numbers per second that is just 4 seconds!
  - Should not be used anymore!

# Lagged Fibonacci generators

$$x_n = x_{n-p} \otimes x_{n-q} \bmod m$$

- Good choices are
  - (2281,1252,+)
  - (9689,5502,+)
  - (44497,23463,+)

- Seed blocks usually generated by linear congruential
- Has very long periods since large block of seeds
- A very fast generator: vectorizes and pipelines very well

# More advanced generators

- As well-established generators fail new tests, better and better generators get developed
  - Mersenne twister (Matsumoto & Nishimura, 1997)
  - Well generator (Panneton and L'Ecuyer , 2004)

- Number theory enters the generator design: predicting the next number is equivalent to solving a very hard mathematical problem

# Are these numbers really random?

# Are these numbers really random?

- No!

# Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?

# Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?
- Statistical tests for distribution and correlations

# Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?
- Statistical tests for distribution and correlations
- Are these tests enough?
  - No! Your calculation could depend in a subtle way on hidden correlations!

# Are these numbers really random?

- No!
- Are they random enough?
  - Maybe?
- Statistical tests for distribution and correlations
- Are these tests enough?
  - No! Your calculation could depend in a subtle way on hidden correlations!
- What is the ultimate test?
  - Run your simulation with various random number generators and compare the results

# Marsaglia's diehard tests

- **Birthday spacings:** Choose random points on a large interval. The spacings between the points should be asymptotically Poisson distributed. The name is based on the birthday paradox.
- **Overlapping permutations:** Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability.
- **Ranks of matrices:** Select some number of bits from some number of random numbers to form a matrix over {0,1}, then determine the rank of the matrix. Count the ranks.
- **Monkey tests:** Treat sequences of some number of bits as "words". Count the overlapping words in a stream. The number of "words" that don't appear should follow a known distribution. The name is based on the infinite monkey theorem.
- **Count the 1s:** Count the 1 bits in each of either successive or chosen bytes. Convert the counts to "letters", and count the occurrences of five-letter "words".
- **Parking lot test:** Randomly place unit circles in a 100 x 100 square. If the circle overlaps an existing one, try again. After 12,000 tries, the number of successfully "parked" circles should follow a certain normal distribution.

# Marsaglia's diehard tests (cont.)

- **Minimum distance test:** Randomly place 8,000 points in a 10,000 x 10,000 square, then find the minimum distance between the pairs. The square of this distance should be exponentially distributed with a certain mean.
- **Random spheres test:** Randomly choose 4,000 points in a cube of edge 1,000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.
- **The squeeze test:** Multiply 231 by random floats on [0,1) until you reach 1. Repeat this 100,000 times. The number of floats needed to reach 1 should follow a certain distribution.
- **Overlapping sums test:** Generate a long sequence of random floats on [0,1). Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and sigma.
- **Runs test:** Generate a long sequence of random floats on [0,1). Count ascending and descending runs. The counts should follow a certain distribution.
- **The craps test:** Play 200,000 games of craps, counting the wins and the number of throws per game. Each count should follow a certain distribution.

# Non-uniform random numbers

- we found ways to generate pseudo random numbers u in the interval [0,1[
- How do we get other uniform distributions?
  - uniform x in [a,b[:  x = a+(b-a) u
- Other distributions:
  - Inversion of integrated distribution
  - Rejection method

# Non-uniform distributions

- How can we get a random number *x* distributed with *f*(*x*) in the interval [*a*,*b*[ from a uniform random number *u*?
- Look at probabilities:

$$P[x < y] = \int_a^y f(t)\,dt =: F(y) \equiv P[u < F(y)]$$

$$\Rightarrow x = F^{-1}(u)$$

- This method is feasible if the integral can be inverted easily
  - exponential distribution $f(x)=\lambda \exp(-\lambda x)$
  - can be obtained from uniform by $x=-1/\lambda \ln(1-u)$

# Normally distributed numbers

- The normal distribution

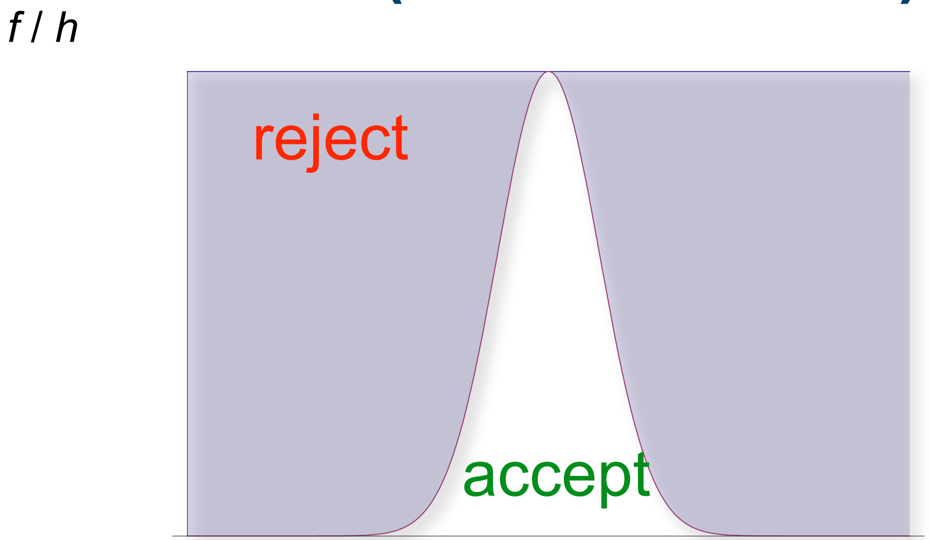
$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2\right)$$

- cannot easily be integrated in one dimension but can be easily integrated in 2 dimensions!
- We can obtain two normally distributed numbers from two uniform ones (Box-Muller method)

$$n_1 = \sqrt{-2\ln(1-u_1)}\,\sin u_2$$

$$n_2 = \sqrt{-2\ln(1-u_1)}\,\cos u_2$$

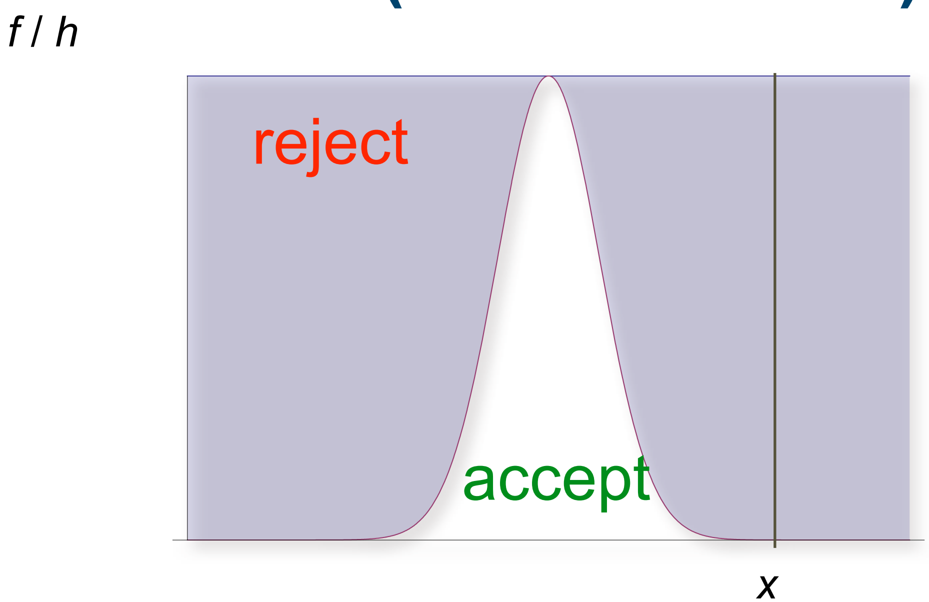
# Rejection method (von Neumann)




- Look for a simple distribution *h* that bounds *f*: $f(x) < \lambda h(x)$
  - Choose an *h*-distributed number *x*
  - Choose a uniform random number number $0 \leq u < 1$
  - Accept *x* if $u < f(x)/\lambda h(x)$,
    otherwise reject *x* and get a new pair $(x,u)$

- Needs a good guess *h* to be efficient, numerical inversion of integral might be faster if no suitable *h* can be found

# Rejection method (von Neumann)



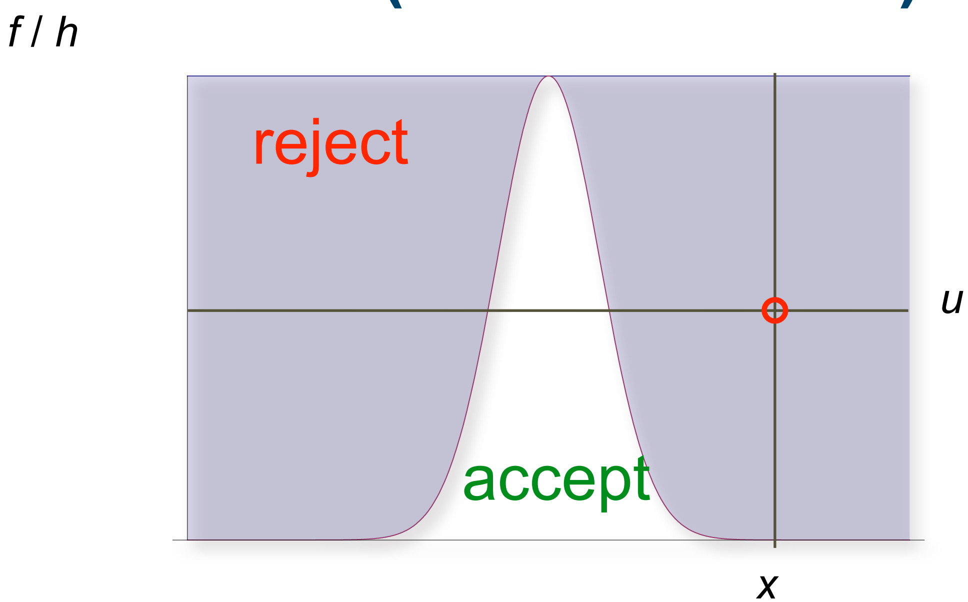

- Look for a simple distribution $h$ that bounds $f$: $f(x) < \lambda h(x)$
  - Choose an $h$-distributed number $x$
  - Choose a uniform random number number $0 \le u < 1$
  - Accept $x$ if $u < f(x)/ \lambda h(x)$,
    otherwise reject $x$ and get a new pair $(x,u)$

- Needs a good guess $h$ to be efficient, numerical inversion of integral might be faster if no suitable $h$ can be found



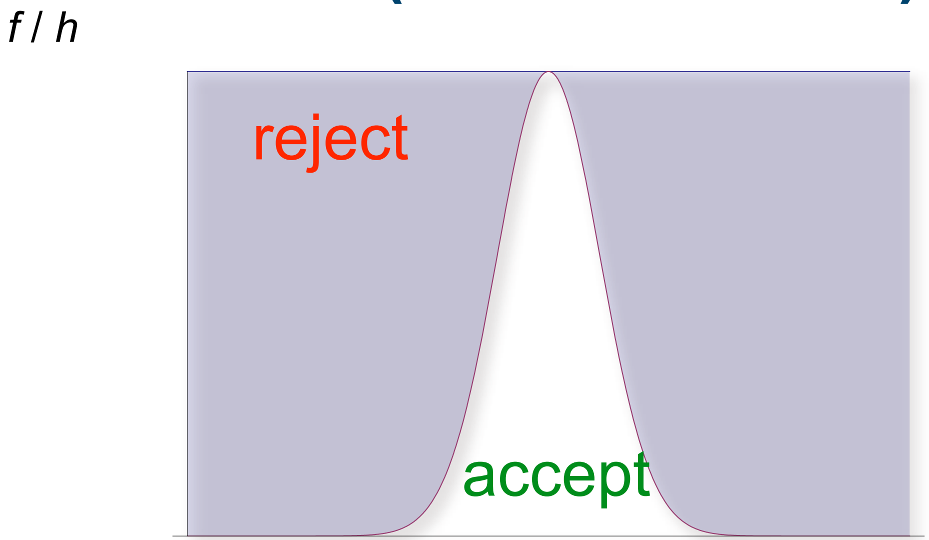
Saturday, January 21, 12

# Rejection method (von Neumann)



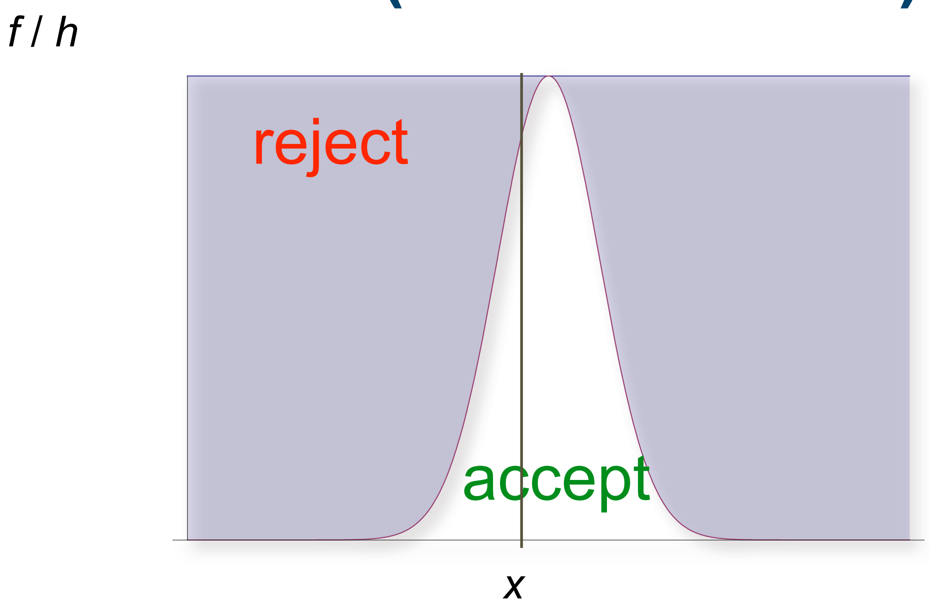

- Look for a simple distribution *h* that bounds *f*: $f(x) < \lambda h(x)$
  - Choose an *h*-distributed number *x*
  - Choose a uniform random number number $0 \leq u < 1$
  - Accept *x* if $u < f(x)/ \lambda h(x)$,
    otherwise reject *x* and get a new pair $(x,u)$
- Needs a good guess *h* to be efficient, numerical inversion of integral might be faster if no suitable *h* can be found

# Rejection method (von Neumann)



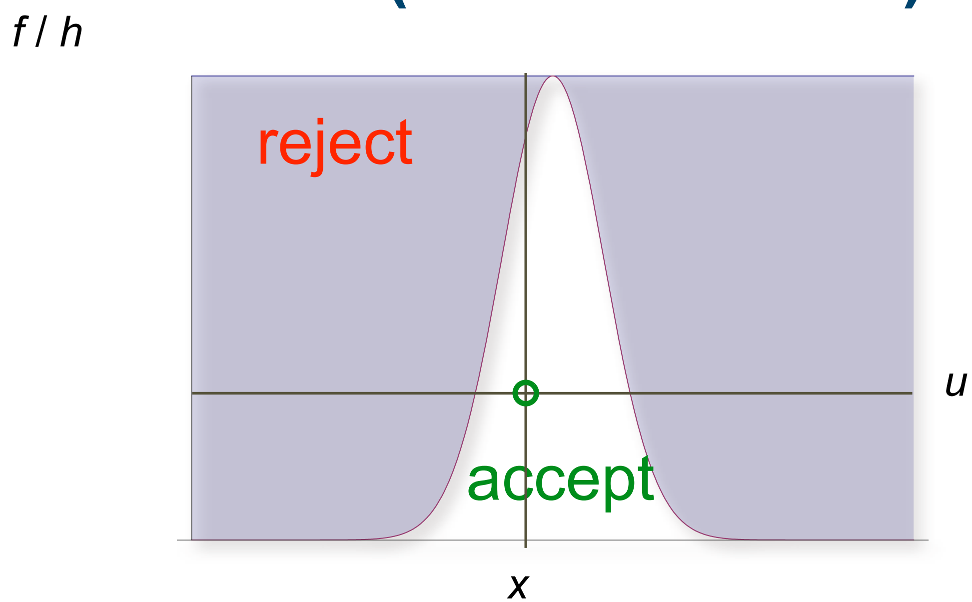

- Look for a simple distribution $h$ that bounds $f$: $f(x) < \lambda h(x)$
  - Choose an $h$-distributed number $x$
  - Choose a uniform random number number $0 \leq u < 1$
  - Accept $x$ if $u < f(x)/\lambda h(x)$,
    otherwise reject $x$ and get a new pair $(x,u)$
- Needs a good guess $h$ to be efficient, numerical inversion of integral might be faster if no suitable $h$ can be found

# Rejection method (von Neumann)




- Look for a simple distribution $h$ that bounds $f$: $f(x) < \lambda h(x)$
  - Choose an $h$-distributed number $x$
  - Choose a uniform random number number $0 \le u < 1$
  - Accept $x$ if $u < f(x)/ \lambda h(x)$,
    otherwise reject $x$ and get a new pair $(x,u)$
- Needs a good guess $h$ to be efficient, numerical inversion of integral might be faster if no suitable $h$ can be found

# Rejection method (von Neumann)




- Look for a simple distribution *h* that bounds *f*: $f(x) < \lambda h(x)$
  - Choose an *h*-distributed number *x*
  - Choose a uniform random number number $0 \le u < 1$
  - Accept *x* if $u < f(x)/\lambda h(x)$,
    otherwise reject *x* and get a new pair $(x,u)$

- Needs a good guess *h* to be efficient, numerical inversion of integral might be faster if no suitable *h* can be found

# 3. The Metropolis Algorithm

# Monte Carlo for classical systems

- Evaluate phase space integral by importance sampling

$$\langle A \rangle = \frac{\int_\Omega A(c)p(c)dc}{\int_\Omega p(c)dc} \longrightarrow \langle A \rangle \approx \overline{A} = \frac{1}{M}\sum_{i=1}^{M} A_{c_i}$$

- Pick configurations with the correct Boltzmann weight

$$P[c] = \frac{p(c)}{Z} = \frac{\exp(-\beta E(c))}{Z}$$

- But how do we create configurations with that distribution? The key problem in statistical mechanics!


- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

computing in SCIENCE & ENGINEERING


- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

computing in SCIENCE & ENGINEERING

# The Metropolis Algorithm (1953)

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

## Equation of State Calculations by Fast Computing Machines

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

AND

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

(Received March 6, 1953)

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

### I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed,

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

In order to reduce the problem to a feasible size for numerical work, we can, of course, consider only a finite number of particles. This number $N$ may be as high as several hundred. Our system consists of a square† con-

# The Metropolis Algorithm (1953)

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

## Equation of State Calculations by Fast Computing Machines

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

AND

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

(Received March 6, 1953)

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

### I. INTRODUCTION

THE purpose of this paper is to describe a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules. Classical statistics is assumed,

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

In order to reduce the problem to a feasible size for numerical work, we can, of course, consider only a finite number of particles. This number $N$ may be as high as several hundred. Our system consists of a square† con-



Saturday, January 21, 12

# Markov chain Monte Carlo

- Instead of drawing independent samples $c_i$ we build a Markov chain

$$c_1 \to c_2 \to \ldots \to c_i \to c_{i+1} \to \ldots$$

- Transition probabilities $W_{x,y}$ for transition $x \to y$ need to satisfy:
  - **Normalization:** $$\sum_y W_{x,y} = 1$$
  - **Ergodicity:** any configuration reachable from any other

$$\forall x,y \, \exists n \; : \; \left(W^n\right)_{x,y} \neq 0$$

  - **Balance:** the distribution should be stationary

$$0 = \frac{d}{dt} p(x) = \sum_y p(y) W_{y,x} - \sum_y p(x) W_{x,y} \Rightarrow p(x) = \sum_y p(y) W_{y,x}$$

  - Detailed balance is sufficient but not necessary for balance

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

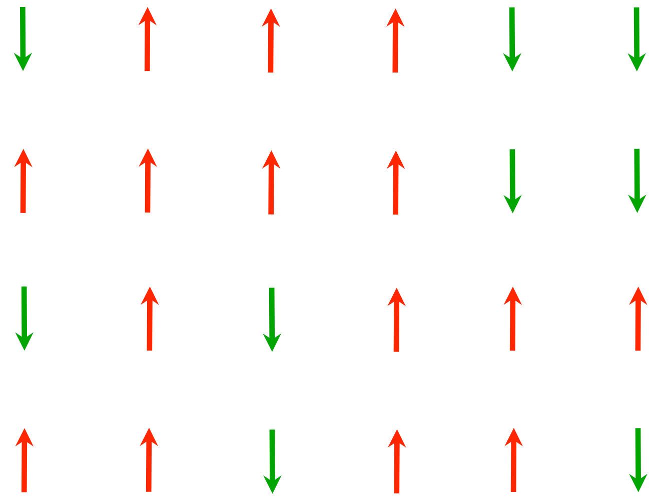
# The Metropolis algorithm

- Teller's proposal was to use rejection sampling:
  - Propose a change with an a-priori proposal rate $A_{x,y}$
  - Accept the proposal with a probability $P_{x,y}$
  - The total transition rate is $W_{x,y} = A_{x,y} P_{x,y}$
- The choice

$$P_{x,y} = \min\left[1, \frac{A_{y,x} p(y)}{A_{x,y} p(x)}\right]$$

satisfies detailed balance and was proposed by Metropolis *et al*

# Metropolis algorithm for the Ising model



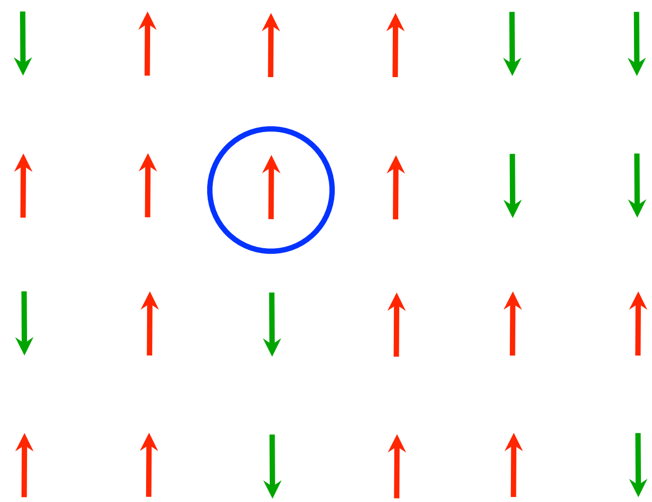
1. Pick a random spin and propose to flip it

2. Accept the flip with probability

$$P = \min\left[1, e^{-(E_{new} - E_{old})/T}\right]$$

3. Perform a measurement independent of whether the proposed flip was accepted or rejected!

# Metropolis algorithm for the Ising model



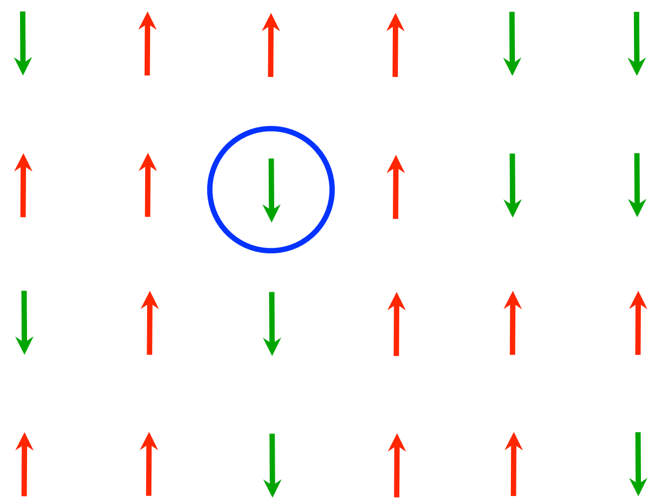
1. Pick a random spin and propose to flip it

2. Accept the flip with probability

$$P = \min\left[1, e^{-(E_{new} - E_{old})/T}\right]$$

3. Perform a measurement independent of whether the proposed flip was accepted or rejected!

# Metropolis algorithm for the Ising model



1. Pick a random spin and propose to flip it

2. Accept the flip with probability

$$P = \min\left[1, e^{-(E_{new} - E_{old})/T}\right]$$

3. Perform a measurement independent of whether the proposed flip was accepted or rejected!

# Equilibration

- Starting from a random initial configuration it takes a while to reach the equilibrium distribution
- The desired equilibrium distribution is a left eigenvector with eigenvalue 1 (this is just the balance condition)

$$p(x) = \sum_y p(y) W_{y,x}$$

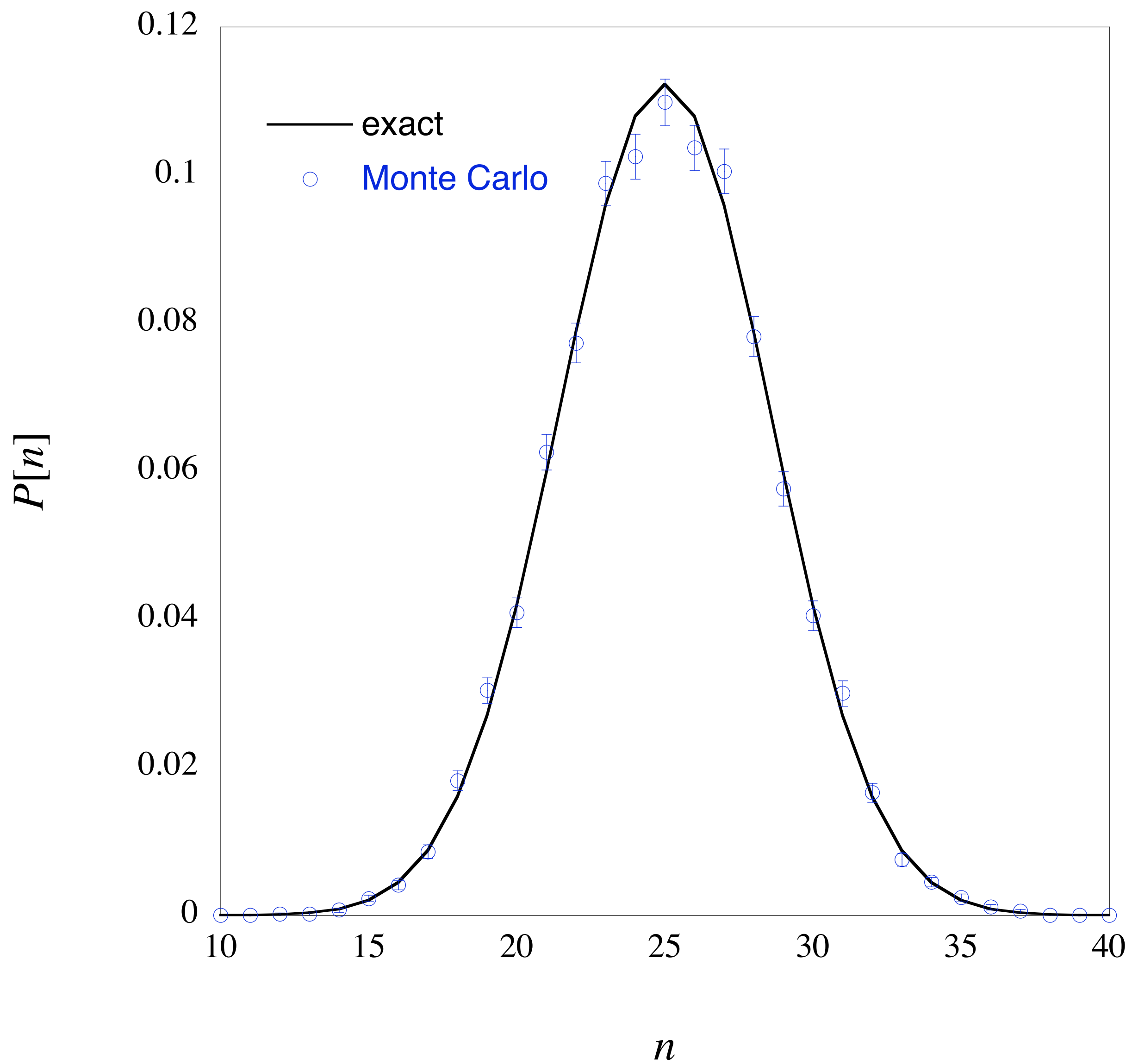
- Convergence is controlled by the s largest eigenvalue

$$p(x,t) = p(x) + O(\lambda_2^t)$$

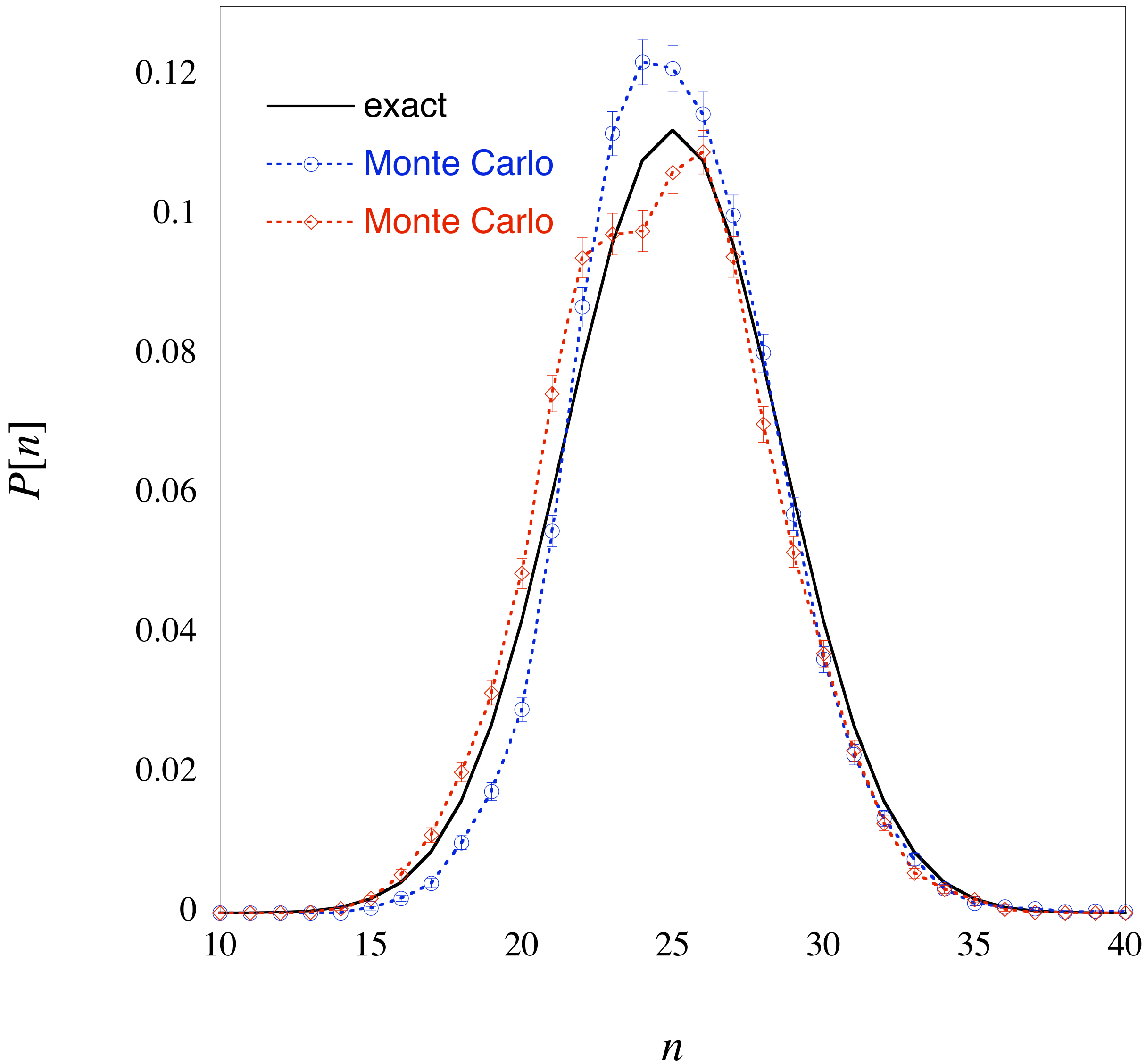
- We need to run the simulation for a while to equilibrate and only then start measuring

# 4. Monte Carlo Error Analysis

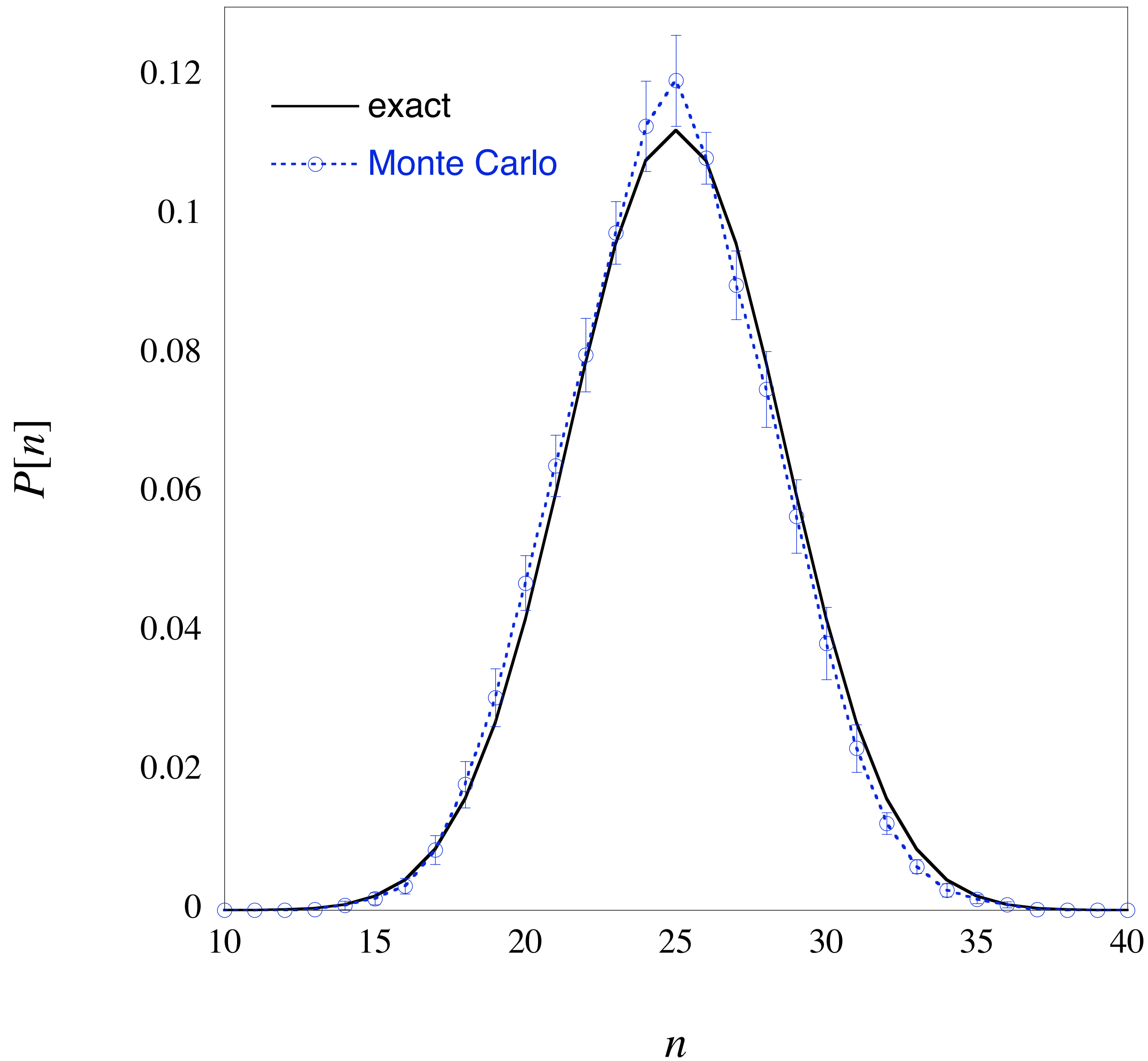
# Dogs and fleas: direct sampling



# Dogs and fleas: naïve errors



# Dogs and fleas: uncorrelated samples



# Monte Carlo error analysis

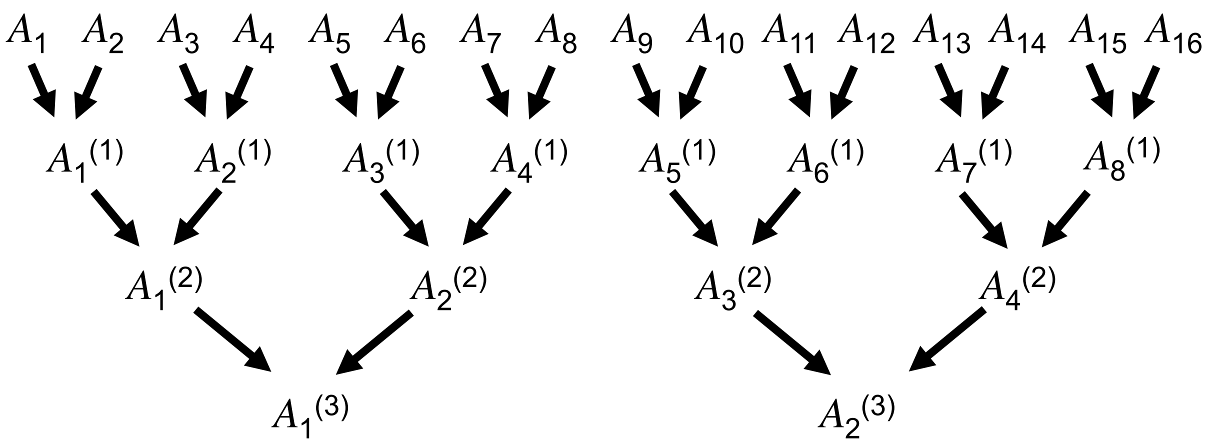
- The simple formula
$$\Delta A = \sqrt{\frac{\mathrm{Var}\, A}{M}}$$
is valid only for independent samples

- The Metropolis algorithm gives us correlated samples!
The number of independent samples is reduced
$$\Delta A = \sqrt{\frac{\mathrm{Var}\, A}{M}(1+2\tau_A)}$$

- The autocorrelation time is defined by
$$\tau_A = \frac{\sum_{t=1}^{\infty}\left(\langle A_{i+t}A_i\rangle - \langle A\rangle^2\right)}{\mathrm{Var}\, A}$$

# Binning analysis

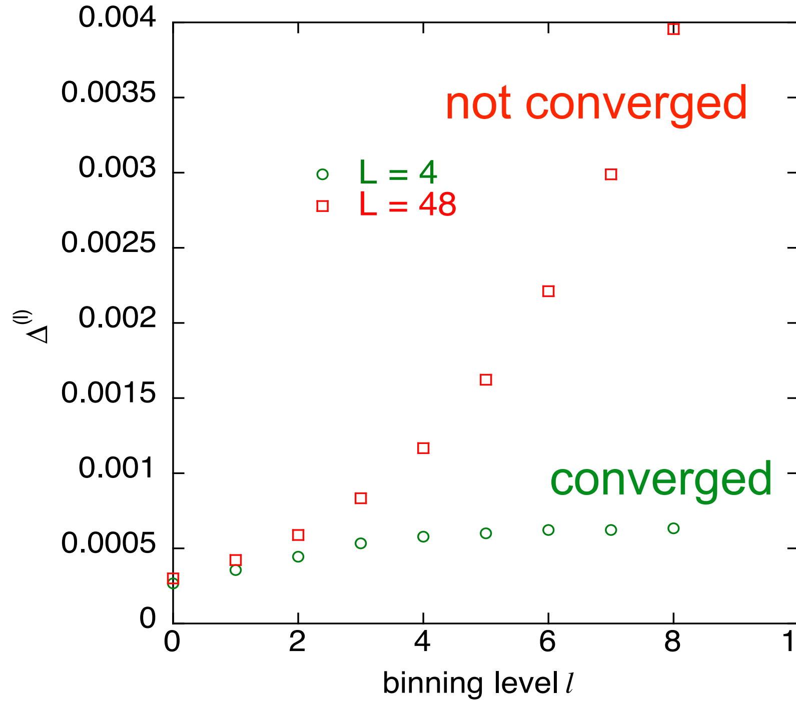
- Take averages of consecutive measurements: averages become less correlated and naive error estimates converge to real error




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

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

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




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

# Seeing convergence in ALPS

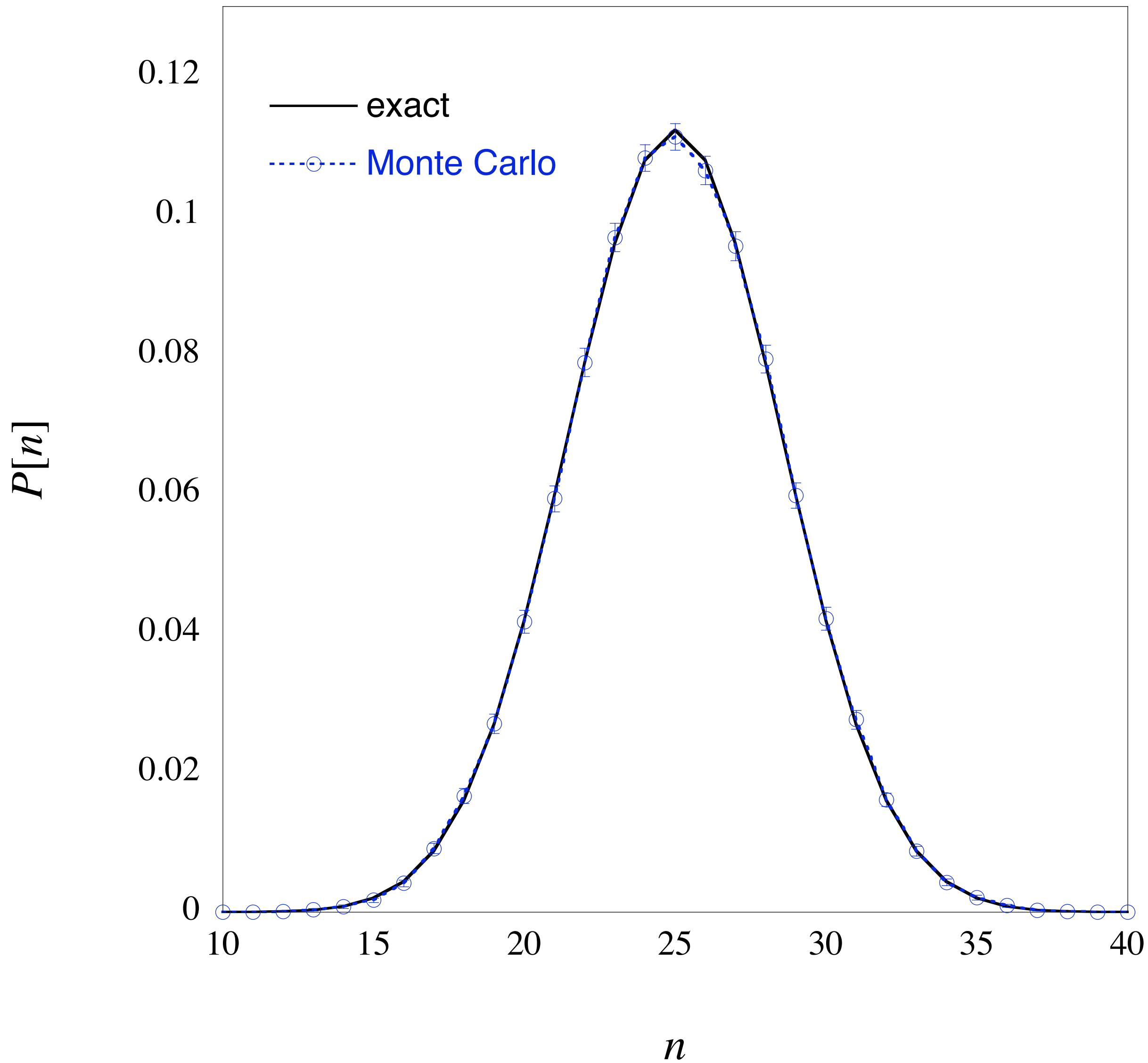
- Look at the ALPS output in the first hands-on session
- 48 x 48 Ising model at the critical point
  - local updates:

| Name | Count | Mean | Error | Tau | Method |
|---|---|---|---|---|---|
| **Susceptibility** | 52529 | 401.08 | 11.3<br>not converged | 99.1 | binning |

  - cluster updates:

| Name | Count | Mean | Error | Tau | Method |
|---|---|---|---|---|---|
| **Susceptibility** | 113433 | 421.642 | 1.57 | 0.821 | binning |

# Dogs and fleas: binning analysis



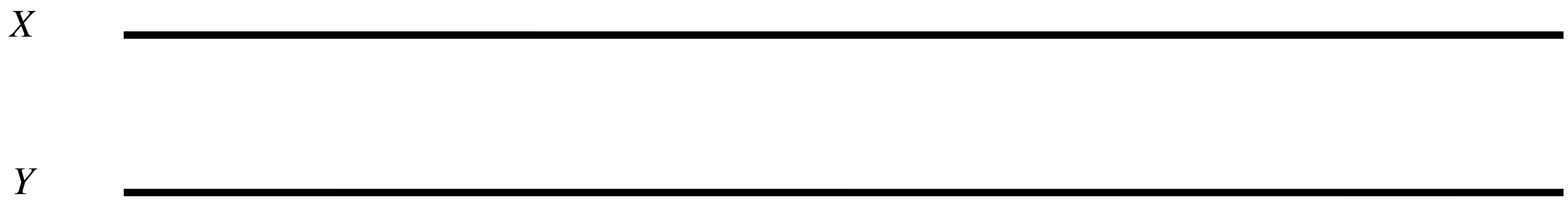
DPHYS
Department of Physics
Institute for Theoretical Physics

# Correlated quantities

- How do we calculate the errors of functions of correlated measurements?
  - specific heat $$c_V = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2}$$
  - Binder cumulant ratio $$U = \frac{\langle m^4 \rangle}{\langle m^2 \rangle^2}$$
- The naïve way of assuming uncorrelated errors is wrong!
- It is not even enough to calculate all crosscorrelations due to nonlinearities except if the errors are tiny!

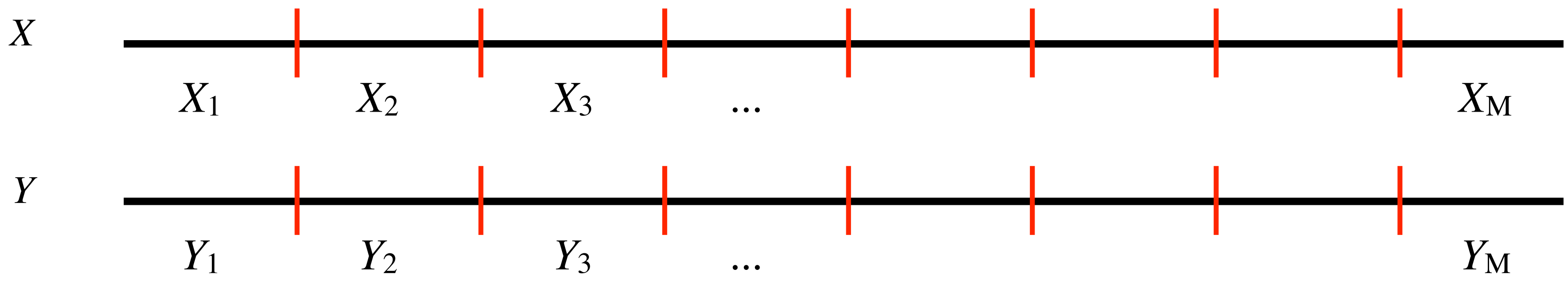
# Splitting the time series

Simplest idea: split the time series and evaluate for each segment



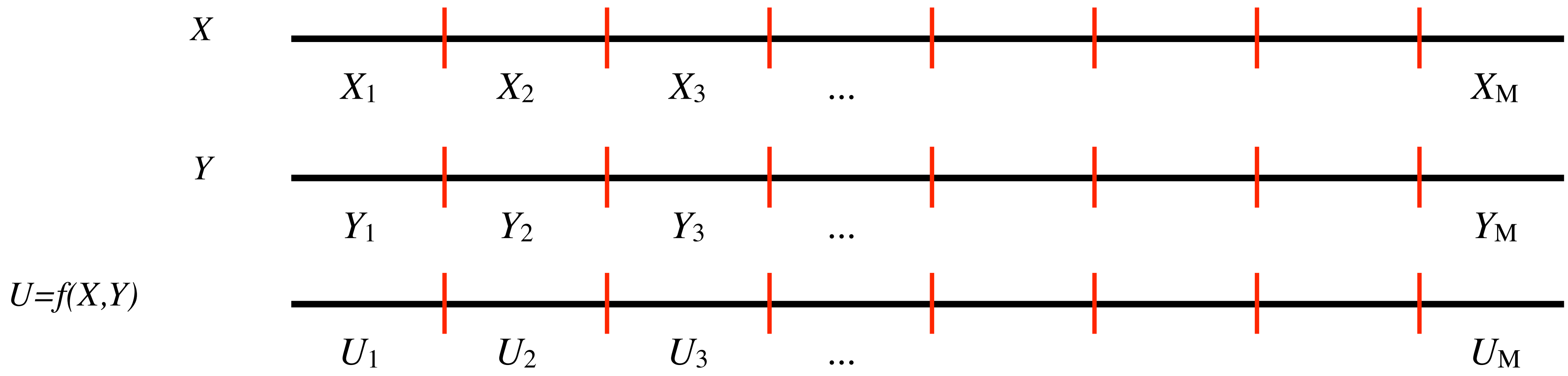
# Splitting the time series

Simplest idea: split the time series and evaluate for each segment



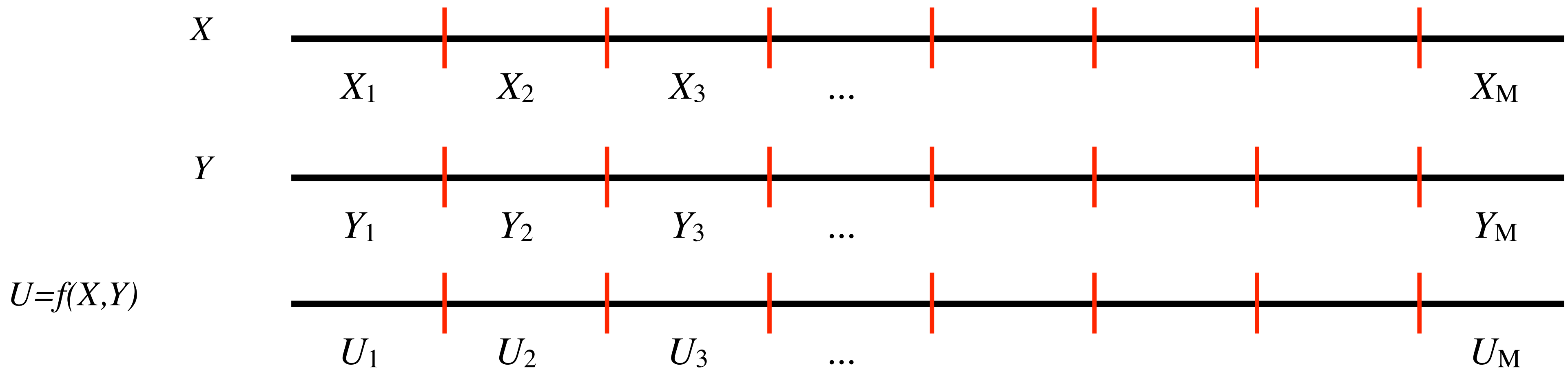
# Splitting the time series

Simplest idea: split the time series and evaluate for each segment



# Splitting the time series

Simplest idea: split the time series and evaluate for each segment

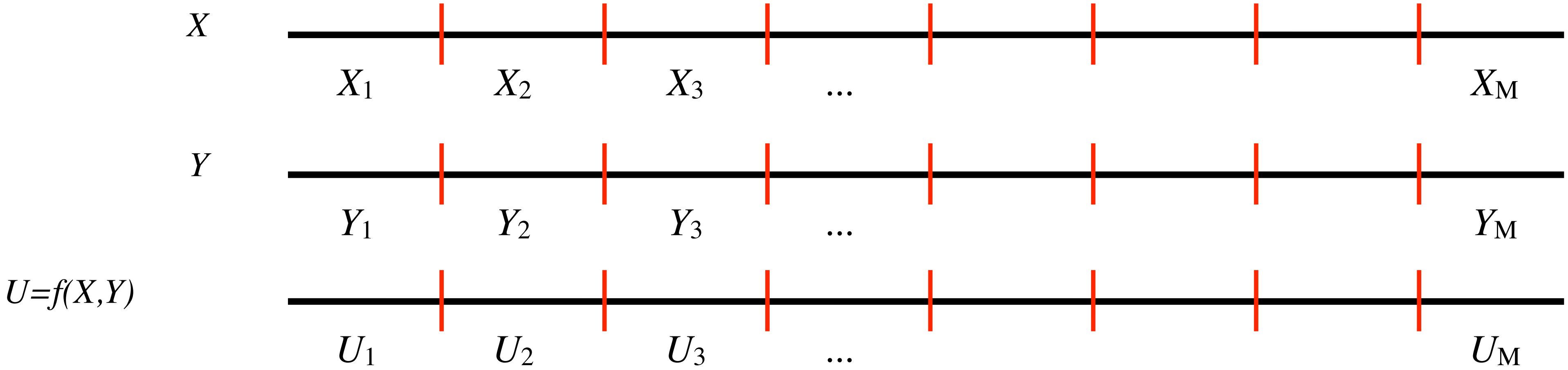



$$\langle U \rangle \approx \overline{U} = \frac{1}{M}\sum_{i=1}^{M} U_i$$

$$\Delta U \approx \sqrt{\frac{1}{M(M-1)}\sum_{i-1}^{M}\left(U_i - \overline{U}\right)^2}$$

# Splitting the time series

Simplest idea: split the time series and evaluate for each segment




$$\langle U \rangle \approx \overline{U} = \frac{1}{M} \sum_{i=1}^{M} U_i$$

$$\Delta U \approx \sqrt{\frac{1}{M(M-1)} \sum_{i-1}^{M} \left(U_i - \overline{U}\right)^2}$$

Problem: can be unstable and noisy for nonlinear functions such as $X/Y$

# Jackknife-analysis

Evaluate the function on all and all but one segment

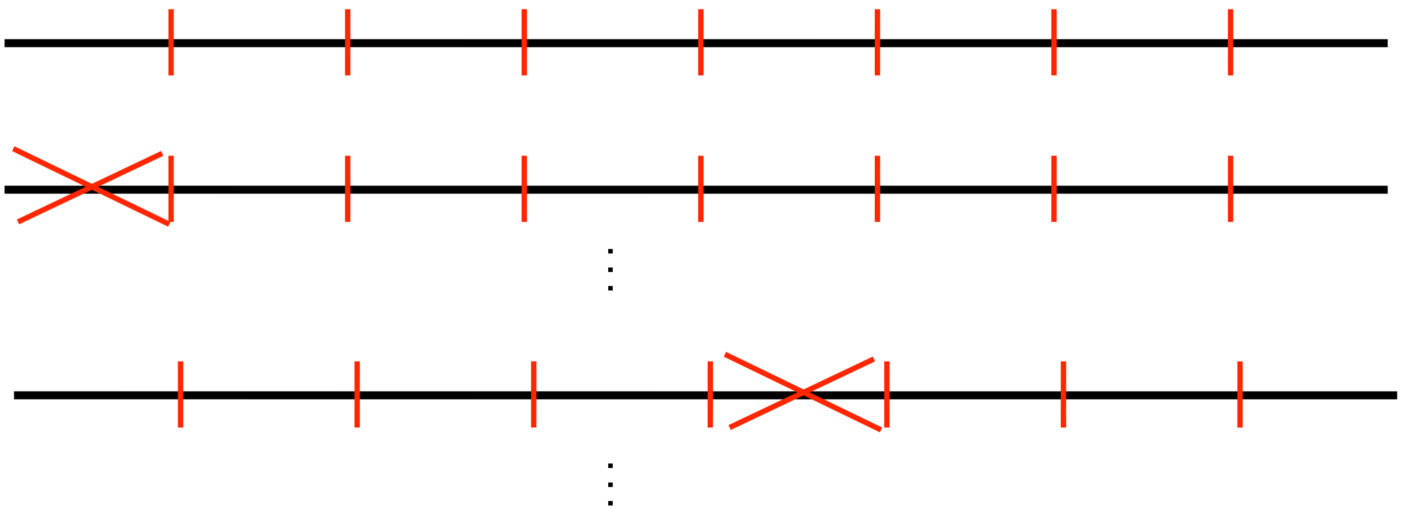
$$U_0 = f\left(\frac{1}{M}\sum_{i=1}^{M} X_i, \frac{1}{M}\sum_{i=1}^{M} Y_i\right)$$

$$U_1 = f\left(\frac{1}{M-1}\sum_{i=2}^{M} X_i, \frac{1}{M-1}\sum_{i=2}^{M} Y_i\right)$$

$$\vdots$$

$$U_j = f\left(\frac{1}{M-1}\sum_{\substack{i=1\\ i\neq j}}^{M} X_i, \frac{1}{M-1}\sum_{\substack{i=1\\ i\neq j}}^{M} Y_i\right)$$

$$\vdots$$



# Jackknife-analysis

Evaluate the function on all and all but one segment

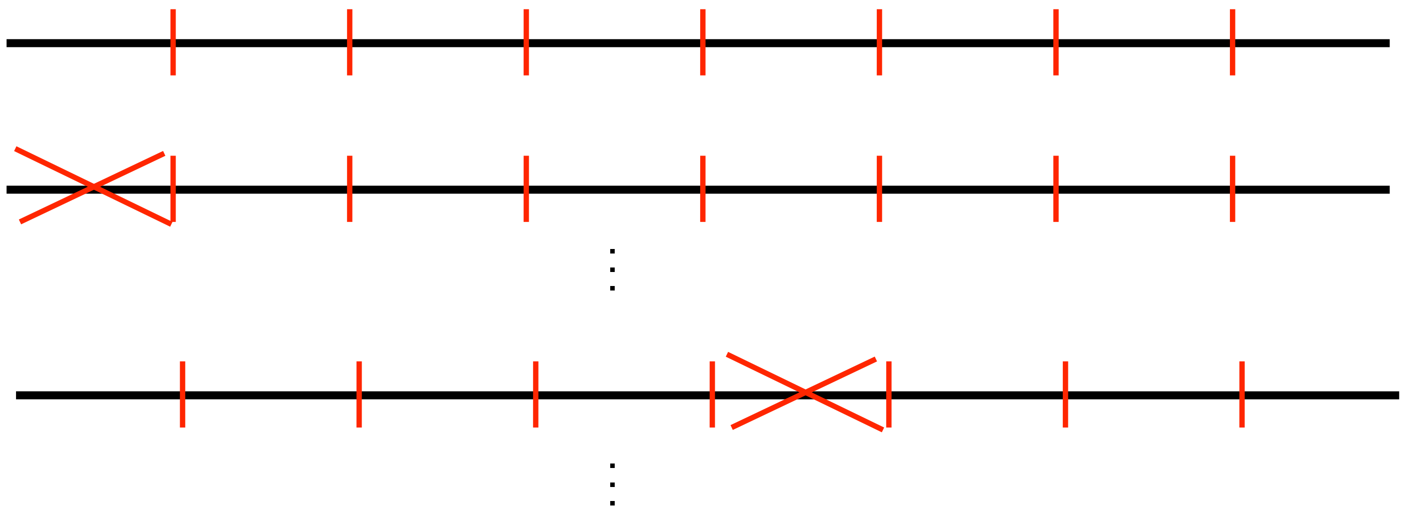
$$U_0 = f\left(\frac{1}{M}\sum_{i=1}^{M} X_i, \frac{1}{M}\sum_{i=1}^{M} Y_i\right)$$

$$U_1 = f\left(\frac{1}{M-1}\sum_{i=2}^{M} X_i, \frac{1}{M-1}\sum_{i=2}^{M} Y_i\right)$$

$$\vdots$$

$$U_j = f\left(\frac{1}{M-1}\sum_{\substack{i=1\\ i\neq j}}^{M} X_i, \frac{1}{M-1}\sum_{\substack{i=1\\ i\neq j}}^{M} Y_i\right)$$

$$\vdots$$



$$\langle U\rangle \approx U_0 - (M-1)(\overline{U} - U_0)$$

$$\overline{U} = \frac{1}{M}\sum_{i=1}^{M} U_i$$

$$\Delta U \approx \sqrt{\frac{M-1}{M}\sum_{i-1}^{M}(U_i - \overline{U})^2}$$

# ALPS Alea library in C++

- The ALPS class library implements reliable error analysis
  - Adding a measurement:

```
alps::RealObservable mag;
…
mag << new_value;
```

  - Evaluating measurements

```
std::cout << mag.mean() << " +/- " << mag.error();
std::cout "Autocorrelation time: " << mag.tau();
```

- Correlated quantities?
  - Such as in Binder cumulant ratios $\langle m^4 \rangle / \langle m^2 \rangle^2$
  - ALPS library uses jackknife analysis to get correct errors

```
alps::RealObsEvaluator binder = mag4/(mag2*mag2);
std::cout << binder.mean() << " +/- " << binder.error();
```

# ALPS Alea library in Python

- The ALPS class library implements reliable error analysis
  - Adding a measurement:

```
mag = pyalps.pyalea.RealObservable('Magnetization');
…
mag << new_value;
```

  - Evaluating measurements

```
print mag.mean, " +/- ", mag.error;
print "Autocorrelation time: ", mag.tau;
```

- Correlated quantities?
  - Such as in Binder cumulant ratios $\langle m^4 \rangle / \langle m^2 \rangle^2$
  - ALPS library uses jackknife analysis to get correct errors of functions of data, after reading data from file

```
print mag4/(mag2*mag2)
```

# The Swendsen-Wang algorithm

# Autocorrelation effects

- The Metropolis algorithm creates a Markov chain

$$c_1 \to c_2 \to \ldots \to c_i \to c_{i+1} \to \ldots$$

- successive configurations are correlated, leading to an increased statistical error

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

- *Critical slowing down* at second order phase transition

$$\tau \propto L^2$$

- Exponential tunneling problem at first order phase transition

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

# From local to cluster updates



- Energy of configurations in Ising model
  - $-J$ if parallel: ↑ ↑   ↓ ↓
  - $+J$ if anti-parallel: ↓ ↑   ↓ ↑

- Probability for flip
  - Anti-parallel: flipping lowers energy, always accepted

    (↓) ↑ ⟶ ↑ ↑   $$\Delta E = -2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = 1$$

  - Parallel: (↑) ↑ ⟶ ↓ ↑   $$\Delta E = +2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = \exp(-2\beta J)$$

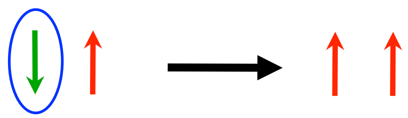
    no change with probability $1 - \exp(-2\beta J)$ !!!

# From local to cluster updates



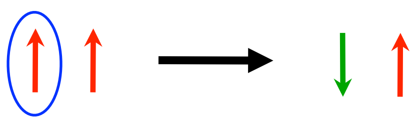
- Energy of configurations in Ising model
  - – *J* if parallel:
    
  - + *J* if anti-parallel:
    

- Probability for flip
  - Anti-parallel: flipping lowers energy, always accepted



$$\Delta E = -2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = 1$$

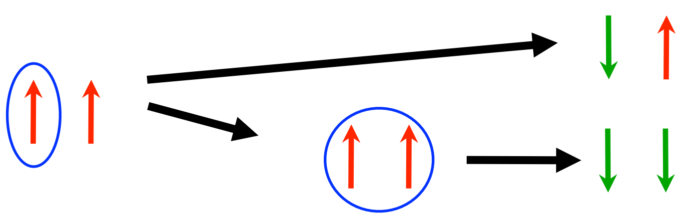
  - Parallel:



$$\Delta E = +2J \Rightarrow P = \min\left(1, e^{-2\Delta E/T}\right) = \exp(-2\beta J)$$

no change with probability $1-\exp(-2\beta J)$ !!!

Alternative: flip both!



$$P = \exp(-2J/T)$$

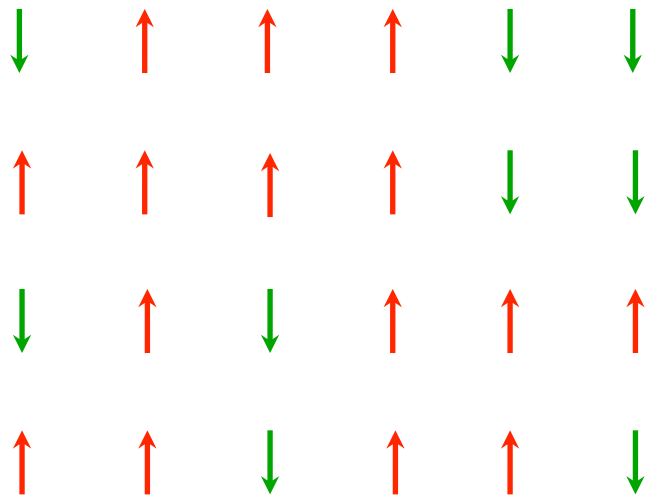
$$P = 1-\exp(-2J/T)$$

# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates

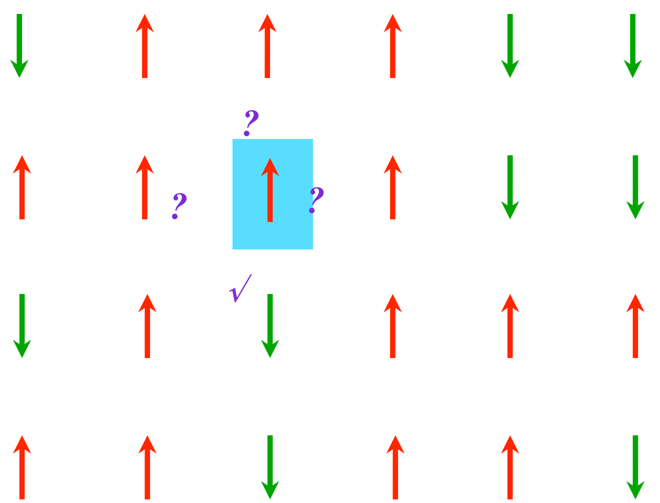
# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor!
    - Repeat for all flipped spins => cluster updates $P = 1 - \exp(-2\beta J)$



# Swendsen-Wang Cluster-Updates

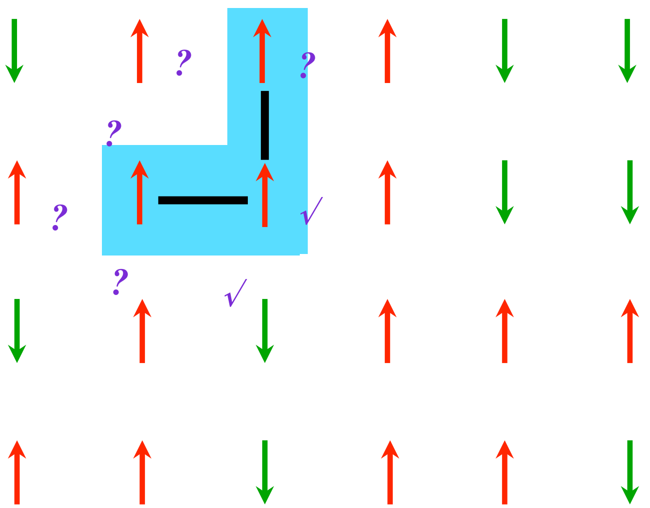
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates




Shall we flip neighbor?

# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



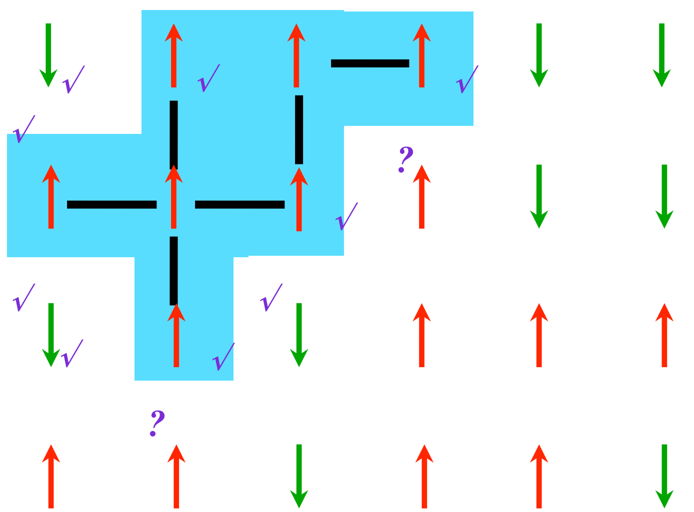

Shall we flip neighbor?

# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with
    - Otherwise: also flip neighbor!
    - Repeat for all flipped spins => cluster updates

$$P = \exp(-2\beta J)$$

$$P = 1 - \exp(-2\beta J)$$



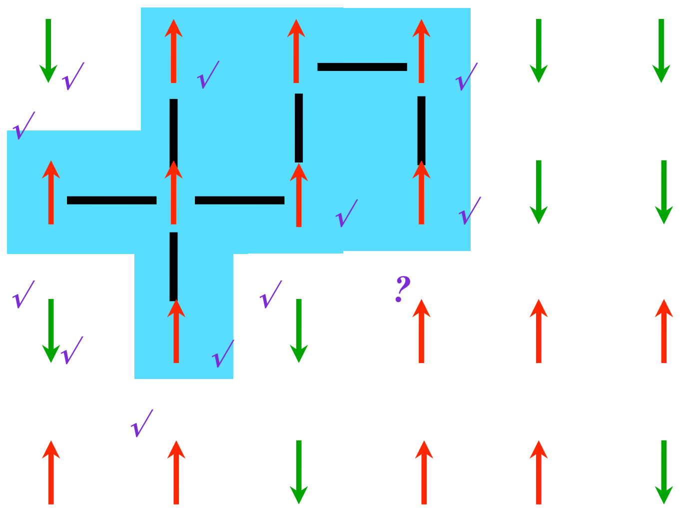
Shall we flip neighbor?



Saturday, January 21, 12

# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



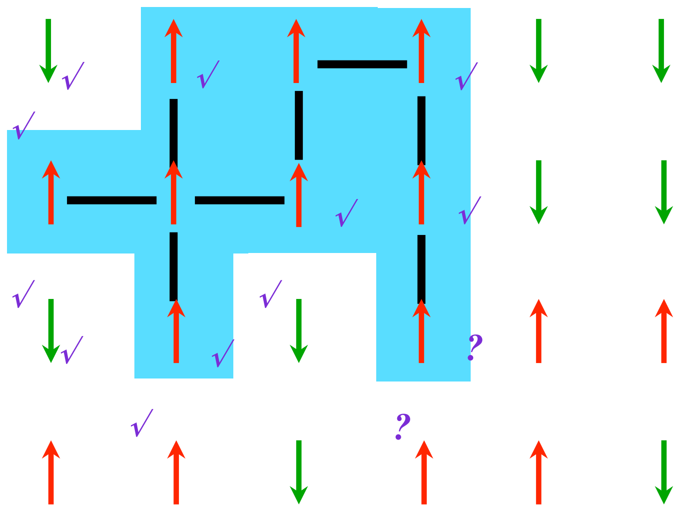
Shall we flip neighbor?

# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with
    - Otherwise: also flip neighbor!
    - Repeat for all flipped spins => cluster updates

$$P = \exp(-2\beta J)$$

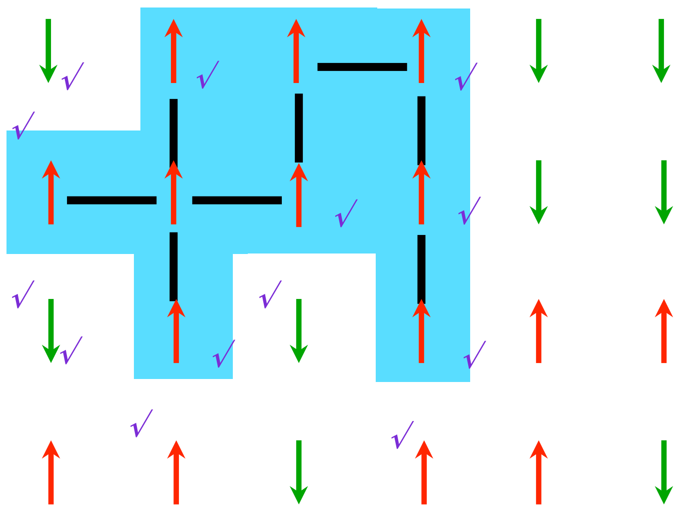
$$P = 1 - \exp(-2\beta J)$$



Shall we flip neighbor?

# Swendsen-Wang Cluster-Updates

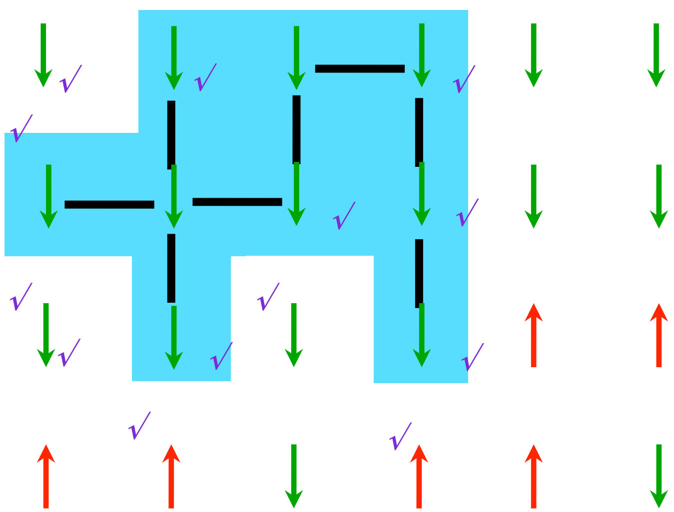
- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: “do we want to flip it against its neighbor?”
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Shall we flip neighbor?

# Swendsen-Wang Cluster-Updates

- No critical slowing down (Swendsen and Wang, 1987) !!!
- Ask for each spin: "do we want to flip it against its neighbor?"
  - antiparallel: yes
  - parallel: costs energy
    - Accept with $P = \exp(-2\beta J)$
    - Otherwise: also flip neighbor! $P = 1 - \exp(-2\beta J)$
    - Repeat for all flipped spins => cluster updates



Done building cluster

Flip all spins in cluster



Saturday, January 21, 12

# 6. Quantum Monte Carlo

# Quantum Monte Carlo

- Not as easy as classical Monte Carlo

$$Z = \sum_c e^{-E_c / k_B T}$$

- Calculating the eigenvalues $E_c$ is equivalent to solving the problem

- Need to find a mapping of the quantum partition function to a classical problem

$$Z = \mathrm{Tr}\, e^{-\beta H} \equiv \sum_c p_c$$

- "Negative sign" problem if some $p_c < 0$

# Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines

THE
PHYSICAL REVIEW

*A journal of experimental and theoretical physics established by E. L. Nichols in 1893*

SECOND SERIES, VOL. 91, NO. 6 SEPTEMBER 15, 1953

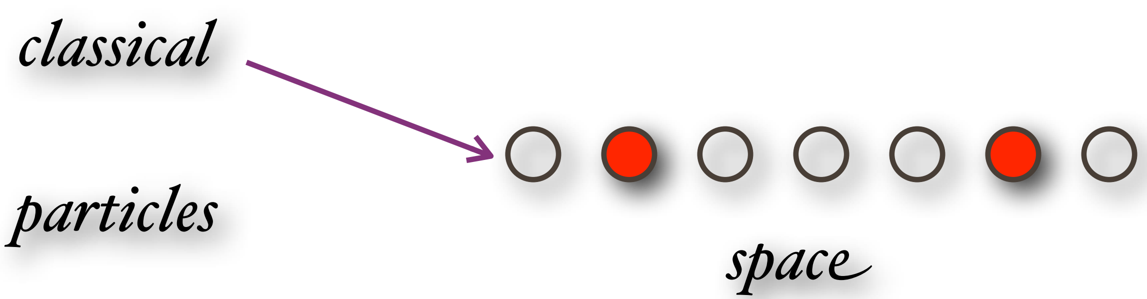
## Atomic Theory of the λ Transition in Helium

R. P. FEYNMAN
*California Institute of Technology, Pasadena, California*

(Received May 15, 1953)

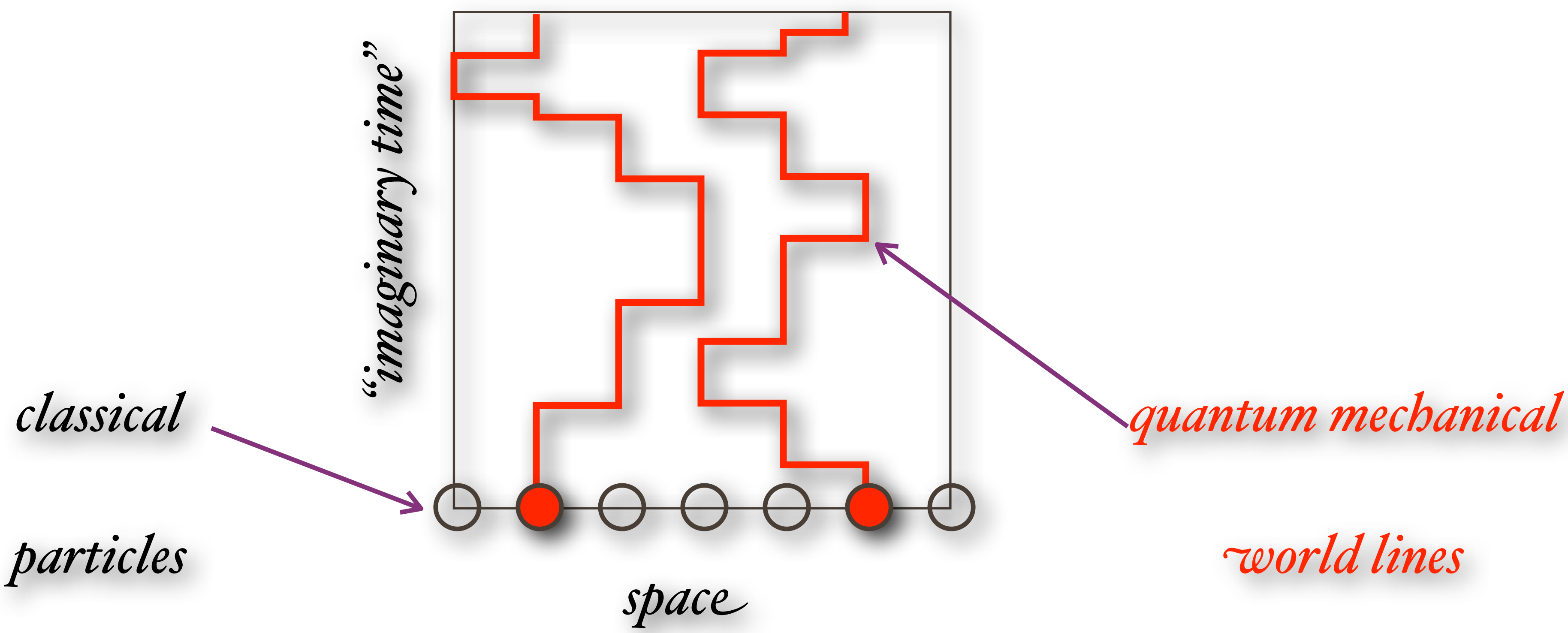
# Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines



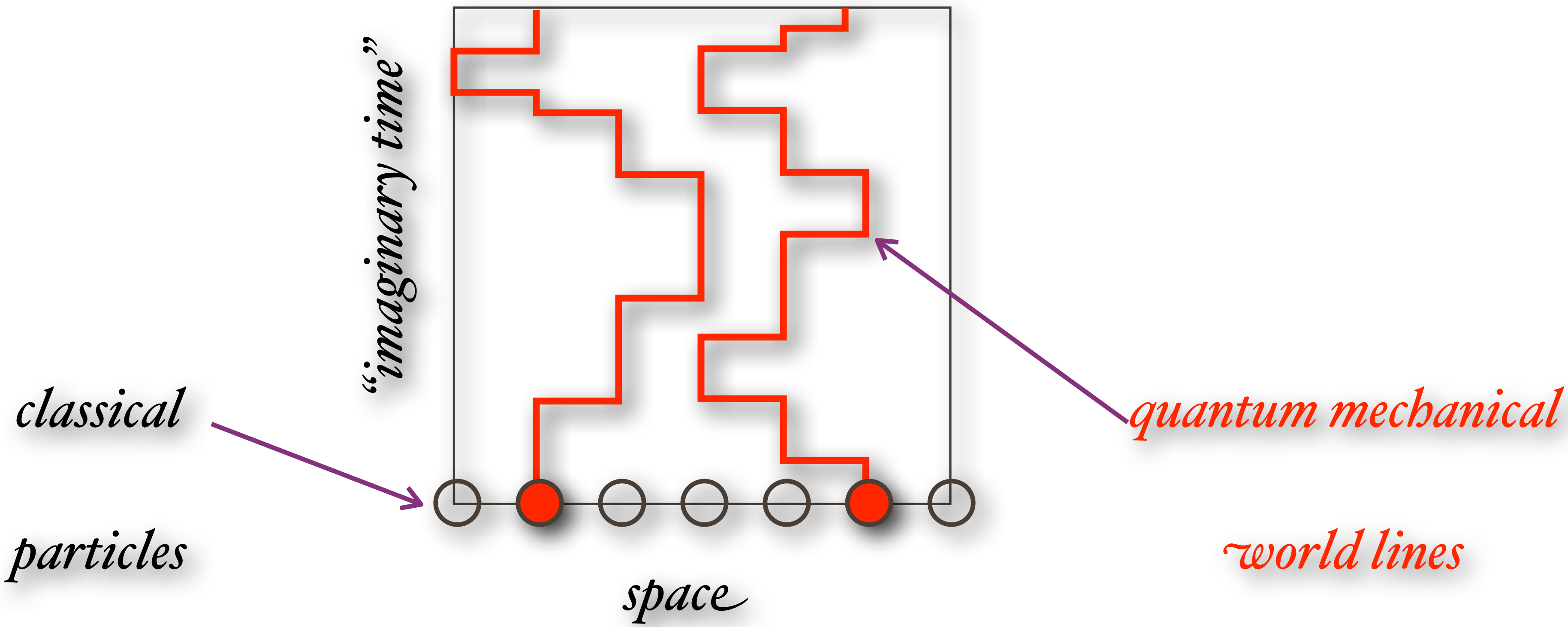
# Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines



# Quantum Monte Carlo

- Feynman (1953) lays foundation for quantum Monte Carlo
- Map quantum system to classical world lines




Use Metropolis algorithm to update world lines

# Diagrammatic QMC

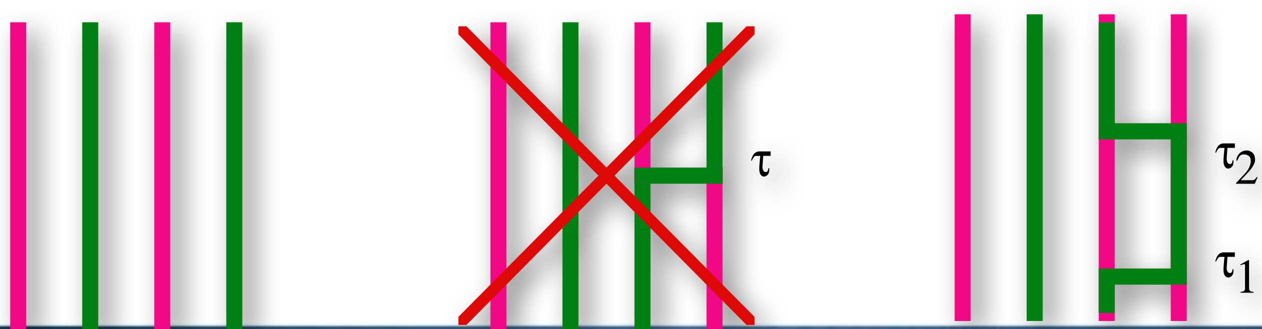
- Split the Hamiltonian into diagonal term *H₀* and perturbation *V*
- Then perform time-dependent perturbation theory

$$H = H_0 + V, \quad H_0 = \sum_{<i,j>} J_{ij}^z S_i^z S_j^z - \sum_i h S_i^z, \quad V = \sum_{<i,j>} J_{ij}^{xy} (S_i^x S_j^x + S_i^y S_j^y)$$

$$Z = \mathrm{Tr}(e^{-\beta H}) = \mathrm{Tr}(e^{-\beta H_0} \mathsf{T} e^{-\int_0^\beta d\tau V(\tau)})$$

$$Z = \mathrm{Tr}(e^{-\beta H_0} (1 - \int_0^\beta d\tau V(\tau) + \int_0^\beta d\tau_1 \int_{\tau_1}^\beta d\tau_2 V(\tau_1) V(\tau_2) + \ldots))$$

- Each term is represented by a diagram (world line configuration)



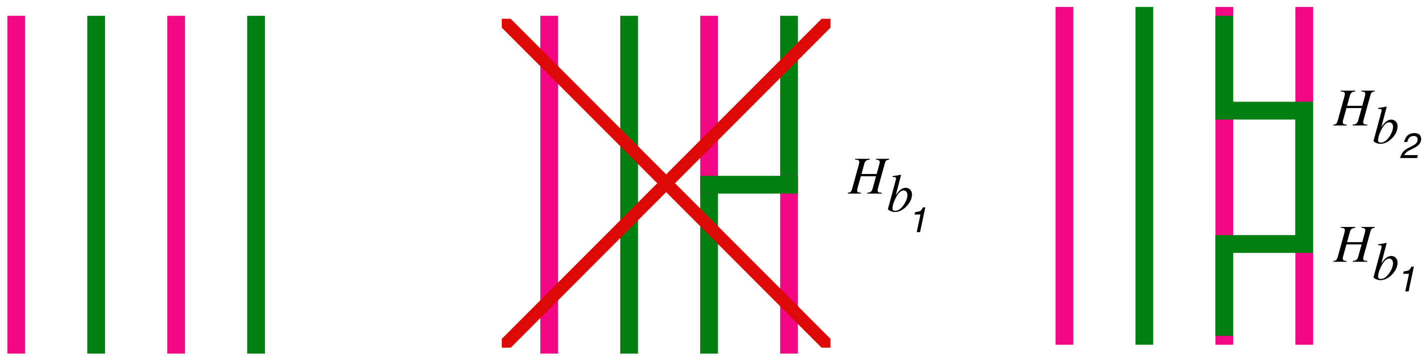
# Stochastic Series Expansion

- based on high temperature expansion, developed by Sandvik

$$Z = \mathrm{Tr}(e^{-\beta H}) = \sum_{n=0}^{\infty} (-\beta)^n \, \mathrm{Tr}(H^n)$$

$$= \sum_{n=0}^{\infty} \frac{\beta^n}{n!} \sum_{|\alpha\rangle} \sum_{(b_1,\ldots,b_n)} \langle \alpha | \prod_{i=1}^{n} (-H_{b_i}) | \alpha \rangle$$

with $H = \sum_i H_i$




- Similar world line representation but without times assigned



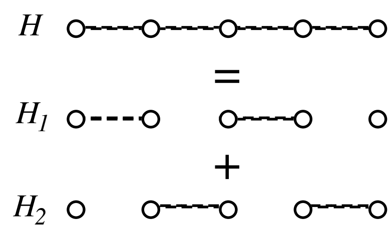
Saturday, January 21, 12

# The Suzuki-Trotter Decomposition

- Generic mapping of a quantum spin system to Ising model
  - basis of most discrete time QMC algorithms
  - not limited to special models
- Split Hamiltonian into two easily diagonalized pieces

$$H = H_1 + H_2$$

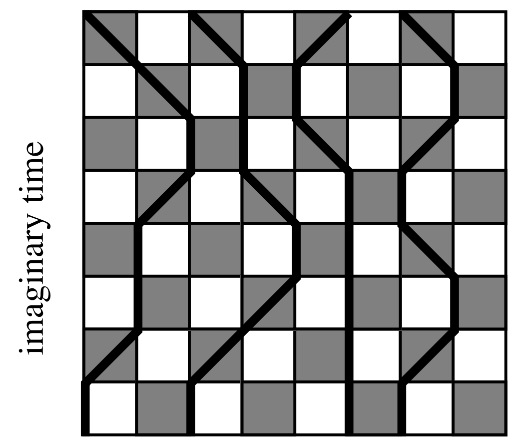
$$e^{-\varepsilon H} = e^{-\varepsilon(H_1+H_2)} = e^{-\varepsilon H_1} e^{-\varepsilon H_2} + O(\varepsilon^2)$$




- Obtain the checkerboard decomposition

$$Z = \mathrm{Tr}[\exp(-\beta H)] = \mathrm{Tr}\left[e^{-\beta(H_1+H_2)}\right]$$

$$= \mathrm{Tr}\left[e^{-(\beta/M)H_1} e^{-(\beta/M)H_2}\right]^M + O(\beta^3/M^2)$$



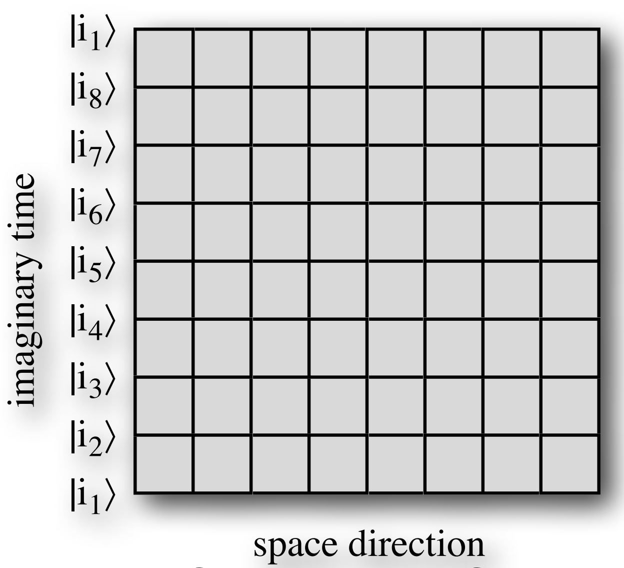
# Path integral QMC

- Use Trotter-Suzuki or a a simple low-order formula

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \ldots i_M)\}} \langle i_1 | 1-\Delta\tau H | i_2 \rangle \langle i_2 | 1-\Delta\tau H | i_3 \rangle \cdots \langle i_M | 1-\Delta\tau H | i_1 \rangle$$

- gives a mapping to a ($d$+1)-dimensional classical model




- partition function of quantum system is sum over classical world lines

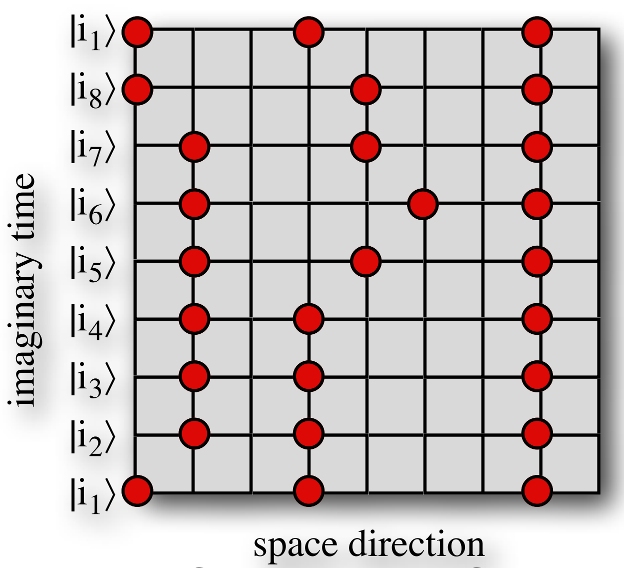
# Path integral QMC

- Use Trotter-Suzuki or a a simple low-order formula

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \ldots i_M)\}} \langle i_1 | 1-\Delta\tau H | i_2 \rangle \langle i_2 | 1-\Delta\tau H | i_3 \rangle \cdots \langle i_M | 1-\Delta\tau H | i_1 \rangle$$

- gives a mapping to a ($d$+1)-dimensional classical model




place particles (spins)

- partition function of quantum system is sum over classical world lines

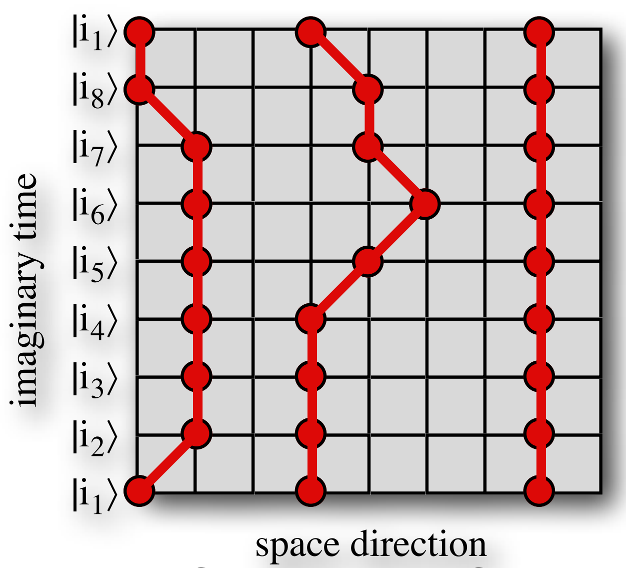
# Path integral QMC

- Use Trotter-Suzuki or a a simple low-order formula

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \ldots i_M)\}} \langle i_1|1-\Delta\tau H|i_2\rangle\langle i_2|1-\Delta\tau H|i_3\rangle\cdots\langle i_M|1-\Delta\tau H|i_1\rangle$$

- gives a mapping to a ($d$+1)-dimensional classical model




place particles (spins)

for Hamiltonians conserving particle number (magnetization) we get world lines

- partition function of quantum system is sum over classical world lines

# Calculating configuration weights

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \ldots i_M)\}} \langle i_1 | 1-\Delta\tau H | i_2 \rangle \langle i_2 | 1-\Delta\tau H | i_3 \rangle \cdots \langle i_M | 1-\Delta\tau H | i_1 \rangle$$

- Examples: particles with nearest neighbor repulsion

$$H = -t \sum_{\langle i,j \rangle} (a_i^\dagger a_j + a_j^\dagger a_i) + V \sum_{\langle i,j \rangle} n_i n_j$$

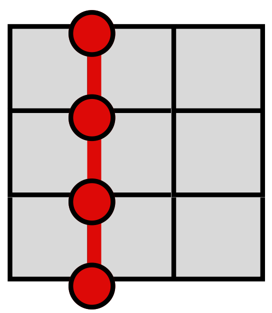
# Calculating configuration weights

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \dots i_M)\}} \langle i_1|1-\Delta\tau H|i_2\rangle\langle i_2|1-\Delta\tau H|i_3\rangle\cdots\langle i_M|1-\Delta\tau H|i_1\rangle$$

- Examples: particles with nearest neighbor repulsion

$$H = -t\sum_{\langle i,j\rangle}(a_i^\dagger a_j + a_j^\dagger a_i) + V\sum_{\langle i,j\rangle} n_i n_j$$



1



Saturday, January 21, 12

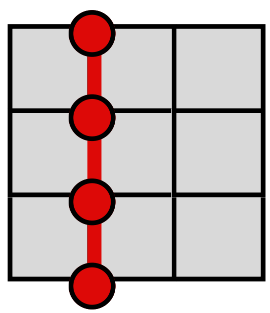
# Calculating configuration weights

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

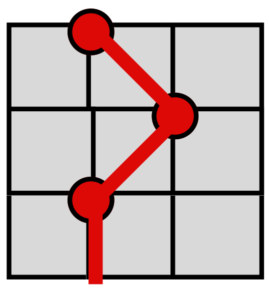
$$= \sum_{\{(i_1 \dots i_M)\}} \langle i_1|1-\Delta\tau H|i_2\rangle\langle i_2|1-\Delta\tau H|i_3\rangle\cdots\langle i_M|1-\Delta\tau H|i_1\rangle$$

- Examples: particles with nearest neighbor repulsion

$$H = -t\sum_{\langle i,j\rangle}(a_i^\dagger a_j + a_j^\dagger a_i) + V\sum_{\langle i,j\rangle} n_i n_j$$



$1$



$(\Delta\tau t)^2$

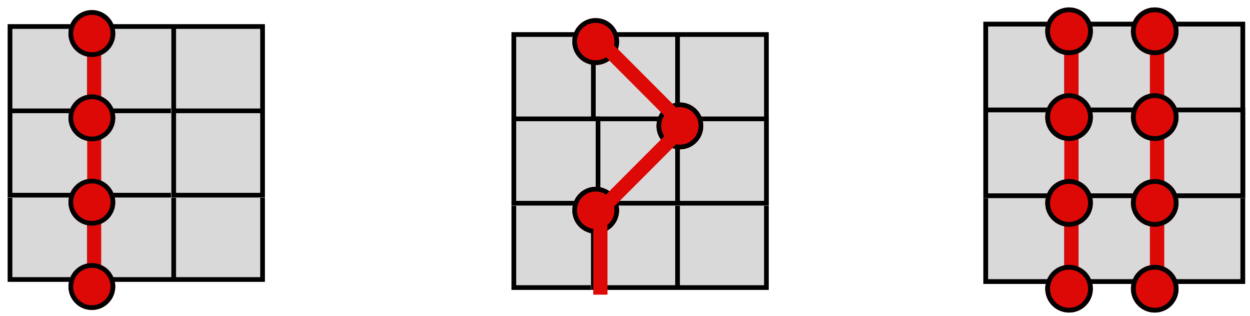
# Calculating configuration weights

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \dots i_M)\}} \langle i_1 | 1-\Delta\tau H | i_2 \rangle \langle i_2 | 1-\Delta\tau H | i_3 \rangle \cdots \langle i_M | 1-\Delta\tau H | i_1 \rangle$$

- Examples: particles with nearest neighbor repulsion

$$H = -t \sum_{\langle i,j \rangle} (a_i^\dagger a_j + a_j^\dagger a_i) + V \sum_{\langle i,j \rangle} n_i n_j$$



$1$ $(\Delta\tau t)^2$ $(1-\Delta\tau V)^3$

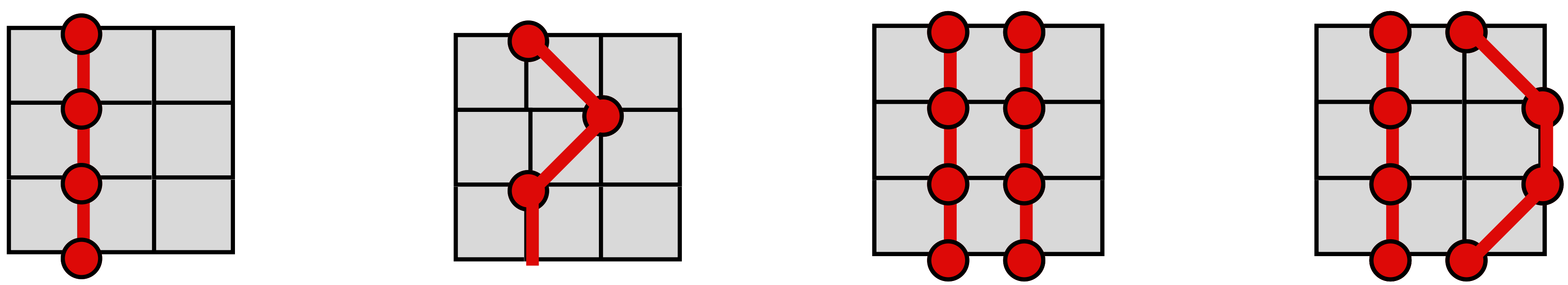
# Calculating configuration weights

$$Z = \mathrm{Tr}\, e^{-\beta H} = \mathrm{Tr}\, e^{-M\Delta\tau H} = \mathrm{Tr}\left(e^{-\Delta\tau H}\right)^M = \mathrm{Tr}(1-\Delta\tau H)^M + O(\beta\Delta\tau)$$

$$= \sum_{\{(i_1 \ldots i_M)\}} \langle i_1 | 1-\Delta\tau H | i_2 \rangle \langle i_2 | 1-\Delta\tau H | i_3 \rangle \cdots \langle i_M | 1-\Delta\tau H | i_1 \rangle$$

- Examples: particles with nearest neighbor repulsion

$$H = -t \sum_{\langle i,j \rangle} (a_i^\dagger a_j + a_j^\dagger a_i) + V \sum_{\langle i,j \rangle} n_i n_j$$



$1$ | $(\Delta\tau t)^2$ | $(1-\Delta\tau V)^3$ | $(\Delta\tau t)^2$

# Monte Carlo updates

- just move the world lines locally
  - probabilities given by matrix element of Hamiltonian
  - example: tight binding model

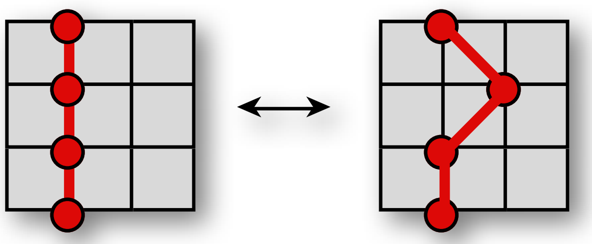
$$H = -t \sum_{\langle i,j \rangle} \left( c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right)$$

# Monte Carlo updates

- just move the world lines locally
  - probabilities given by matrix element of Hamiltonian
  - example: tight binding model

$$H = -t \sum_{\langle i,j \rangle} \left( c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right)$$

introduce or remove two kinks:

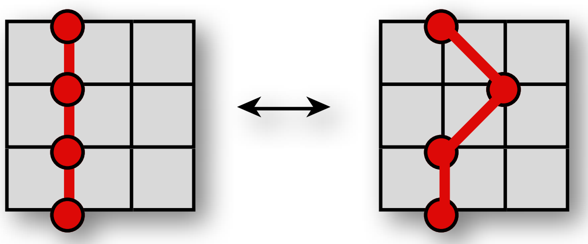


# Monte Carlo updates

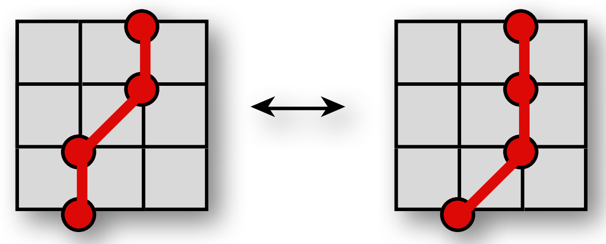
- just move the world lines locally
  - probabilities given by matrix element of Hamiltonian
  - example: tight binding model

$$H = -t \sum_{\langle i,j \rangle} \left( c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right)$$

introduce or remove two kinks:



shift a kink:

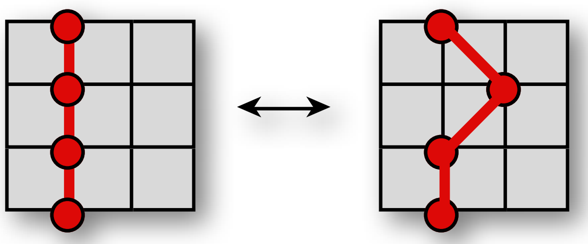


# Monte Carlo updates

- just move the world lines locally
  - probabilities given by matrix element of Hamiltonian
  - example: tight binding model

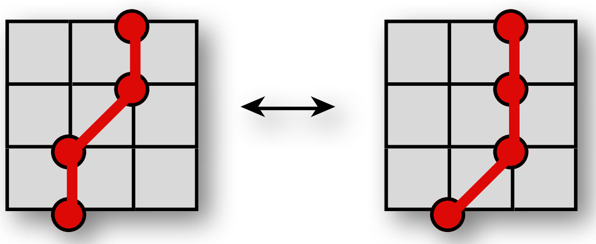
$$H = -t \sum_{\langle i,j \rangle} \left( c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right)$$

introduce or remove two kinks:



$P = 1$ $\qquad$ $P = (\Delta\tau t)^2$

shift a kink:



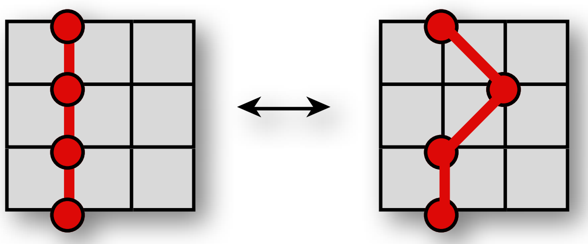
$P = \Delta\tau t$

# Monte Carlo updates

- just move the world lines locally
  - probabilities given by matrix element of Hamiltonian
  - example: tight binding model

$$H = -t \sum_{\langle i,j \rangle} \left( c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i \right)$$

introduce or remove two kinks:

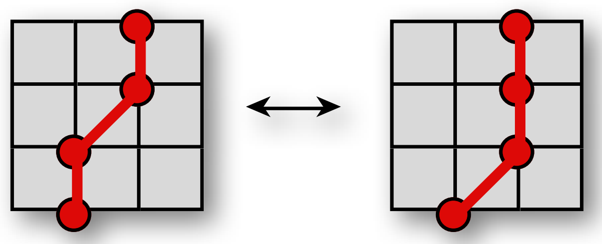


$$P = 1 \qquad P = (\Delta \tau t)^2$$

$$P_{\rightarrow} = \min\left[1, (\Delta \tau t)^2\right]$$

$$P_{\leftarrow} = \min\left[1, 1/(\Delta \tau t)^2\right]$$

shift a kink:



$$P = \Delta \tau t$$

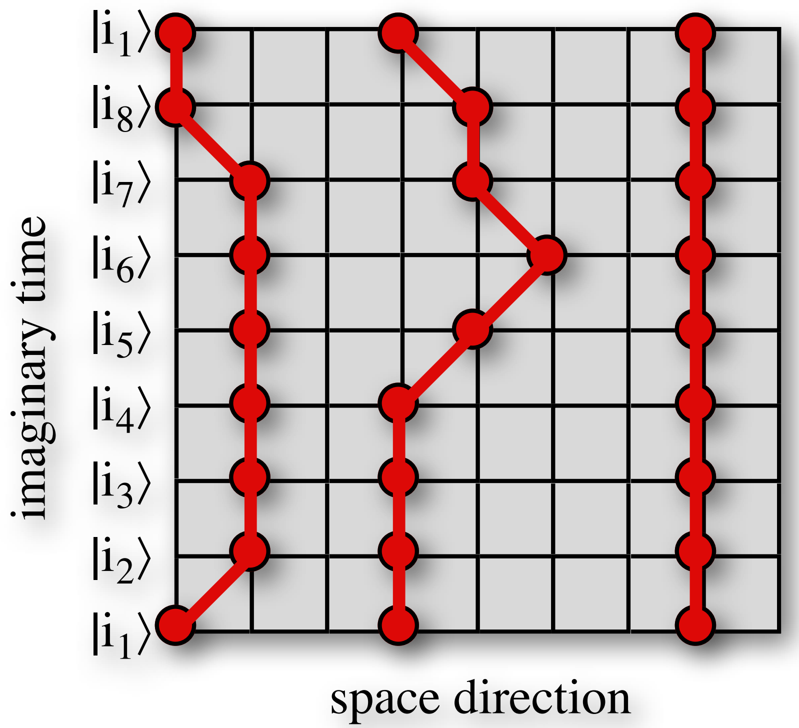
$$P_{\rightarrow} = P_{\leftarrow} = 1$$

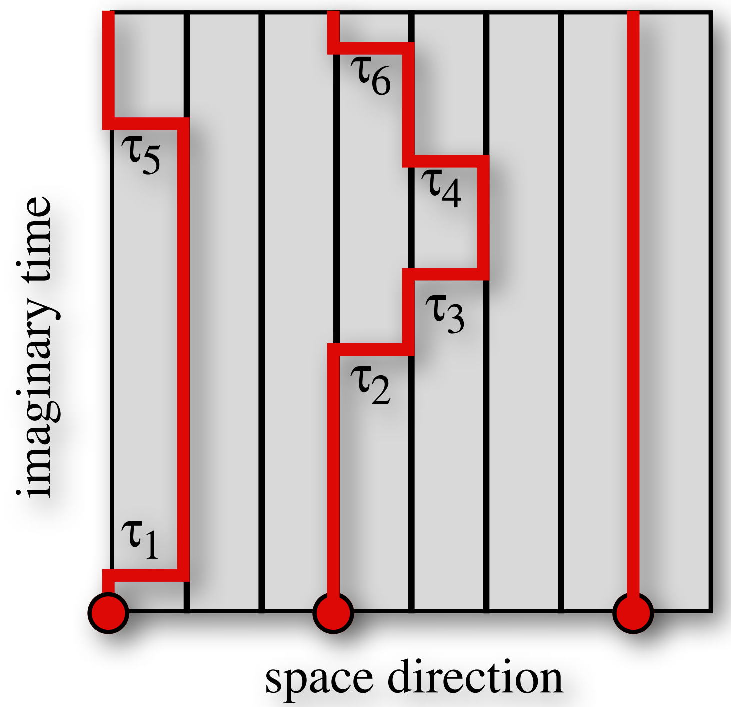
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DPHYS
Department of Physics
Institute for Theoretical Physics

# The continuous time limit

- the limit $\Delta\tau \rightarrow 0$ can be taken in the algorithm
  [Prokof'ev *et al.*, Pis'ma v Zh.Eks. Teor. Fiz. **64**, 853 (1996)]




- discrete time: store configuration at all time steps
- continuous time: store times at which configuration changes

# Calculating configuration weights

- Continuous time algorithms just sample time-dependent perturbation expansion

$$Z = \mathrm{Tr}\left(e^{-\beta H_0}\mathcal{T} e^{-\int_0^\beta d\tau \mathcal{V}(\tau)}\right)$$

- Examples: particles with nearest neighbor repulsion

$$H_0 = V\sum_{\langle i,j\rangle} n_i n_j \qquad \mathcal{V} = -t\sum_{\langle i,j\rangle}(a_i^\dagger a_j + a_j^\dagger a_i)$$

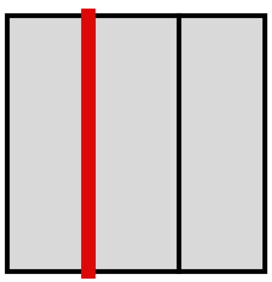
# Calculating configuration weights

- Continuous time algorithms just sample time-dependent perturbation expansion

$$Z = \mathrm{Tr}\left(e^{-\beta H_0} \mathcal{T} e^{-\int_0^\beta d\tau \mathcal{V}(\tau)}\right)$$

- Examples: particles with nearest neighbor repulsion

$$H_0 = V \sum_{\langle i,j \rangle} n_i n_j \qquad \mathcal{V} = -t \sum_{\langle i,j \rangle} (a_i^\dagger a_j + a_j^\dagger a_i)$$



$1$

DPHYS
Department of Physics
Institute for Theoretical Physics

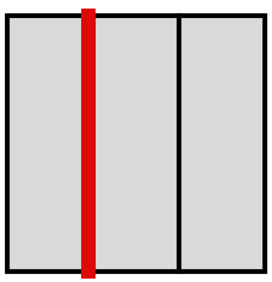
# Calculating configuration weights

- Continuous time algorithms just sample time-dependent perturbation expansion

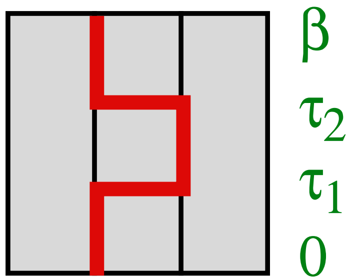
$$Z = \mathrm{Tr}\left(e^{-\beta H_0}\mathcal{T}e^{-\int_0^\beta d\tau \mathcal{V}(\tau)}\right)$$

- Examples: particles with nearest neighbor repulsion

$$H_0 = V\sum_{\langle i,j\rangle} n_i n_j \qquad \mathcal{V} = -t\sum_{\langle i,j\rangle}(a_i^\dagger a_j + a_j^\dagger a_i)$$



$1$




$t^2 d\tau_1 d\tau_2$

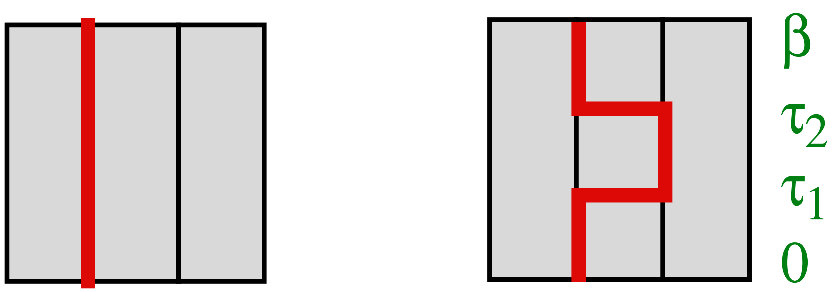
# Calculating configuration weights

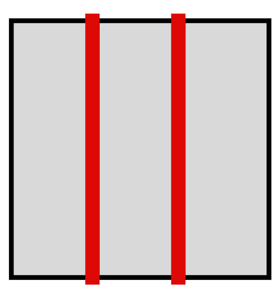
- Continuous time algorithms just sample time-dependent perturbation expansion

$$Z = \mathrm{Tr}\left(e^{-\beta H_0} \mathcal{T} e^{-\int_0^\beta d\tau \mathcal{V}(\tau)}\right)$$

- Examples: particles with nearest neighbor repulsion

$$H_0 = V \sum_{\langle i,j \rangle} n_i n_j \qquad \mathcal{V} = -t \sum_{\langle i,j \rangle} (a_i^\dagger a_j + a_j^\dagger a_i)$$



$$1 \qquad t^2 d\tau_1 d\tau_2 \qquad e^{-\beta V}$$

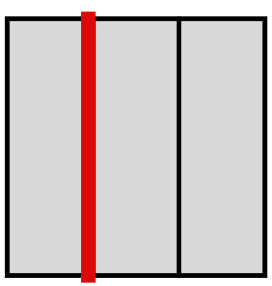
# Calculating configuration weights

- Continuous time algorithms just sample time-dependent perturbation expansion

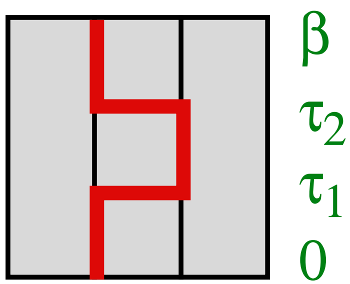
$$Z = \mathrm{Tr}\left(e^{-\beta H_0}\mathcal{T}e^{-\int_0^\beta d\tau \mathcal{V}(\tau)}\right)$$

- Examples: particles with nearest neighbor repulsion

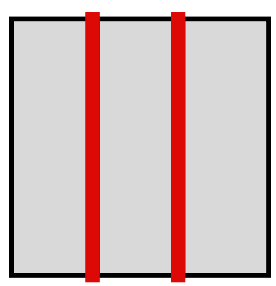
$$H_0 = V\sum_{\langle i,j\rangle} n_i n_j \qquad \mathcal{V} = -t\sum_{\langle i,j\rangle}(a_i^\dagger a_j + a_j^\dagger a_i)$$



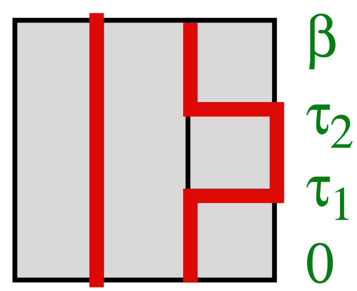
$$1$$




$$t^2 d\tau_1 d\tau_2$$



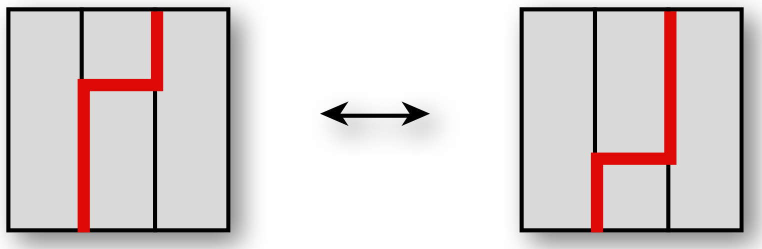
$$e^{-\beta V}$$



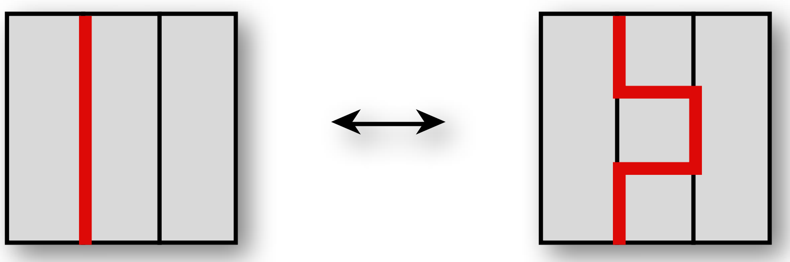

$$e^{-\tau_1 V}e^{-(\beta-\tau_2)V}t^2 d\tau_1 d\tau_2$$

# Updates in continuous time

- Shift a kink to any new position:

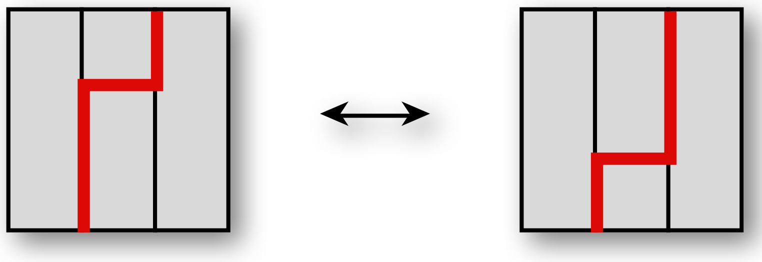


- Insert a pair of kinks:

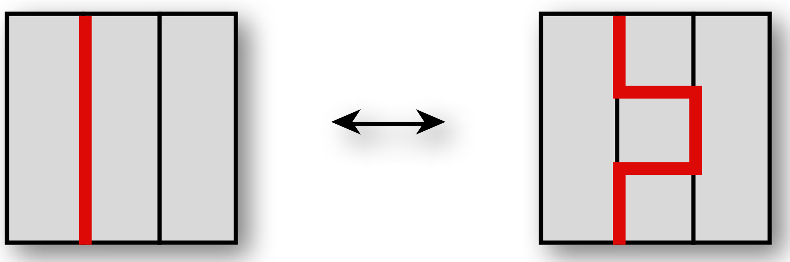


# Updates in continuous time

- Shift a kink to any new position:



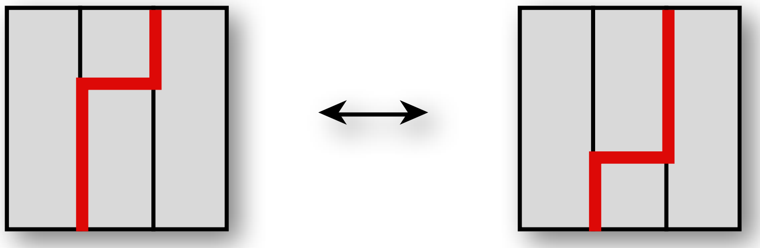
- Insert a pair of kinks:



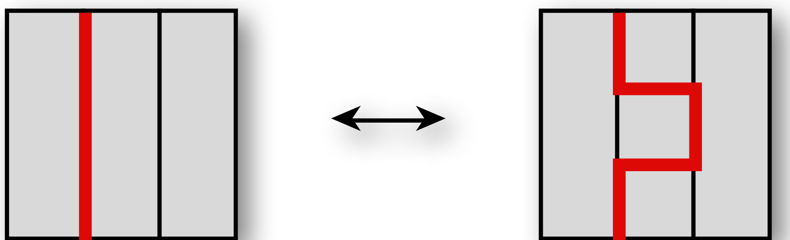
$$P = 1 \qquad P = (\Delta\tau t)^2 \to 0$$

# Updates in continuous time

- Shift a kink to any new position:



- Insert a pair of kinks:



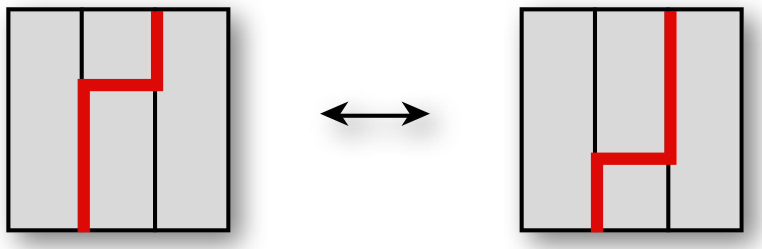
$$P = 1 \qquad\qquad P = (\Delta\tau t)^2 \to 0$$

*vanishing acceptance rate*

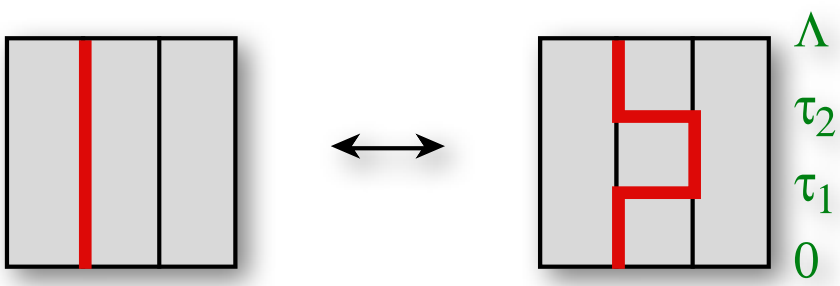
$$P_{\rightarrow} = \min\left[1, (\Delta\tau t)^2\right] \to 0$$

# Updates in continuous time

- Shift a kink to any new position:



- Insert a pair of kinks:




*solution:*

*integrate over all possible insertions in an interval*

$$P = 1$$

$$P = (\Delta\tau t)^2 \to 0$$

$$P = \int_0^{\Lambda}\int_{\tau_1}^{\Lambda} t^2 d\tau_2 d\tau_1 = \frac{\Lambda^2 t^2}{2} \neq 0$$

*vanishing acceptance rate*

$$P_{\to} = \min\left[1,(\Delta\tau t)^2\right] \to 0$$

$$P_{\to} = \min\left[1,\Lambda^2 t^2/2\right] \neq 0$$

# Advantages of continuous time

- No need to extrapolate in time step
  - a single simulation is sufficient
  - no additional errors from extrapolation
- Less memory and CPU time required
  - Instead of a time step $\Delta\tau << t$ we only have to store changes in the configuration happening at mean distances $\approx t$
  - Speedup of $1 / \Delta\tau \approx 10$
- Conceptual advantage
  - we directly sample a diagrammatic perturbation expansion

# 7. The loop algorithm

# Problems with local updates

# Problems with local updates

- Local updates cannot change global topological properties
  - number of world lines (particles, magnetization) conserved
  - winding conserved
  - braiding conserved
  - cannot sample grand-canonical ensemble

# Problems with local updates

- Local updates cannot change global topological properties
  - number of world lines (particles, magnetization) conserved
  - winding conserved
  - braiding conserved
  - cannot sample grand-canonical ensemble
- Critical slowing down at second order phase transitions
  - solved by cluster updates

# Problems with local updates

- Local updates cannot change global topological properties
  - number of world lines (particles, magnetization) conserved
  - winding conserved
  - braiding conserved
  - cannot sample grand-canonical ensemble
- Critical slowing down at second order phase transitions
  - solved by cluster updates
- Tunneling problem at first order phase transitions

# Cluster algorithms: the formal explanation

- Extend the phase space to configurations + graphs $(C,G)$

$$Z=\sum_C W(C)=\sum_C\sum_G W(C,G) \text{ with } W(C)=\sum_G W(C,G)$$

- Choose graph weights independent of configuration

$$W(C,G)=\Delta(C,G)V(G) \text{ where } \Delta(C,G)=\begin{cases}1 & \text{graph } G \text{ allowed for } C\\ 0 & \text{otherwise}\end{cases}$$

- Perform updates

$$C_i \rightarrow (C_i,G) \rightarrow G \rightarrow (C_{i+1},G) \rightarrow C_{i+1}$$

- Detailed balance is satisfied

# Cluster algorithms: the formal explanation

- Extend the phase space to configurations + graphs $(C,G)$

$$Z = \sum_C W(C) = \sum_C \sum_G W(C,G) \text{ with } W(C) = \sum_G W(C,G)$$

- Choose graph weights independent of configuration

$$W(C,G) = \Delta(C,G)V(G) \text{ where } \Delta(C,G) = \begin{cases} 1 & \text{graph } G \text{ allowed for } C \\ 0 & \text{otherwise} \end{cases}$$

- Perform updates

$$C_i \to (C_i,G) \to G \to (C_{i+1},G) \to C_{i+1}$$

1. Pick a graph $G$ $P[G] = \frac{V(G)}{W(C)}$

- Detailed balance is satisfied

# Cluster algorithms: the formal explanation

- Extend the phase space to configurations + graphs $(C,G)$

$$Z = \sum_C W(C) = \sum_C \sum_G W(C,G) \text{ with } W(C) = \sum_G W(C,G)$$

- Choose graph weights independent of configuration

$$W(C,G) = \Delta(C,G)V(G) \text{ where } \Delta(C,G) = \begin{cases} 1 & \text{graph } G \text{ allowed for } C \\ 0 & \text{otherwise} \end{cases}$$

- Perform updates

2. Discard configuration

$$C_i \rightarrow (C_i,G) \rightarrow G \rightarrow (C_{i+1},G) \rightarrow C_{i+1}$$

1. Pick a graph $G$ $P[G] = \dfrac{V(G)}{W(C)}$

- Detailed balance is satisfied

# Cluster algorithms: the formal explanation

- Extend the phase space to configurations + graphs $(C,G)$

$$Z = \sum_C W(C) = \sum_C \sum_G W(C,G) \text{ with } W(C) = \sum_G W(C,G)$$

- Choose graph weights independent of configuration

$$W(C,G) = \Delta(C,G)V(G) \text{ where } \Delta(C,G) = \begin{cases} 1 & \text{graph } G \text{ allowed for } C \\ 0 & \text{otherwise} \end{cases}$$

- Perform updates

2. Discard configuration

$$C_i \rightarrow (C_i, G) \rightarrow G \rightarrow (C_{i+1}, G) \rightarrow C_{i+1}$$

1. Pick a graph $G$ $P[G] = \frac{V(G)}{W(C)}$

3. Pick any allowed new configuration

- Detailed balance is satisfied

# Cluster algorithms: the formal explanation

- Extend the phase space to configurations + graphs $(C,G)$

$$Z = \sum_C W(C) = \sum_C \sum_G W(C,G) \text{ with } W(C) = \sum_G W(C,G)$$

- Choose graph weights independent of configuration

$$W(C,G) = \Delta(C,G)V(G) \text{ where } \Delta(C,G) = \begin{cases} 1 & \text{graph } G \text{ allowed for } C \\ 0 & \text{otherwise} \end{cases}$$

- Perform updates

$$C_i \rightarrow (C_i,G) \rightarrow G \rightarrow (C_{i+1},G) \rightarrow C_{i+1}$$

1. Pick a graph $G$ $P[G] = \frac{V(G)}{W(C)}$
2. Discard configuration
3. Pick any allowed new configuration
4. Discard graph

- Detailed balance is satisfied

# Cluster algorithms: the formal explanation

- Extend the phase space to configurations + graphs $(C,G)$

$$Z=\sum_C W(C)=\sum_C\sum_G W(C,G) \text{ with } W(C)=\sum_G W(C,G)$$

- Choose graph weights independent of configuration

$$W(C,G)=\Delta(C,G)V(G) \text{ where } \Delta(C,G)=\begin{cases}1 & \text{graph } G \text{ allowed for } C\\ 0 & \text{otherwise}\end{cases}$$

- Perform updates

$$C_i \rightarrow (C_i,G) \rightarrow G \rightarrow (C_{i+1},G) \rightarrow C_{i+1}$$

1. Pick a graph $G$ $P[G]=\frac{V(G)}{W(C)}$
2. Discard configuration
3. Pick any allowed new configuration
4. Discard graph

- Detailed balance is satisfied

$$\frac{P[(C_i,G)\rightarrow(C_{i+1},G)]}{P[(C_{i+1},G)\rightarrow(C_i,G)]}=\frac{1/\mathrm{N_C}}{1/\mathrm{N_C}}=1=\frac{\Delta(C_{i+1},G)V(G)}{\Delta(C_i,G)V(G)}=\frac{P[(C_{i+1},G)]}{P[(C_i,G)]}$$

# Cluster algorithms: Ising model

- We need to find $\Delta(C,G)$ and $V(G)$ to fulfill $W(C)=\sum_G W(C,G)=\sum_G \Delta(C,G)V(G)$

| $\Delta(C,G)$ | o-o | o o | $W(C)$ |
|---|---|---|---|
| ↑↑, ↓↓ | 1 | 1 | $e^{+\beta J}$ |
| ↑↓, ↓↑ | 0 | 1 | $e^{-\beta J}$ |
| $W(G)$ | $e^{+\beta J}-e^{-\beta J}$ | $e^{-\beta J}$ | |

- This means for: $C_i \rightarrow (C_i,G) \rightarrow G$
  - Parallel spins: pick connected graph o-o with $P($ o-o $)=\frac{e^{+\beta J}+e^{-\beta J}}{e^{+\beta J}}=1-e^{-2\beta J}$
  - Antiparallel spins: always pick open graph o o
- And for: $G \rightarrow (C_{i+1},G) \rightarrow C_{i+1}$
  - Configuration must be allowed ⇒ connected spins must be parallel ⇒ connected spins flipped as one cluster

PHYSICAL REVIEW LETTERS

VOLUME 70 | 15 FEBRUARY 1993 | NUMBER 7

# Cluster Algorithm for Vertex Models

Hans Gerd Evertz,[(1),(a)] Gideon Lana,[(2),(b)] and Mihai Marcu[(2),(c)]
[(1)] *Supercomputer Computations Research Institute, Florida State University, Tallahassee, Florida 32306*
[(2)] *School of Physics and Astronomy, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, 69978 Tel Aviv, Israel*

(Received 17 November 1992)

We present a new type of cluster algorithm that strongly reduces critical slowing down in simulations of vertex models. Since the clusters are closed paths of bonds, we call it the *loop algorithm*. The basic steps in constructing a cluster are the breakup and the freezing of vertices. We concentrate on the case of the F model, which is a subset of the six-vertex model exhibiting a Kosterlitz-Thouless transition. The loop algorithm is also applicable to simulations of other vertex models and of one- and two-dimensional quantum spin systems.

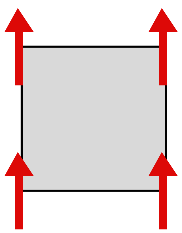
PACS numbers: 02.70.–c, 05.50.+q, 68.35.Rh, 75.10.Jm

# The loop algorithm (Evertz *et al*)

- Swendsen-Wang cluster algorithm for the Ising model
  - two choices on each bond: *connected (flip both spins)* or *disconnected*

  - all connected spins are flipped together

- Loop algorithm is a generalization to quantum systems
  - world lines may not be broken
  - always 2 or 4 spins must be flipped together



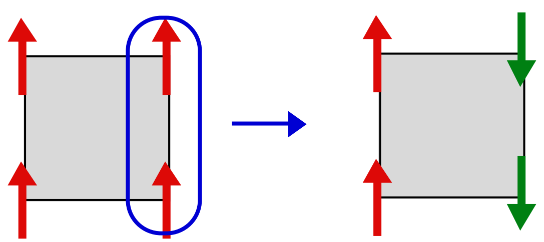
  - four different connection types

# The loop algorithm (Evertz *et al*)

- Swendsen-Wang cluster algorithm for the Ising model
  - two choices on each bond: *connected (flip both spins)* or *disconnected*

  - all connected spins are flipped together

- Loop algorithm is a generalization to quantum systems
  - world lines may not be broken
  - always 2 or 4 spins must be flipped together



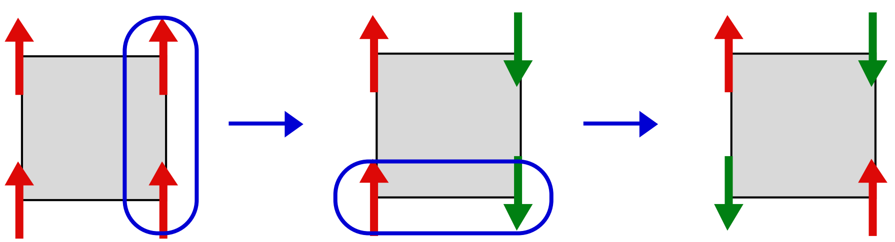
  - four different connection types

# The loop algorithm (Evertz *et al*)

- Swendsen-Wang cluster algorithm for the Ising model
  - two choices on each bond: *connected (flip both spins)* or *disconnected*

  - all connected spins are flipped together

- Loop algorithm is a generalization to quantum systems
  - world lines may not be broken
  - always 2 or 4 spins must be flipped together



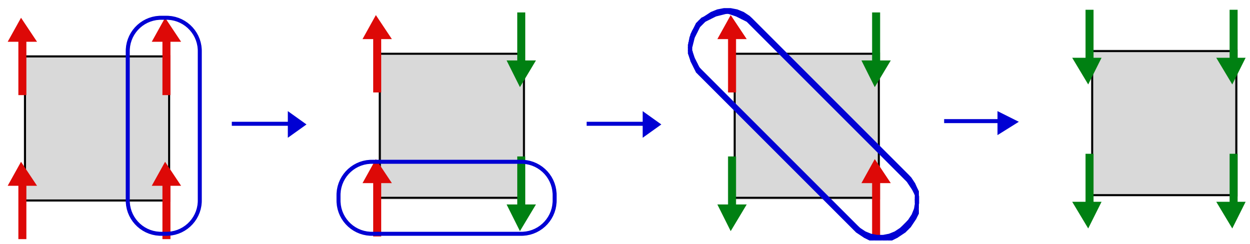
  - four different connection types

# The loop algorithm (Evertz *et al*)

- Swendsen-Wang cluster algorithm for the Ising model
  - two choices on each bond: *connected (flip both spins)* or *disconnected*

  - all connected spins are flipped together

- Loop algorithm is a generalization to quantum systems
  - world lines may not be broken
  - always 2 or 4 spins must be flipped together

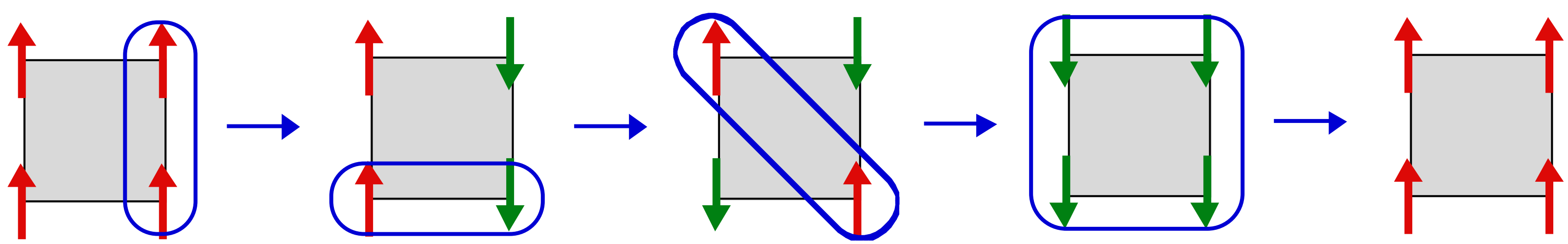


  - four different connection types

# The loop algorithm (Evertz *et al*)

- Swendsen-Wang cluster algorithm for the Ising model
  - two choices on each bond: *connected (flip both spins)* or *disconnected*

  - all connected spins are flipped together
- Loop algorithm is a generalization to quantum systems
  - world lines may not be broken
  - always 2 or 4 spins must be flipped together



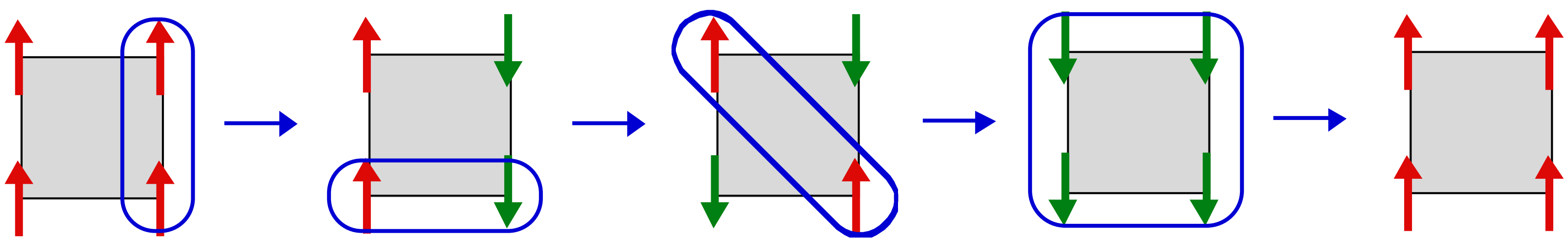
  - four different connection types

# The loop algorithm (Evertz *et al*)

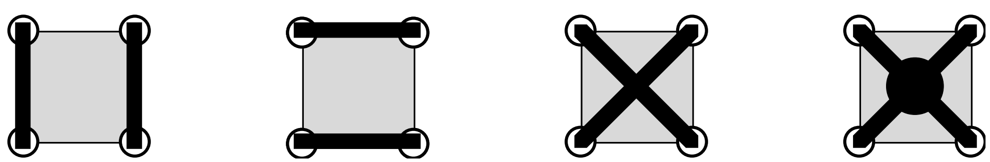
- Swendsen-Wang cluster algorithm for the Ising model
  - two choices on each bond: *connected (flip both spins)* or *disconnected*

  - all connected spins are flipped together

- Loop algorithm is a generalization to quantum systems
  - world lines may not be broken
  - always 2 or 4 spins must be flipped together



  - four different connection types



# Hamiltonian of spin-1/2 models

- Consider a 2-site quantum spin-1/2 model

$$H_{XXZ} = J_{xz}(S_1^x S_2^x + S_1^y S_2^y) + J_z S_1^z S_2^z - h\left(S_1^z + S_2^z\right)$$
$$= \frac{J_{xz}}{2}(S_1^+ S_2^- + S_1^- S_2^+) + J_z S_1^z S_2^z - h\left(S_1^z + S_2^z\right)$$

  - Heisenberg model if $J_{xy} = J_z = J$

$$H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$$

- Hamiltonian matrix in 2-site basis

$$\{\ |\uparrow\uparrow\rangle,\ |\uparrow\downarrow\rangle,\ |\downarrow\uparrow\rangle,\ |\downarrow\downarrow\rangle\ \}$$

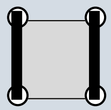
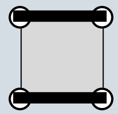
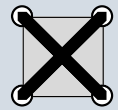
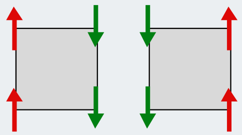
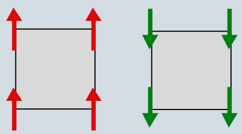
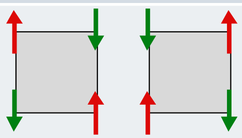
$$H_{XXZ} = \begin{pmatrix} \frac{J_z}{4}+h & 0 & 0 & 0 \\ 0 & -\frac{J_z}{4} & \frac{J_{xy}}{2} & 0 \\ 0 & \frac{J_{xy}}{2} & -\frac{J_z}{4} & 0 \\ 0 & 0 & 0 & \frac{J_z}{4}-h \end{pmatrix}$$

# Cluster building rules: XY-like antiferromagnet

$$H_{XXZ} = \frac{J_{xz}}{2} \sum_{\langle i,j \rangle} (S_i^+ S_j^- + S_i^- S_j^+) + J_z \sum_{\langle i,j \rangle} S_i^z S_j^z$$

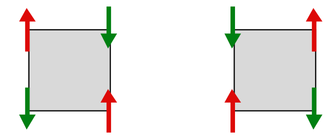
with $0 \le J_z \le J_{xy}$

$$W(C) = \sum_G W(C,G) = \sum_G \Delta(C,G) V(G)$$

| $\Delta(C,G)$ |  |  |  | $W(C)$ |
|---|---|---|---|---|
|  | 1 | 1 | – | $1+ (J_z/4)\, d\tau$ |
|  | 1 | – | 0 | $1-(J_z/4)\, d\tau$ |
|  | – | 1 | 1 | $(J_{xy}/2)\, d\tau$ |
| $V(G)$ | $1-(J_z/4)\, d\tau$ | $(J_z/2)\, d\tau$ | $(J_{xy}- J_z)/2\, d\tau$ | |

# How to deal with infinitesimals ?

- How do we deal with the vanishing $d\tau$ terms in continuous time?
- First example: the exchange process



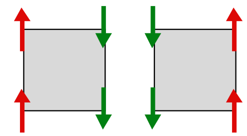
  - Possible graph connections:

  - Graph weights: $\frac{J_z}{2} d\tau$ $\qquad$ $\frac{J_{xy} - J_z}{2} d\tau$
  - Probability to pick graph: (divide weight by sum) $\frac{J_z}{J_{xy}}$ $\qquad$ $\frac{J_{xy} - J_z}{J_{xy}}$

- The infinitesimal $d\tau$ terms cancel out
  - Randomly pick one of the graphs (with appropriate probabilities)

# How to deal with infinitesimals ?

- How do we deal with the vanishing $d\tau$ terms in continuous time?
- Second example: the "decay" process



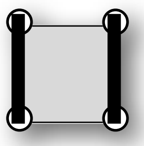
  - Possible graph connections:

  - Graph weights: $1-\frac{J_z}{4}d\tau$ $\quad$ $\frac{J_z}{2}d\tau$
  - Probability to pick graph: (divide weight by sum) $1-\frac{J_z}{2}d\tau$ $\quad$ $\frac{J_z}{2}d\tau$

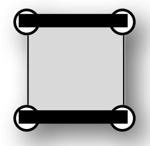
- The infinitesimal $d\tau$ terms remain
- Infinitesimal acceptance rate at infinitely many time steps?

# How to deal with infinitesimals ?

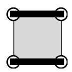
- We have to tackle the problem of vanishing probabilities
  - Example: Heisenberg antiferromagnet



$$P_{||} = 1 - \frac{J}{2}\Delta\tau \to 1$$

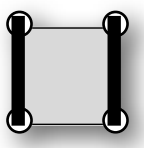


$$P_{=} = \frac{J}{2}\Delta\tau \to 0$$

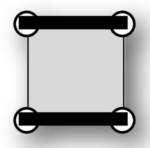
- Interpret the  connection as a “decay process” where the loop jumps

# How to deal with infinitesimals ?

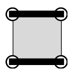
- We have to tackle the problem of vanishing probabilities
  - Example: Heisenberg antiferromagnet



$$P_{\parallel} = 1 - \frac{J}{2}\Delta\tau \to 1$$



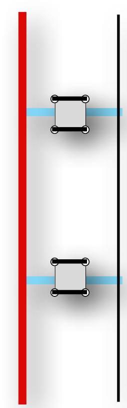
$$P_{=} = \frac{J}{2}\Delta\tau \to 0$$

- Interpret the  connection as a "decay process" where the loop jumps

$$P_{=} = \frac{J}{2}d\tau$$

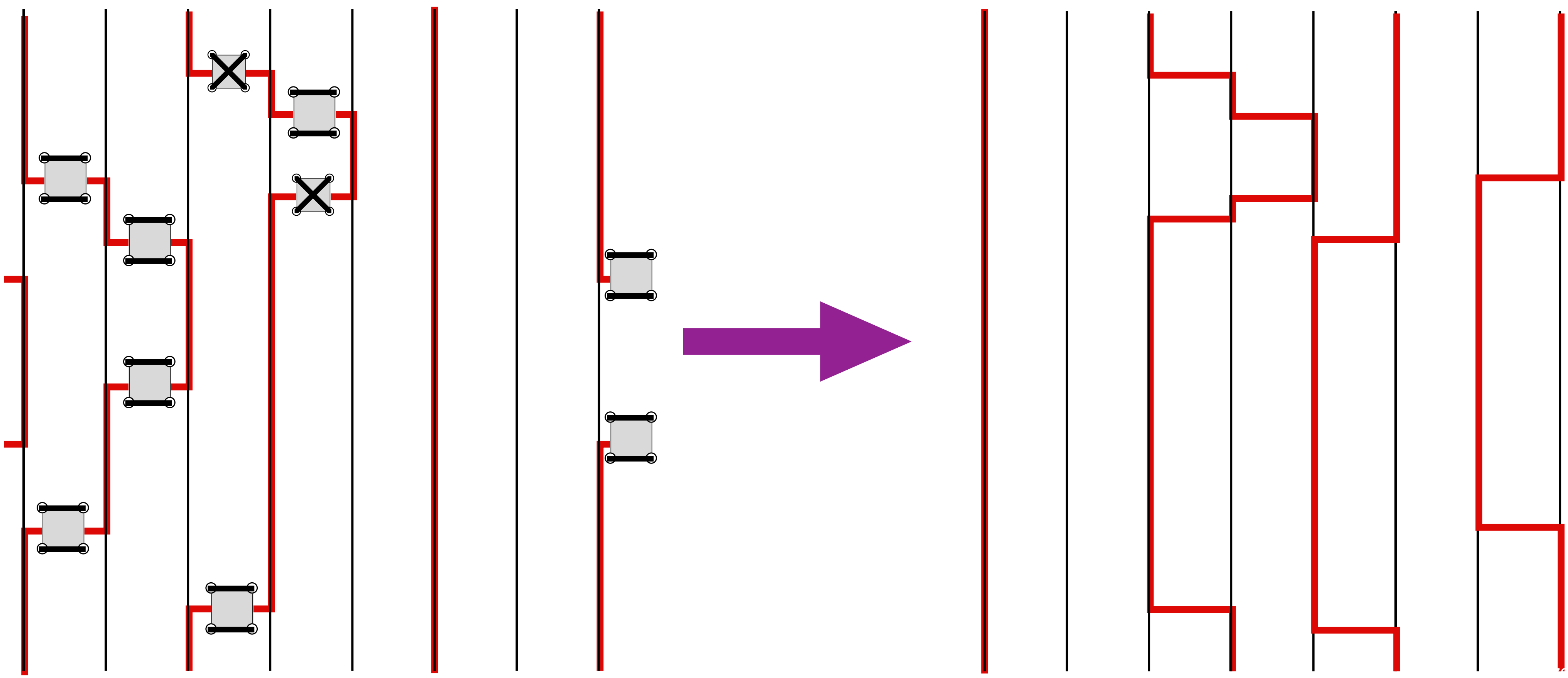
$$\text{decay constant } \lambda = \frac{J}{2}$$

$$\text{mean distance } \; d = \frac{2}{J}$$



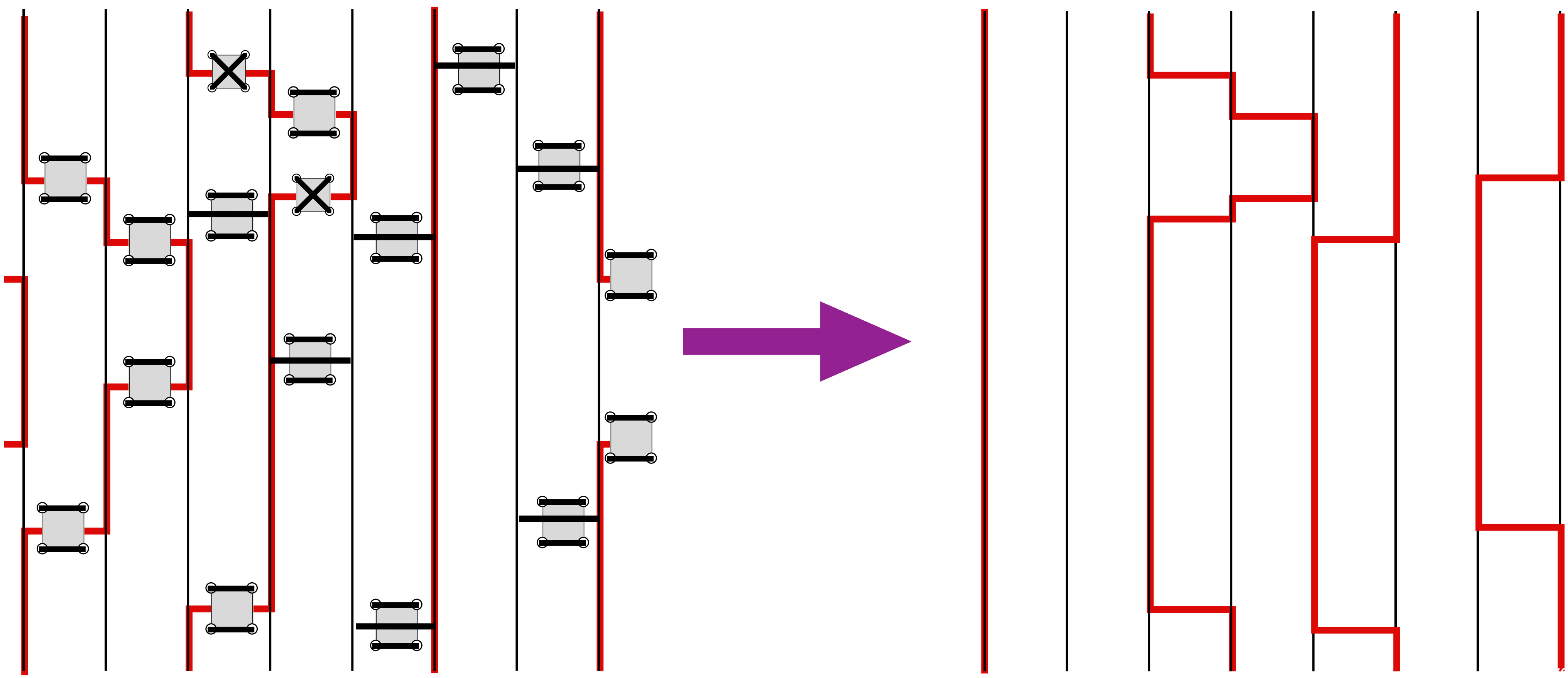
# Loop-cluster updates

1. Connect spins according to loop-custer building rules



# Loop-cluster updates

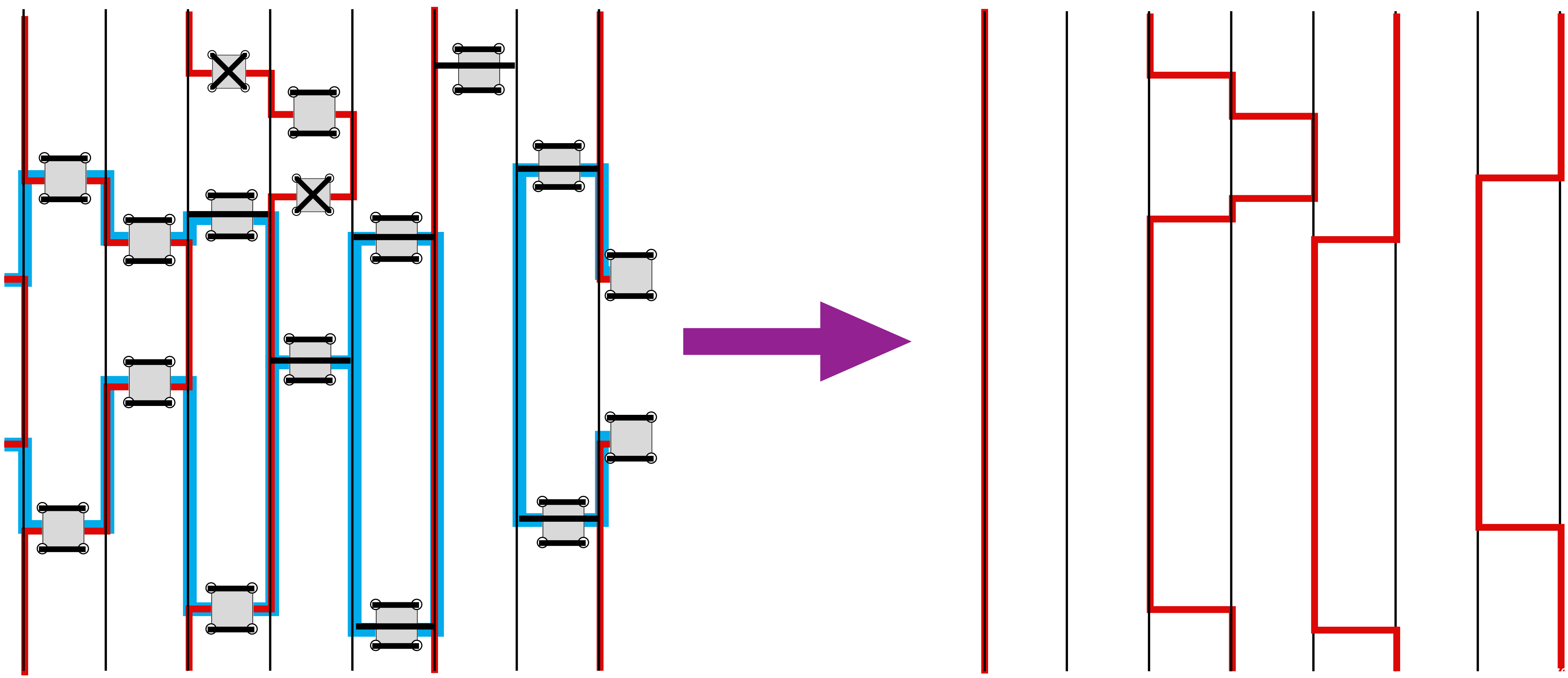
1. Connect spins according to loop-custer building rules



# Loop-cluster updates

1. Connect spins according to loop-custer building rules

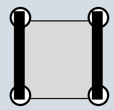
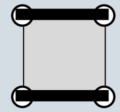
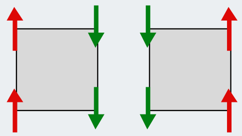
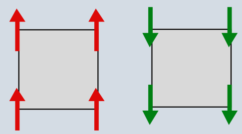
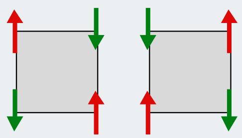
2. Build and flip loop-cluster



# Heisenberg antiferromagnet

$$H_{\text{Heisenberg}} = J \sum_{\langle i,j \rangle} \vec{S}_i \vec{S}_j$$

$$W(C) = \sum_G W(C,G) = \sum_G \Delta(C,G) V(G)$$

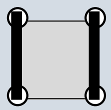
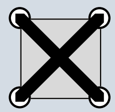
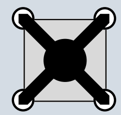
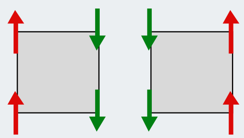
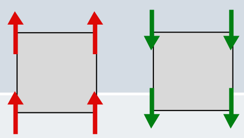
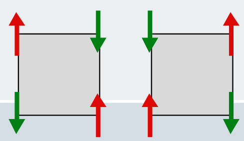
| $\Delta(C,G)$ |  |  | $W(C)$ |
|---|---|---|---|
|  | 1 | 1 | $1+ (J/4)\, d\tau$ |
|  | 1 | 0 | $1-(J/4)\, d\tau$ |
|  | 0 | 1 | $(J/2)\, d\tau$ |
| $V(G)$ | $1-(J/4)\, d\tau$ | $(J/2)\, d\tau$ | |

- Connected spins form a cluster and have to be flipped together
- Very simple and deterministic for Heisenberg model

# Ising-like ferromagnet

$$H_{XXZ} = -\frac{J_{xz}}{2} \sum_{\langle i,j \rangle} (S_i^+ S_j^- + S_i^- S_j^+) - J_z \sum_{\langle i,j \rangle} S_i^z S_j^z \qquad W(C) = \sum_G W(C,G) = \sum_G \Delta(C,G) V(G)$$

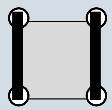
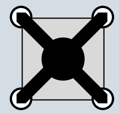
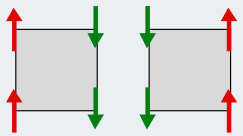
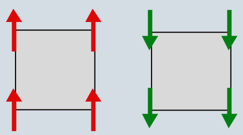
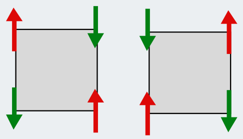
with $0 \le J_{xy} \le J_z$

| $\Delta(C,G)$ |  |  |  | $W(C)$ |
|---|---|---|---|---|
|  | 1 | 0 | 0 | $1-(J_z/4)\,d\tau$ |
|  | 1 | 1 | 1 | $1+(J_z/4)\,d\tau$ |
|  | 0 | 1 | 0 | $(J_{xy}/2)\,d\tau$ |
| $V(G)$ | $1-(J_z/4)\,d\tau$ | $(J_{xy}/2)\,d\tau$ | $(J_z-J_{xy})/2\,d\tau$ | |

- Now 4-spin freezing graph is needed: connects (freezes) loops

# Ising ferromagnet

$$H_{\text{Ising}} = -J \sum_{\langle i,j \rangle} S_i^z S_j^z = -\frac{J}{4} \sum_{\langle i,j \rangle} \sigma_i \sigma_j \qquad W(C) = \sum_G W(C,G) = \sum_G \Delta(C,G) V(G)$$

| $\Delta(C,G)$ |  |  | $W(C)$ |
|---|---|---|---|
|  | 1 | 0 | $1 - (J/4)\, d\tau$ |
|  | 1 | 1 | $1 + (J/4)\, d\tau$ |
|  | 0 | 0 | 0 |
| $V(G)$ | $1-(J/4)\, d\tau$ | $(J/2)\, d\tau$ | |

- Two spins are frozen if there is any freezing graph along the world line

$$P_{\text{no freezing}} = \lim_{M \to \infty} (1 - (\beta/M) J/2)^M = \exp(-\beta J/2) = \exp(-2\beta J_{classical})$$

- We recover the Swendsen Wang algorithm: probability for no freezing

# 8. The worm algorithm

DPHYS
Department of Physics
Institute for Theoretical Physics

# Loop algorithm in a magnetic field

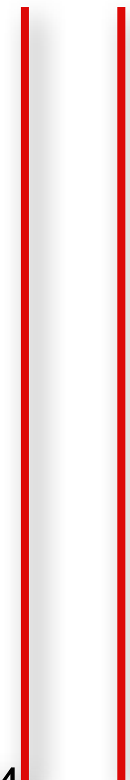
- Loop cluster algorithm requires spin inversion symmetry
  - Magnetic field implemented by a-posteriori acceptance rate
- Example: spin dimer at $J = h = 1$

$$H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$$

# Loop algorithm in a magnetic field

- Loop cluster algorithm requires spin inversion symmetry
  - Magnetic field implemented by a-posteriori acceptance rate
- Example: spin dimer at $J = h = 1$

$$H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$$

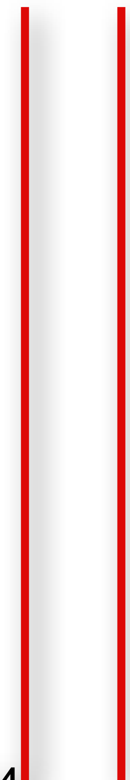


Triplet

E = J/4 - h = -3/4

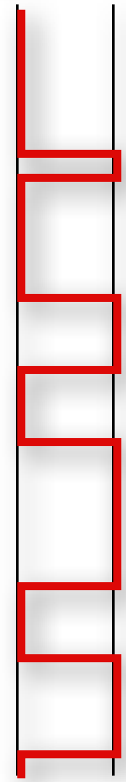
# Loop algorithm in a magnetic field

- Loop cluster algorithm requires spin inversion symmetry
  - Magnetic field implemented by a-posteriori acceptance rate
- Example: spin dimer at $J = h = 1$ $\quad H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$



Triplet

E = J/4 - h = -3/4



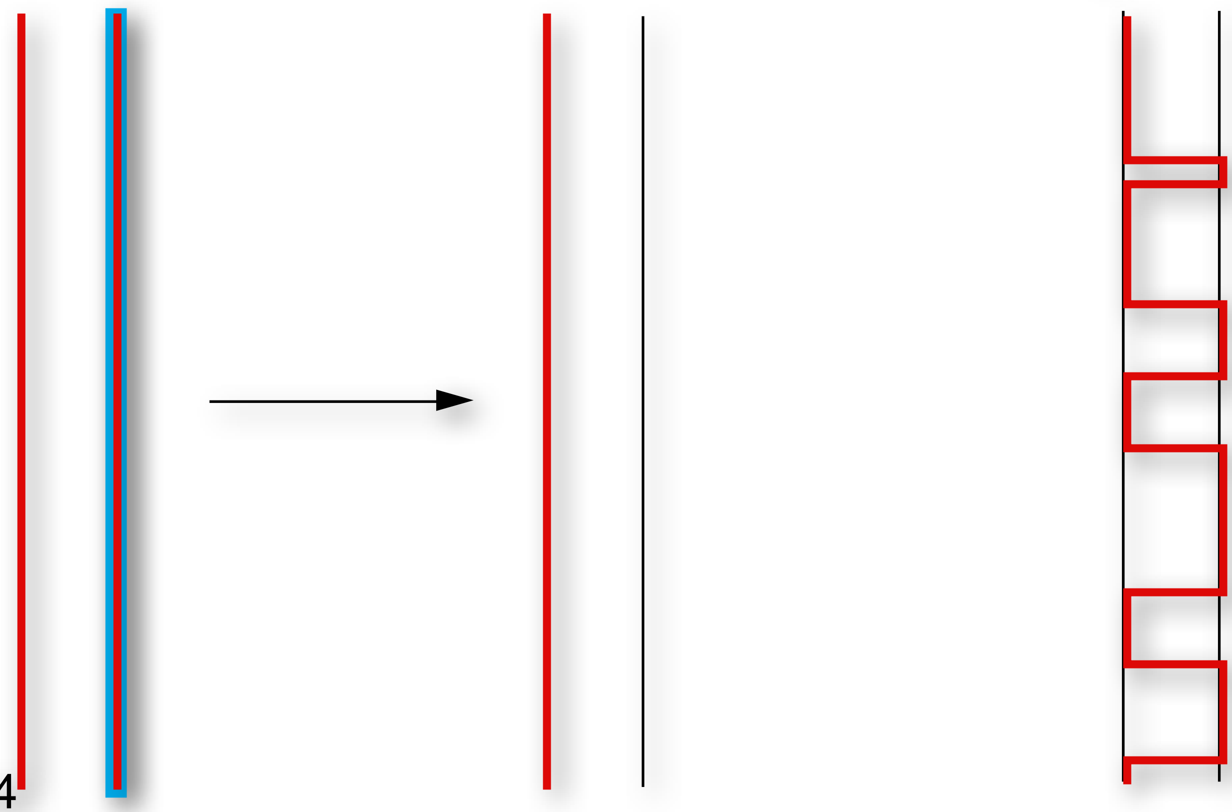
Singlet

E = -3J/4 = -3/4

# Loop algorithm in a magnetic field

- Loop cluster algorithm requires spin inversion symmetry
  - Magnetic field implemented by a-posteriori acceptance rate
- Example: spin dimer at $J = h = 1$

$$H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$$



Triplet

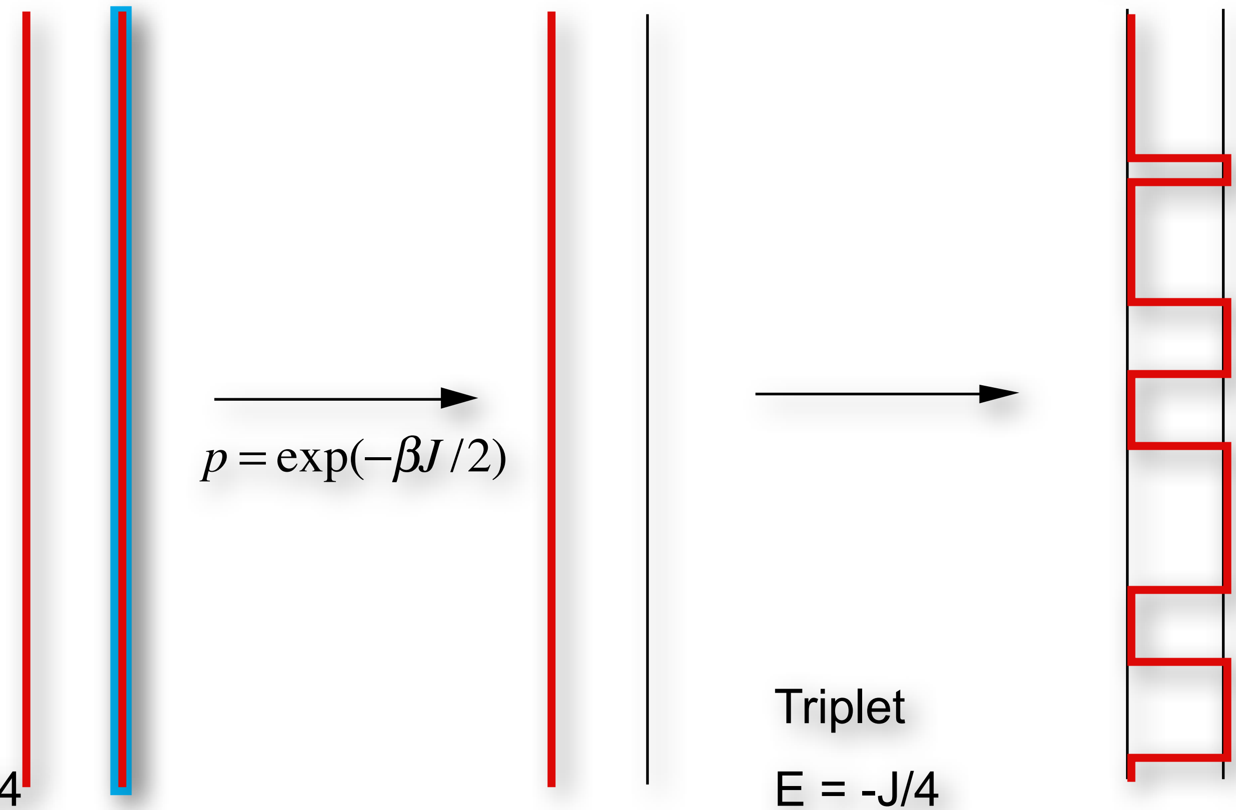
E = J/4 - h = -3/4

Singlet

E = -3J/4 = -3/4

# Loop algorithm in a magnetic field

- Loop cluster algorithm requires spin inversion symmetry
  - Magnetic field implemented by a-posteriori acceptance rate
- Example: spin dimer at $J = h = 1$ $\quad H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$




Triplet

E = J/4 - h = -3/4

Triplet

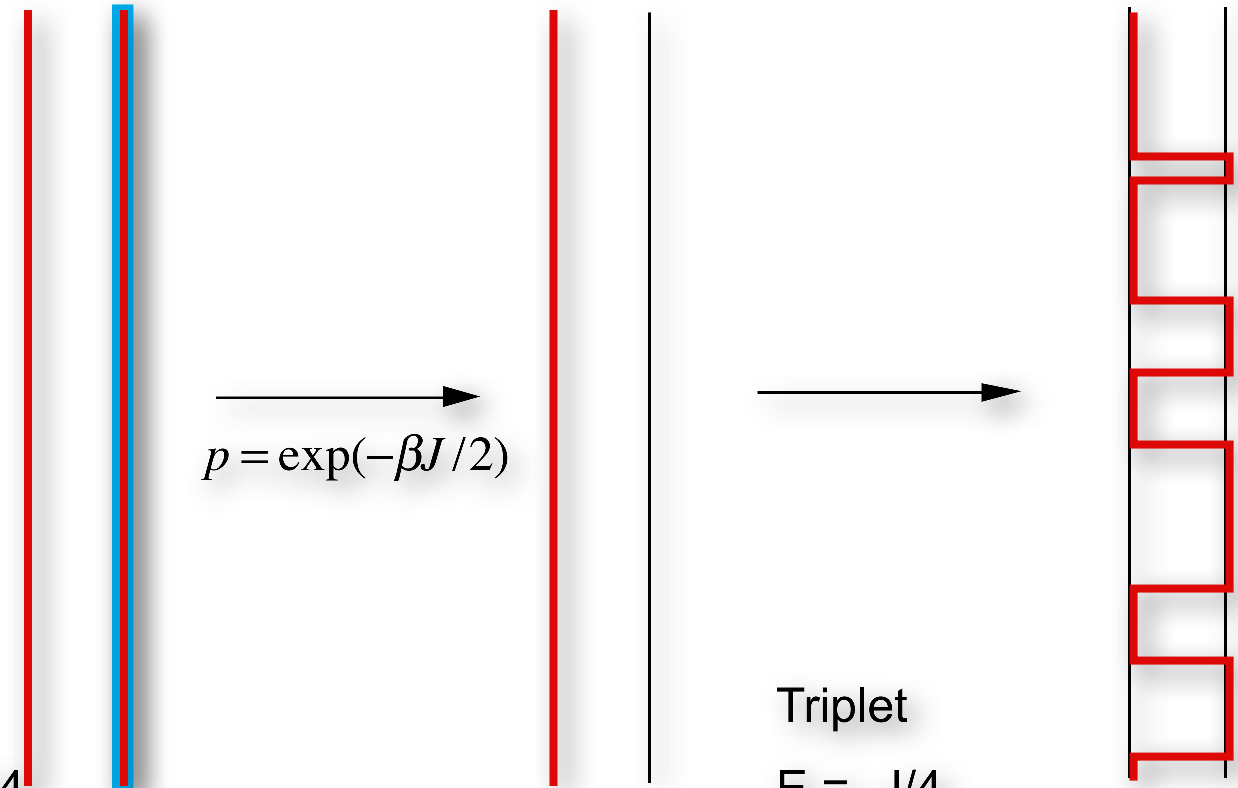
E = -J/4

Singlet

E = -3J/4 = -3/4

# Loop algorithm in a magnetic field

- Loop cluster algorithm requires spin inversion symmetry
  - Magnetic field implemented by a-posteriori acceptance rate
- Example: spin dimer at $J = h = 1$ $H = J\vec{S}_1\vec{S}_2 - h\left(S_1^z + S_2^z\right)$



$p = \exp(-\beta J/2)$

Triplet

E = J/4 - h = -3/4

Triplet

E = -J/4

Singlet

E = -3J/4 = -3/4

Exponential slowdown due to high energy intermediate state

# High-*T* expansion of the Ising model

$$Z = \sum_{s_1 \ldots s_N} \prod_{\langle ij \rangle} e^{K s_i s_j} = \sum_{s_1 \ldots s_N} \prod_{\langle ij \rangle} \Big[ \cosh(K) \Big( 1 + \tanh(K) s_i s_j \Big) \Big]$$

i.e.

$$Z = \cosh(K)^{2N} \sum_{s_1 \ldots s_N} \prod_{bonds} \sum_{n_b = 0}^{1} \Big[ \tanh(K)]^{n_b} s_i^{n_b} s_j^{n_b} \Big] \propto \sum_{\{n_b\}} \tanh(K)^{\sum n_b} \sum_{s_1 \ldots s_N} \prod_{bonds} s_i^{n_b} s_j^{n_b}$$

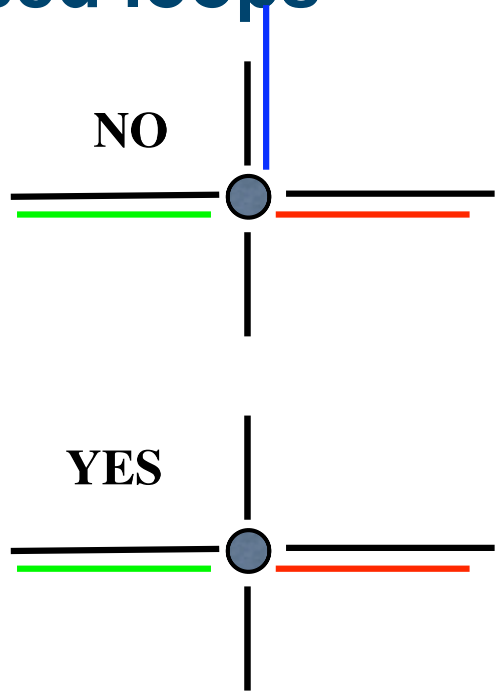
$n_b = 0, 1$: **power** associated to *bond* $\langle ij \rangle$

$$\sum_{s_1 \ldots s_N} \prod_{\langle ij \rangle} s_i^n s_j^n \equiv \prod_i \sum_{s_i} s_i^{p_i}, \quad p_i \quad \text{total power associated to site i}$$

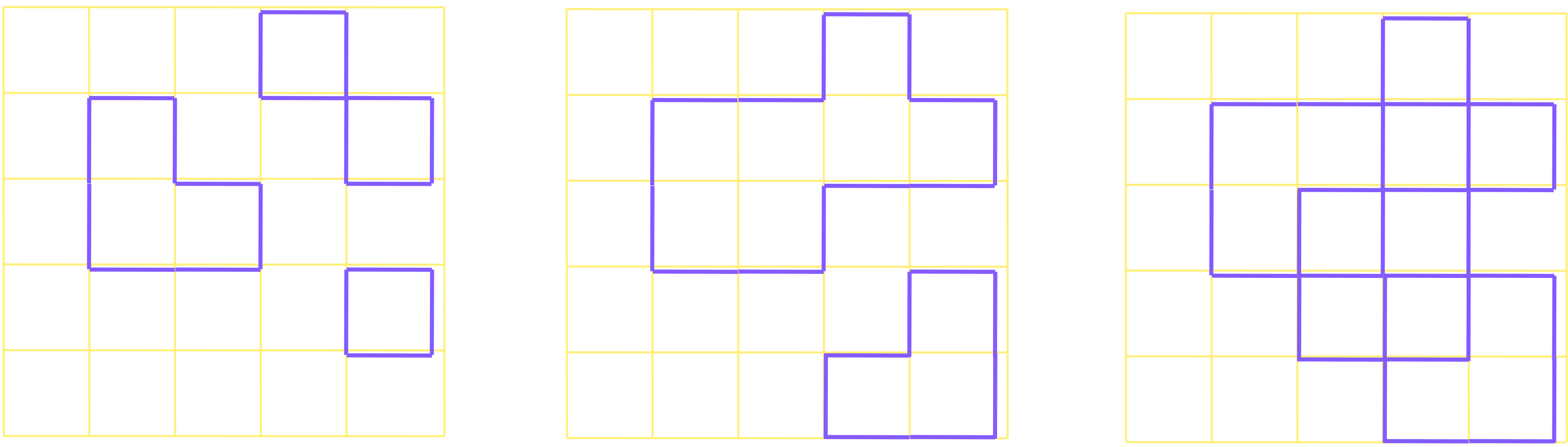
For a spin-1/2 system one has $\sum_s s^p = 2$ if $p$ is even, zero otherwise

$$\text{Hence,} \quad Z = 2^N \sum_{\{n_b\}} \Big[ \tanh(K) \Big]^{\sum n_b} \qquad (\textit{closed loops})$$

# Closed loops

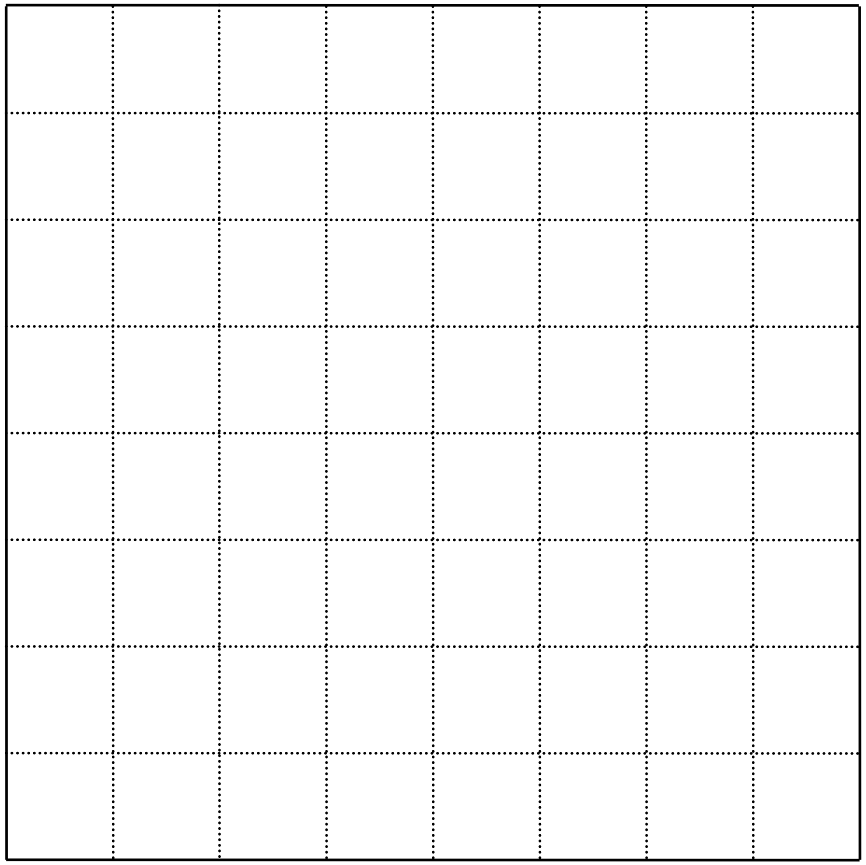



- Nonzero weight only if total even number of bond powers, thus all labeled bonds form closed loops
- Open-ended loops require one additional spin operator for each end: give correlation function measurements



# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

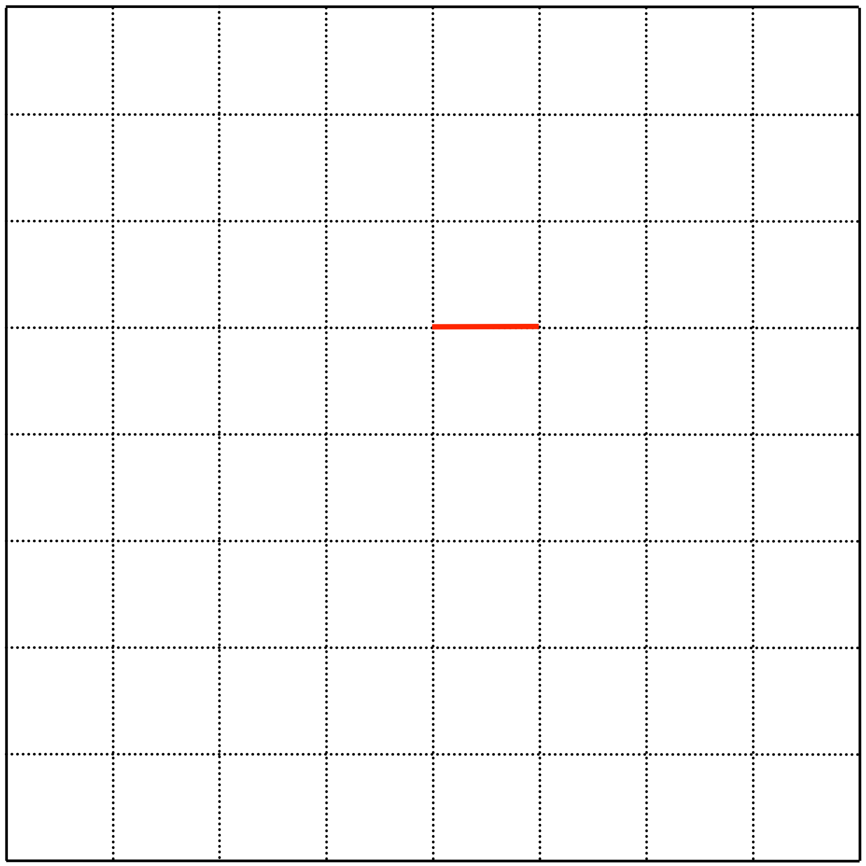
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

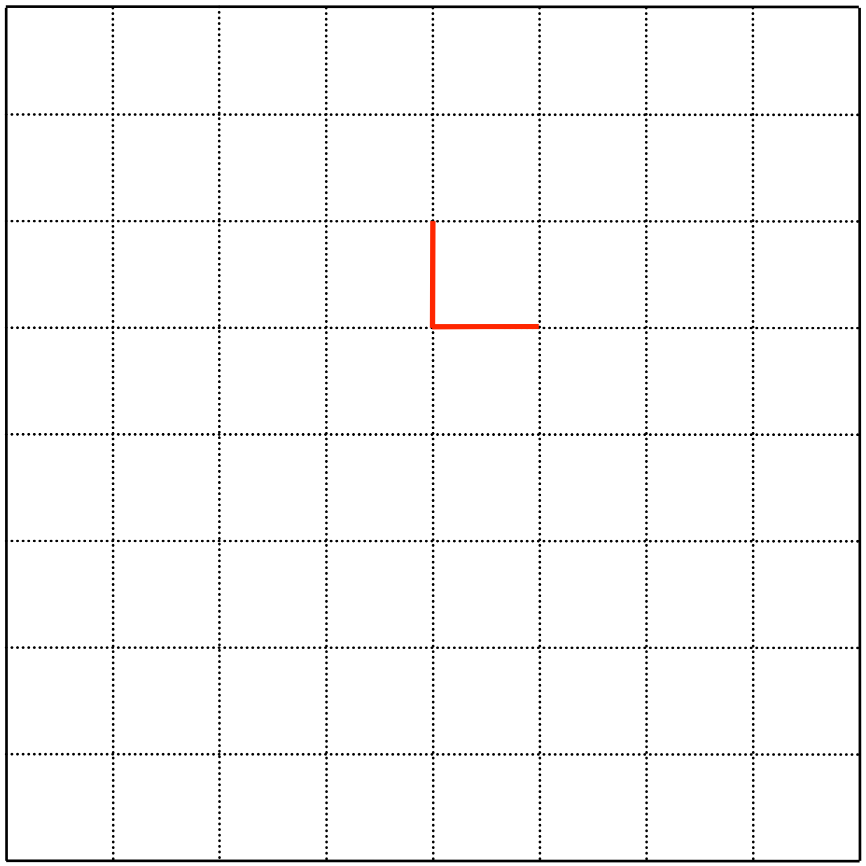
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

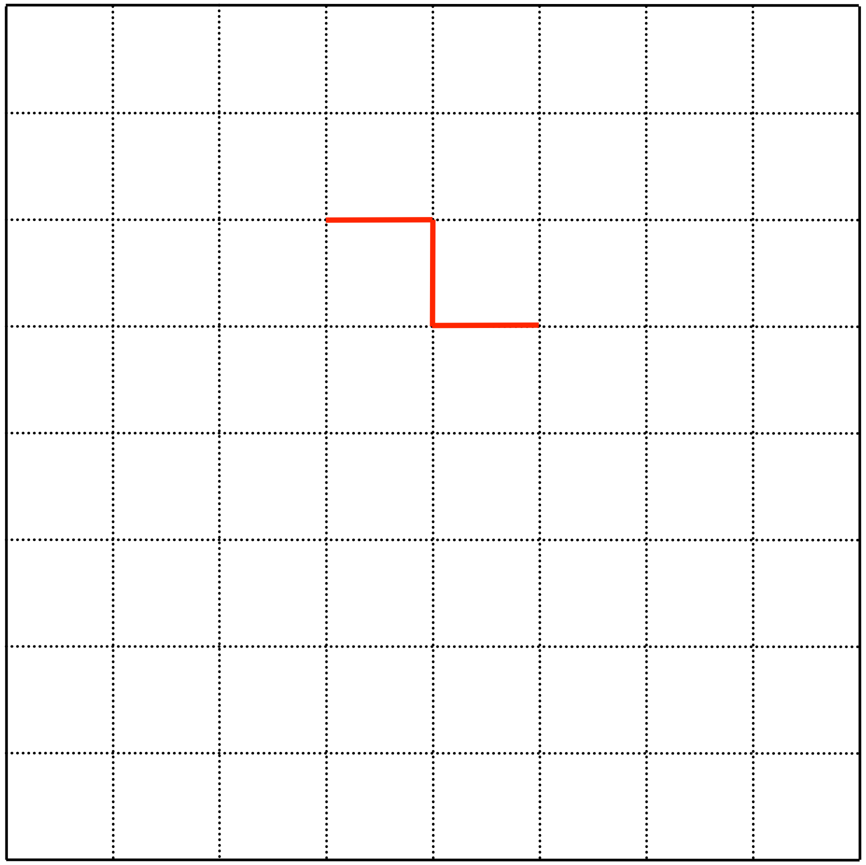
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

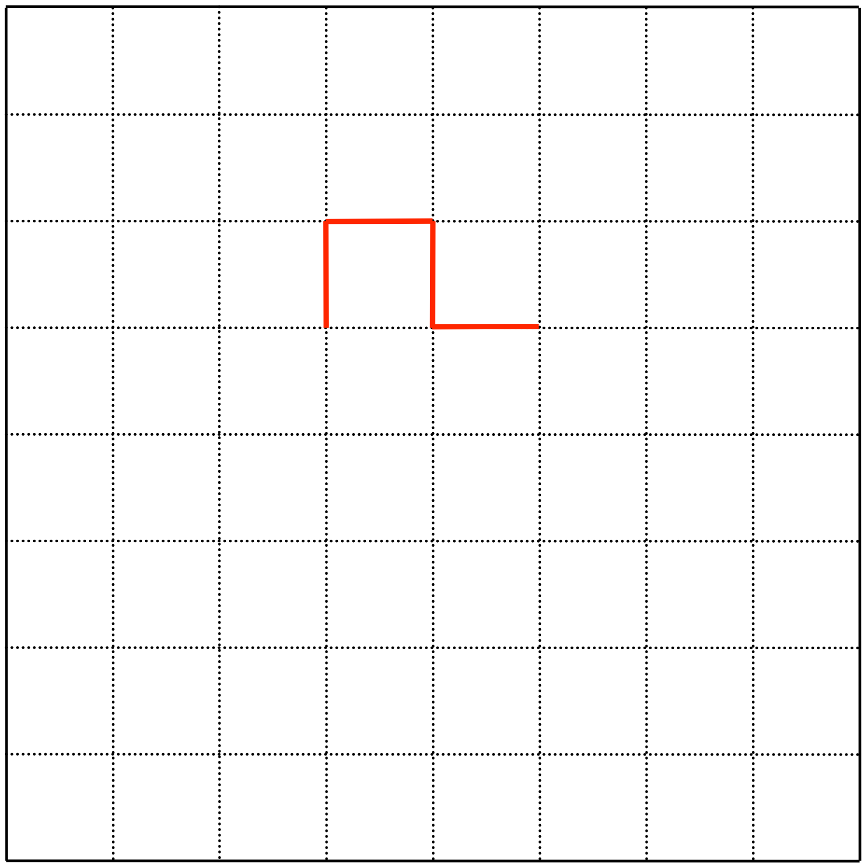
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

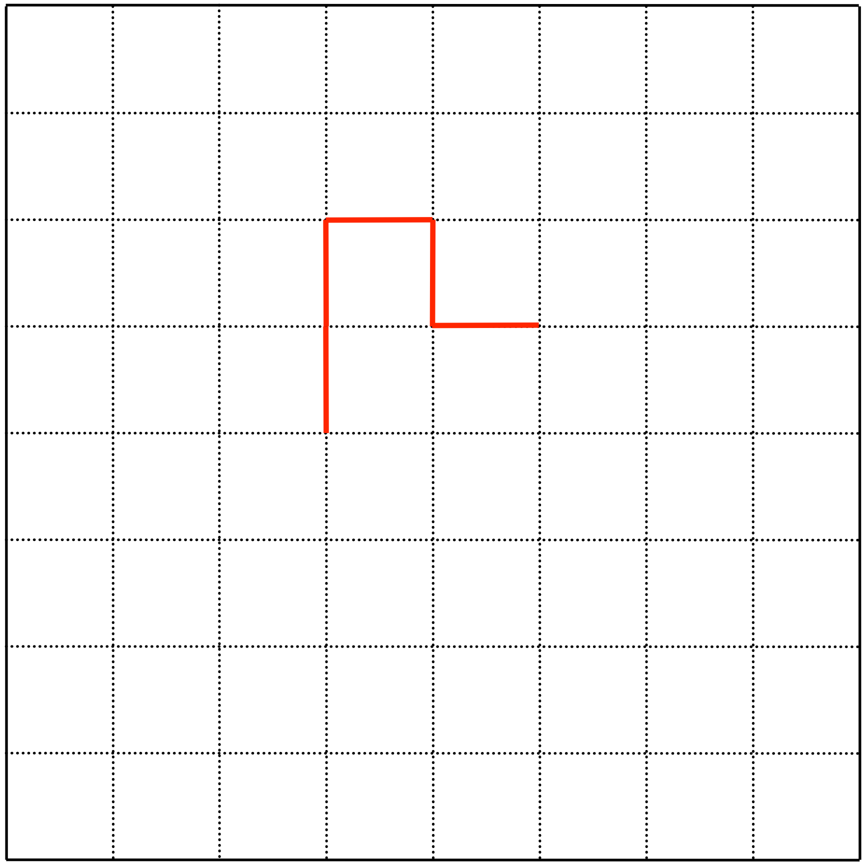
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

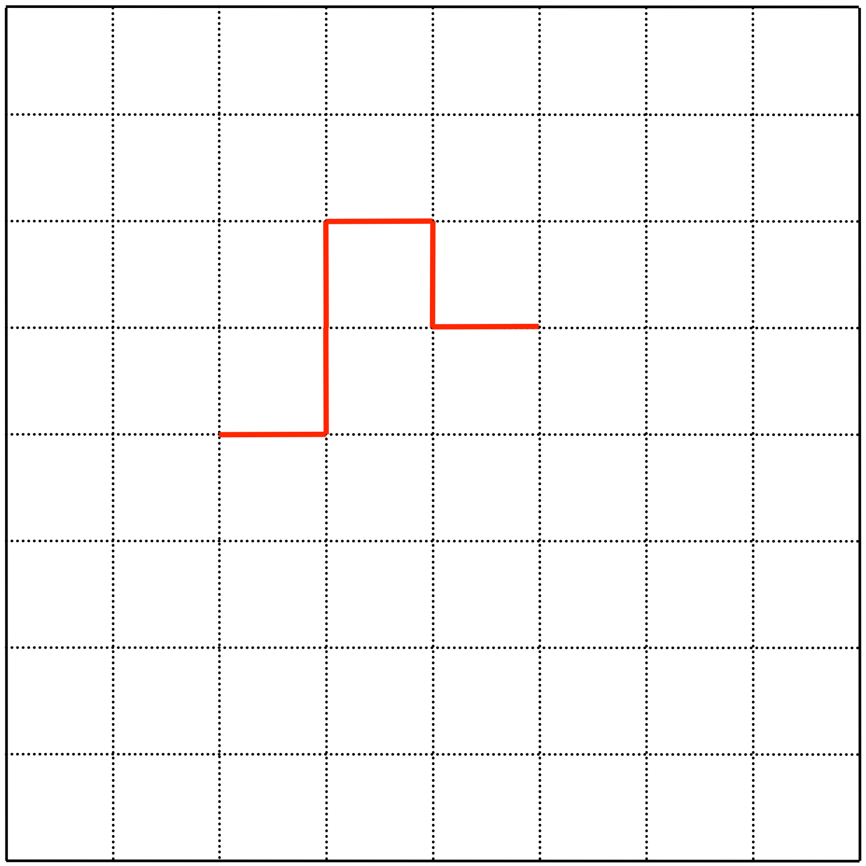
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

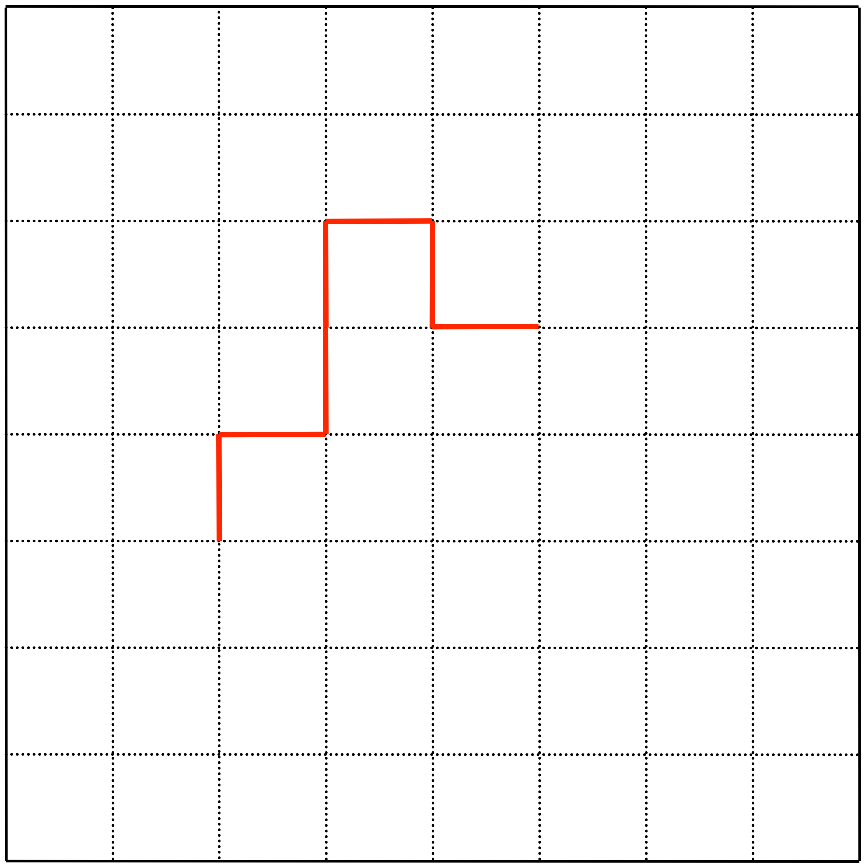
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

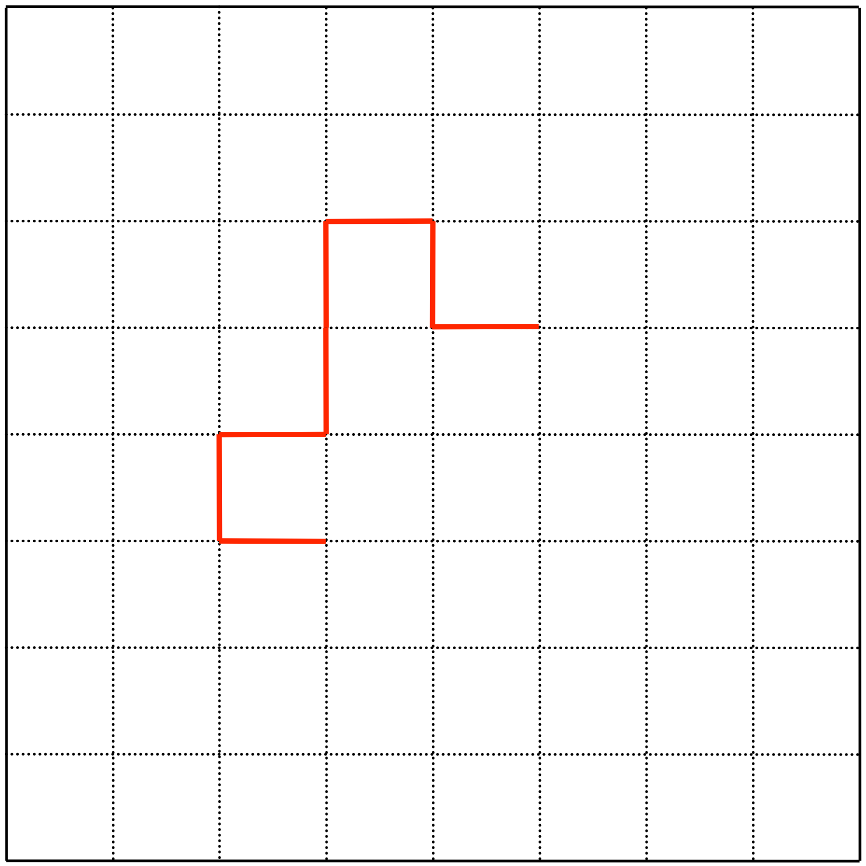
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

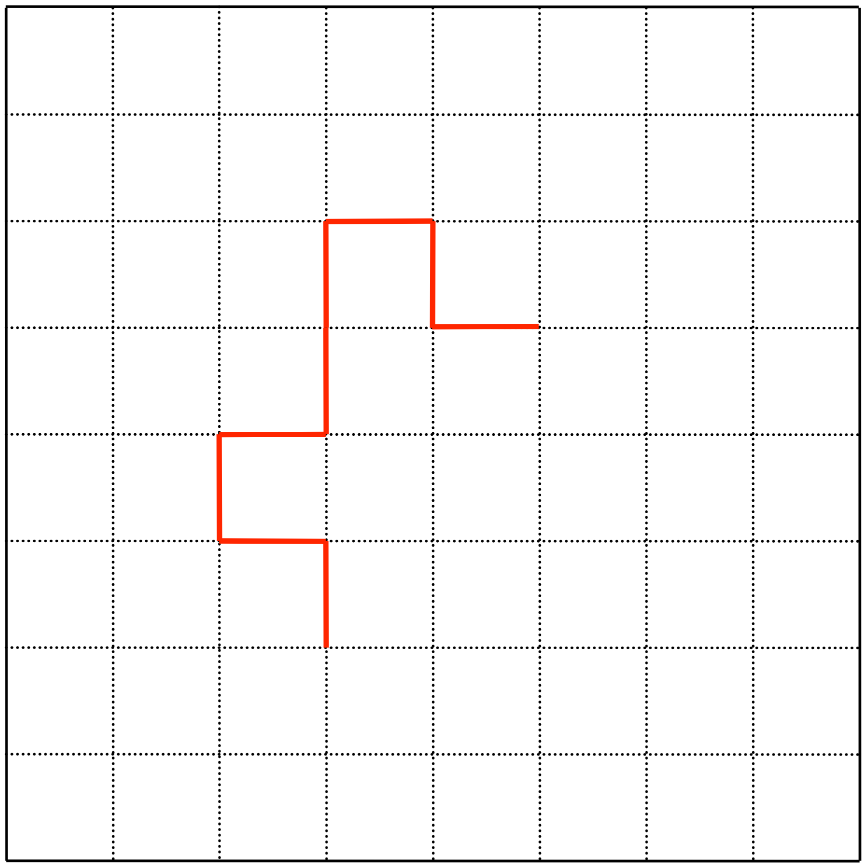
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

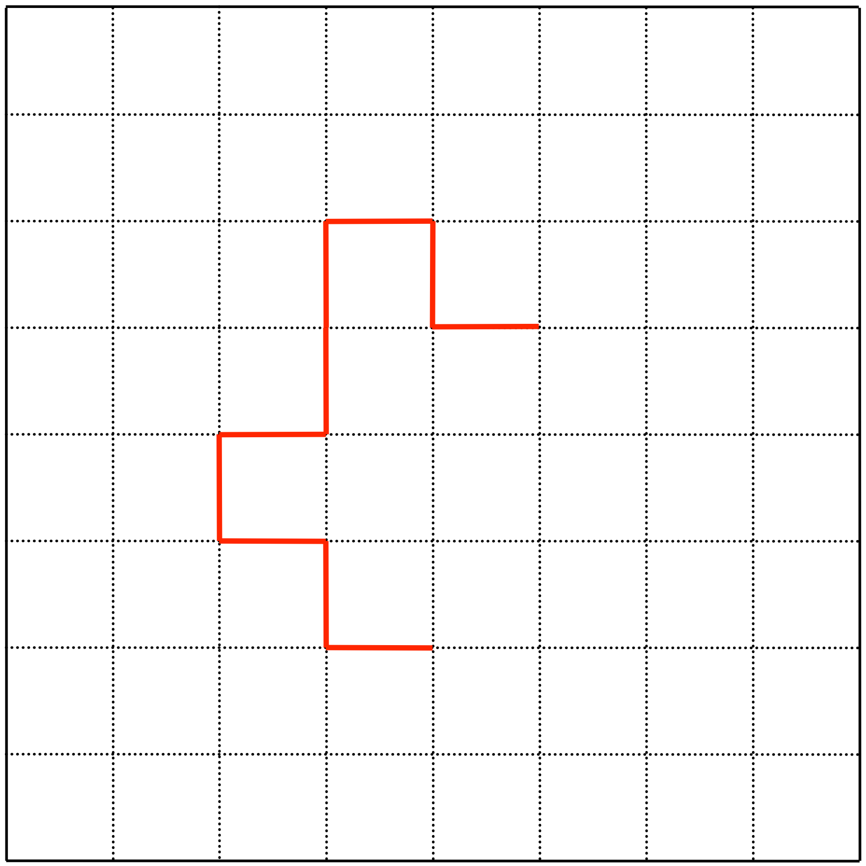
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

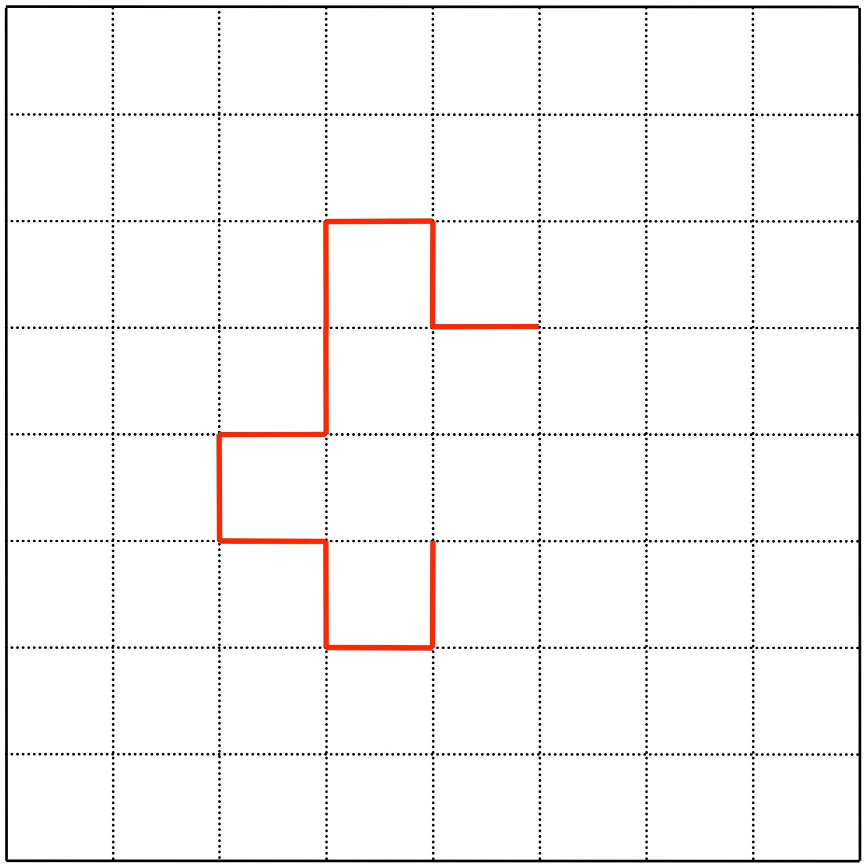
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

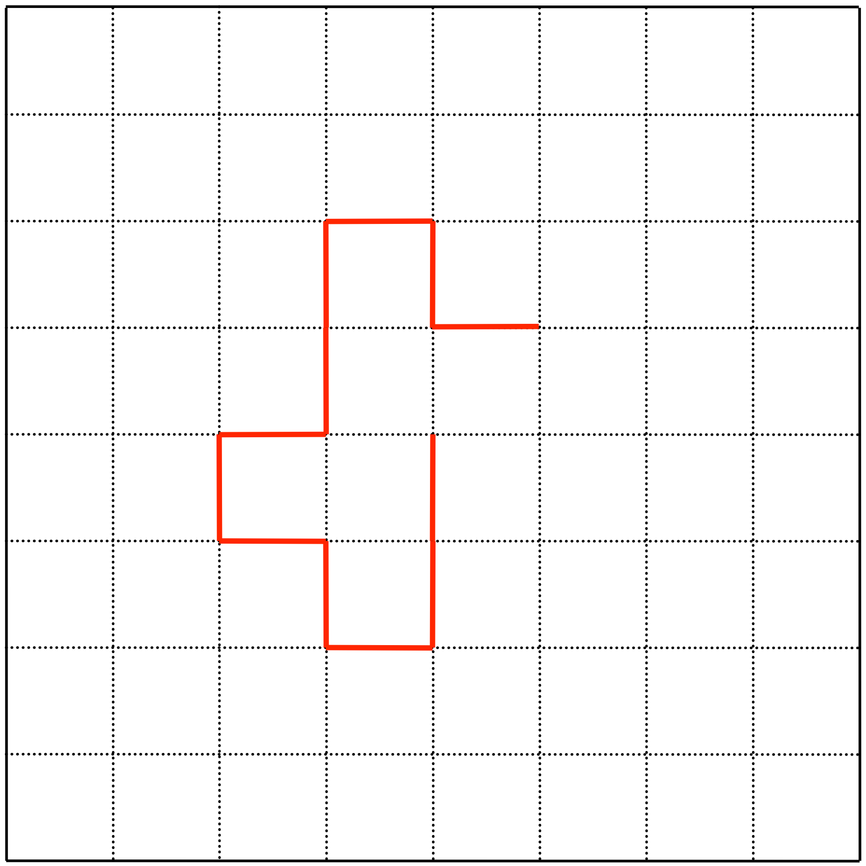
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

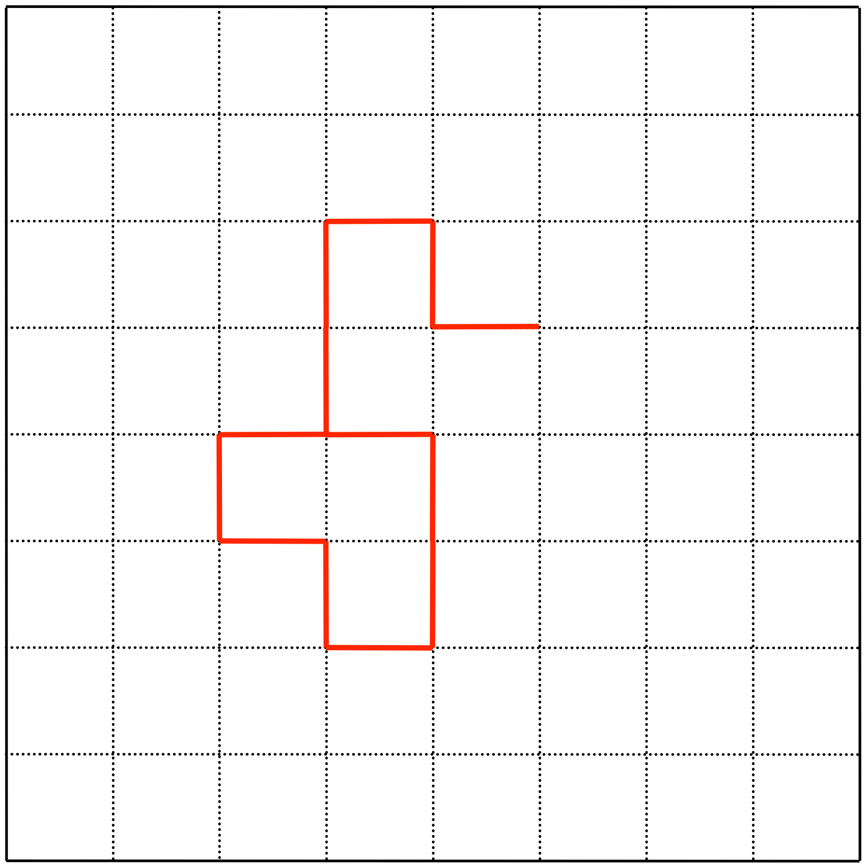
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

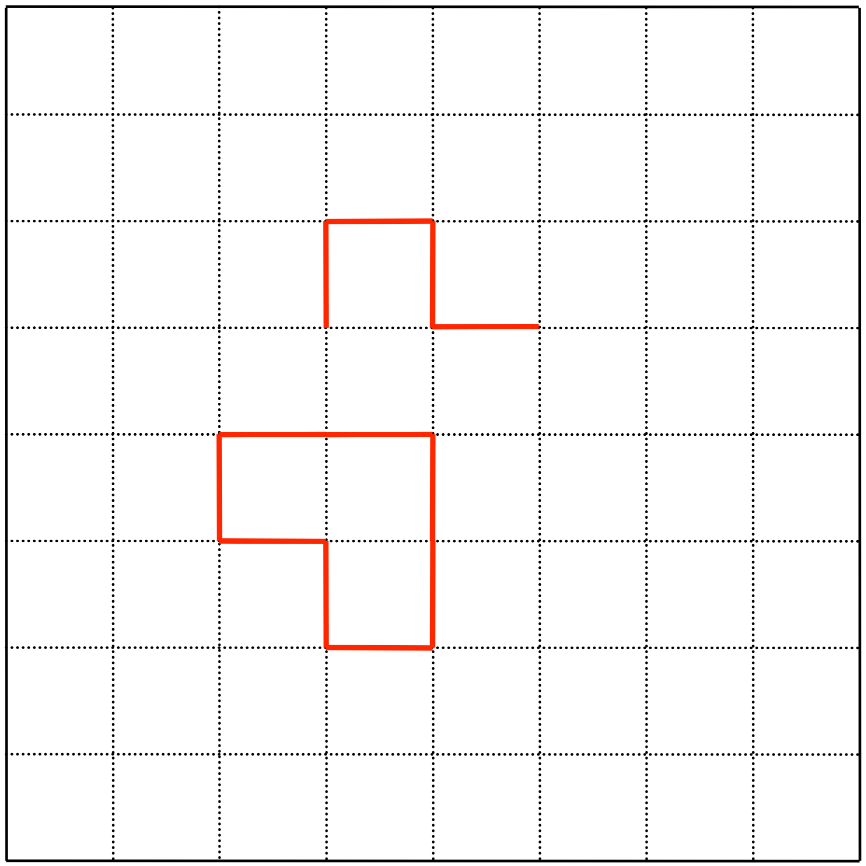
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

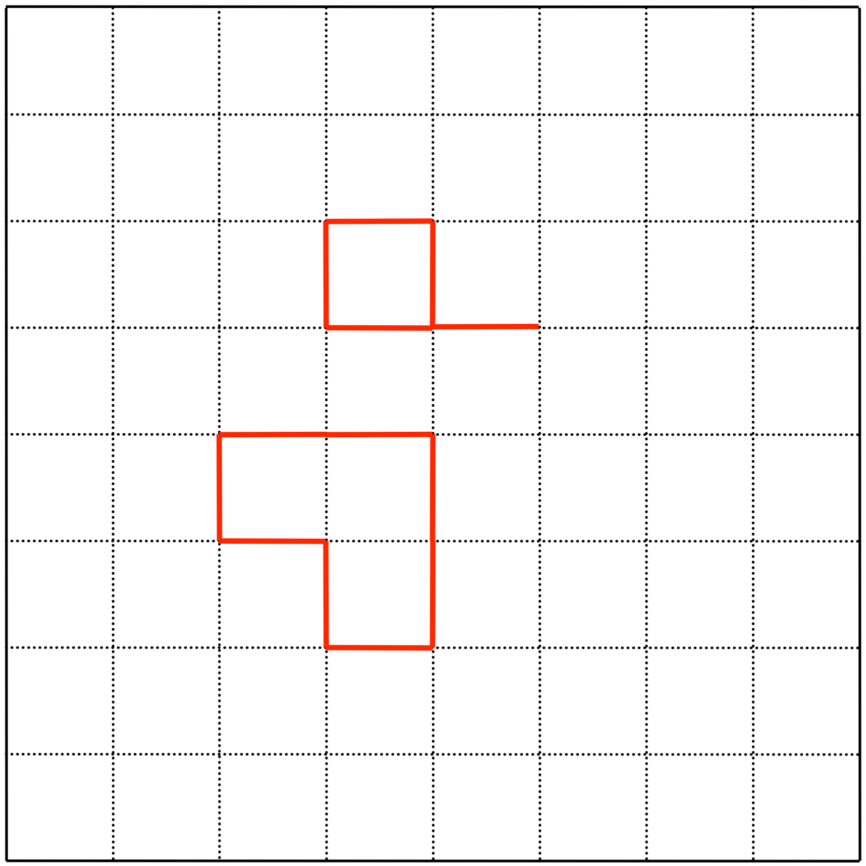
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

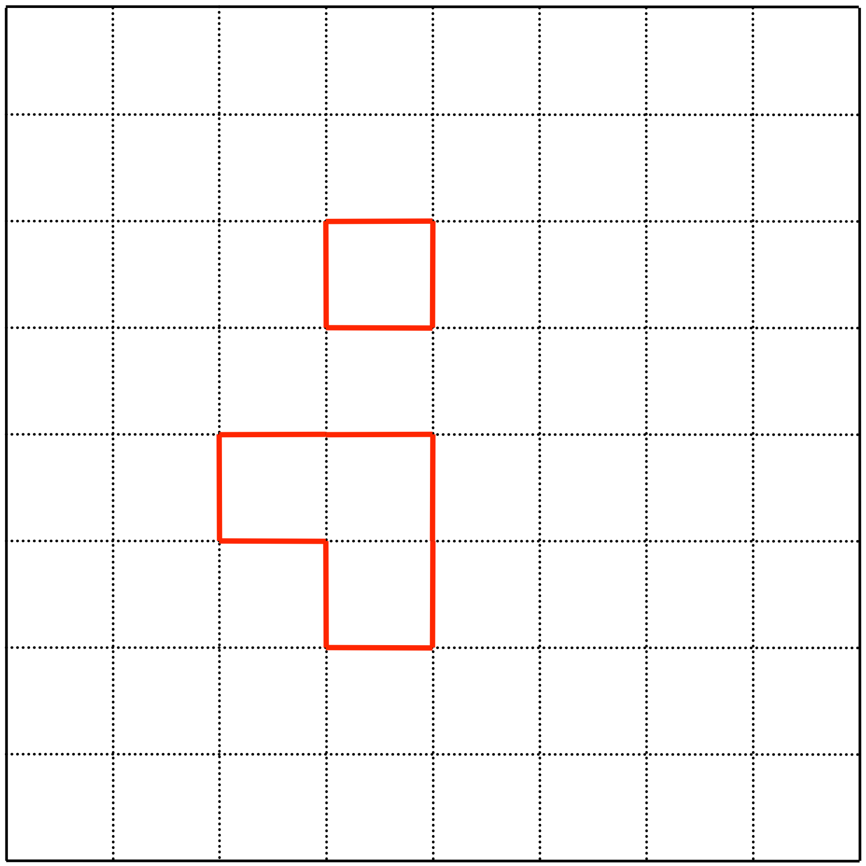
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

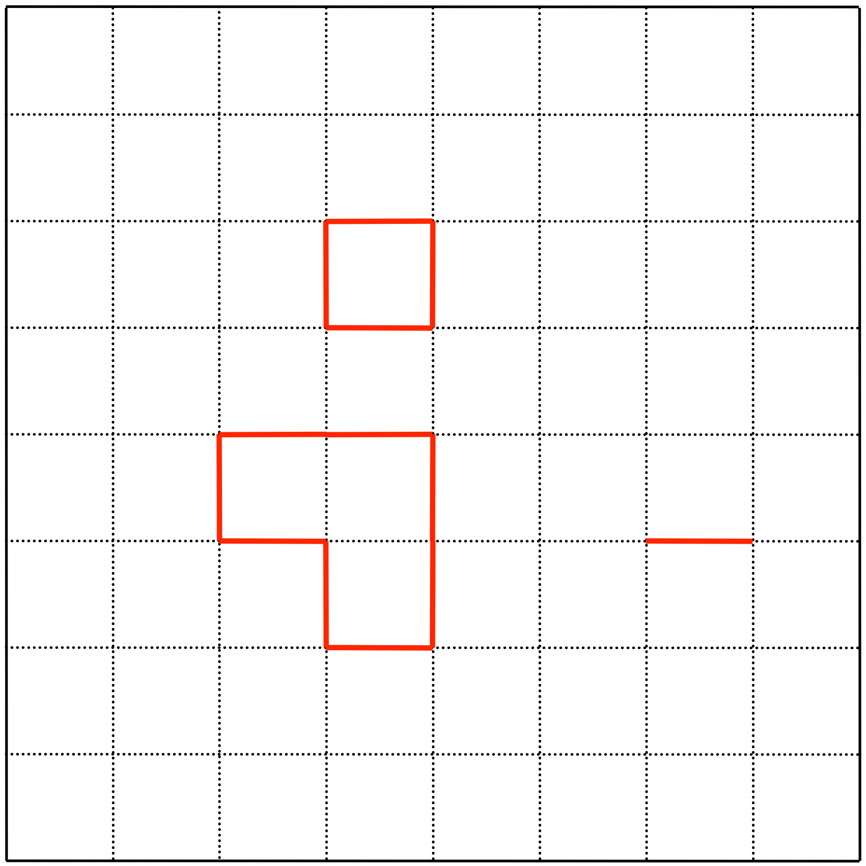
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

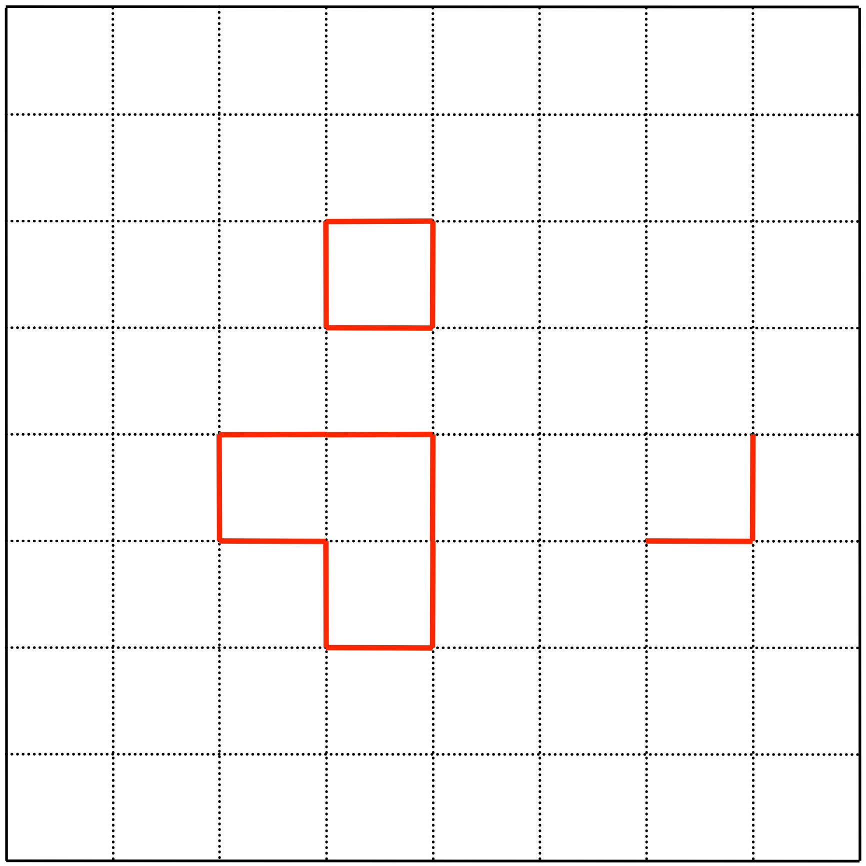
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

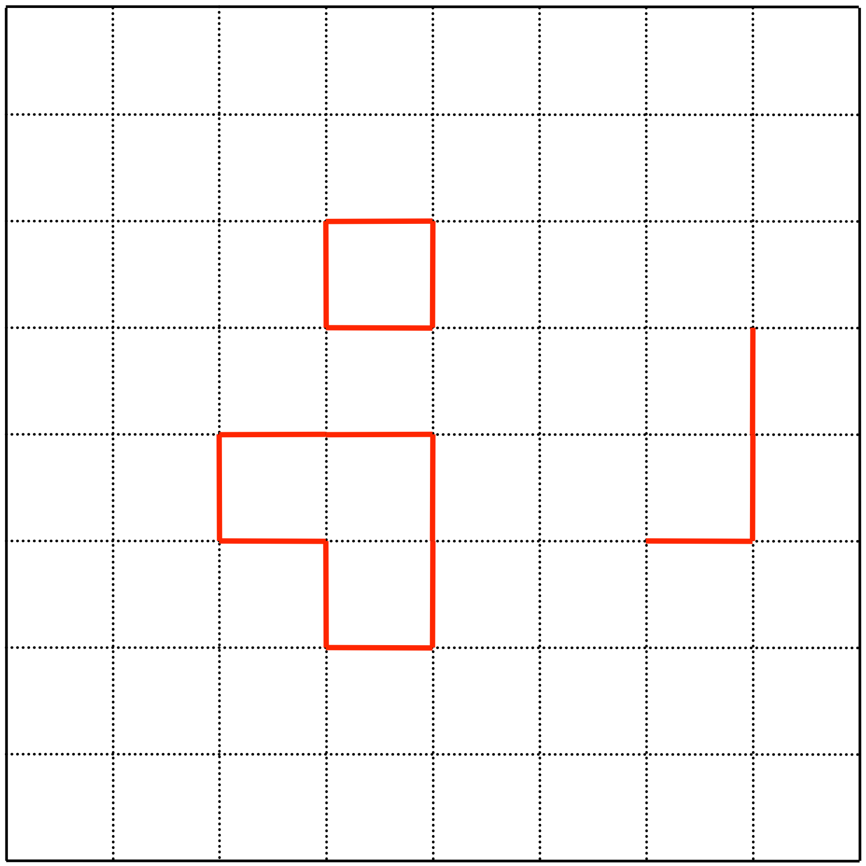
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

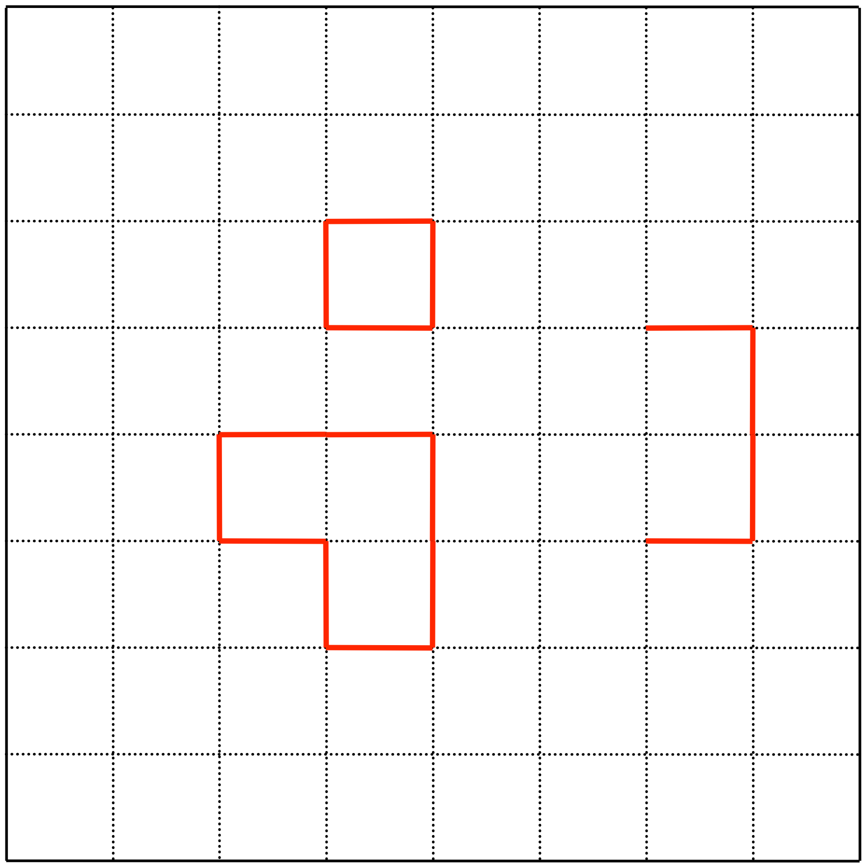
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

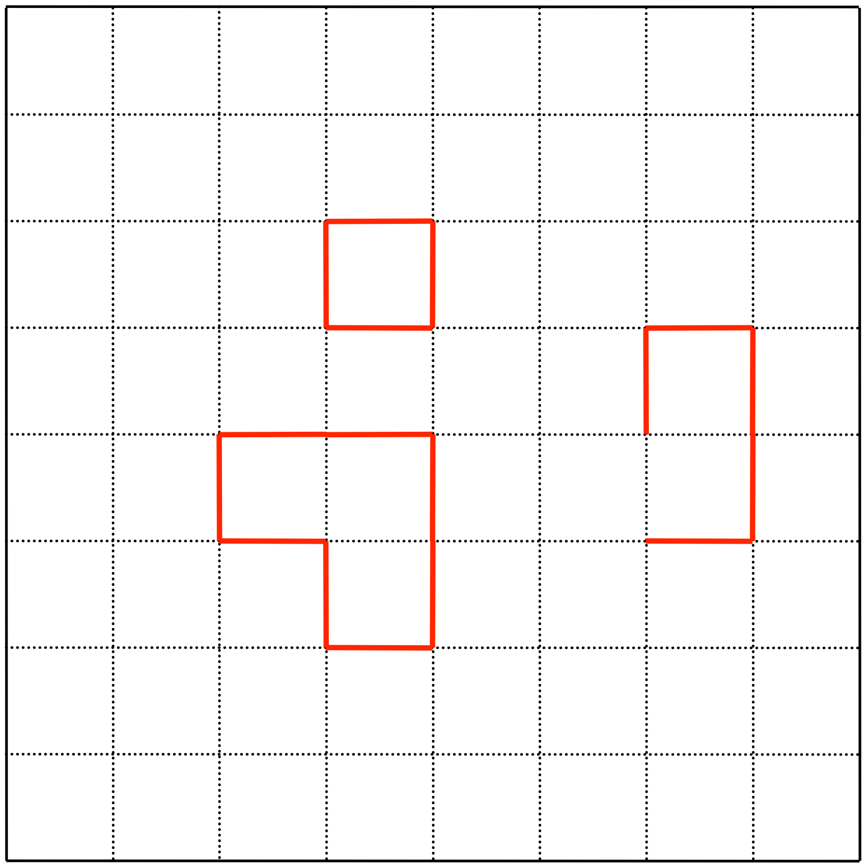
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

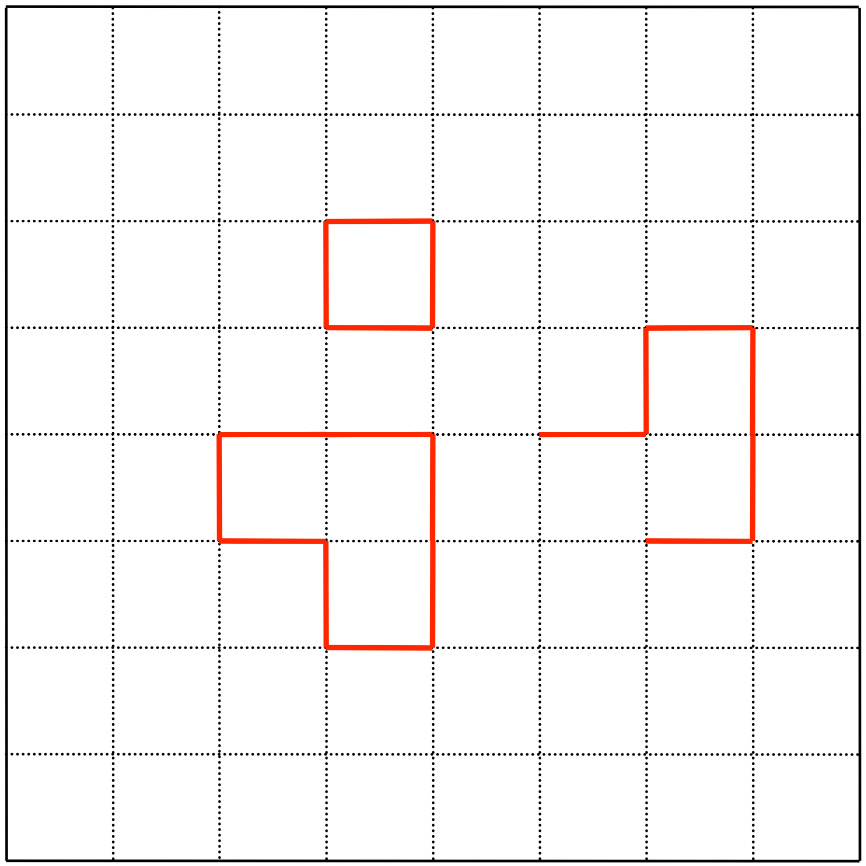
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

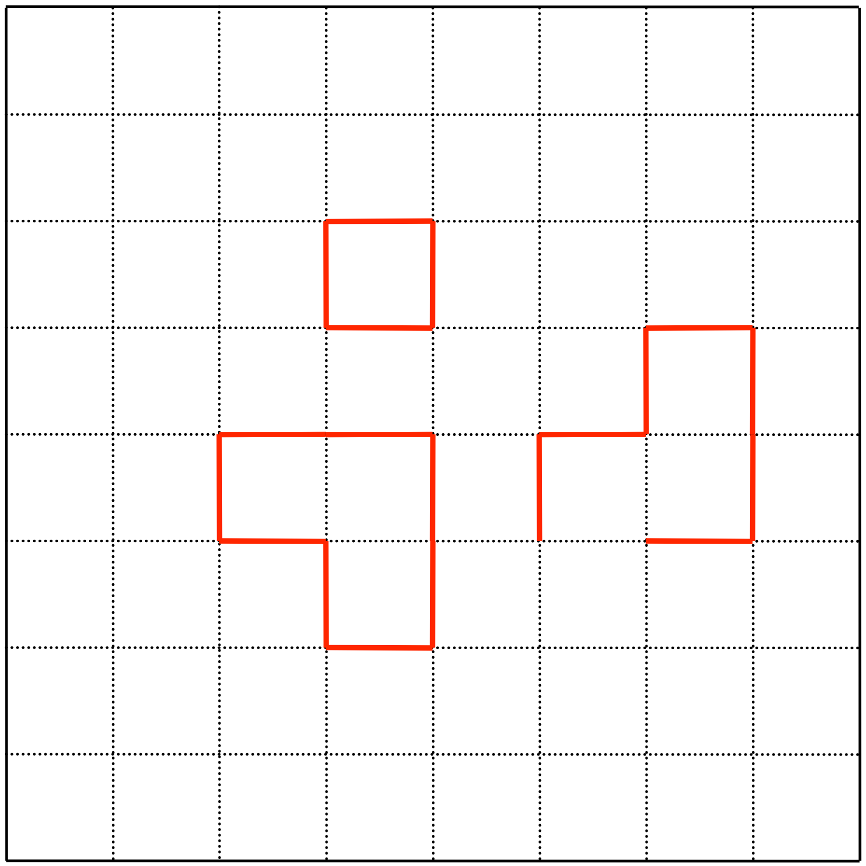
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

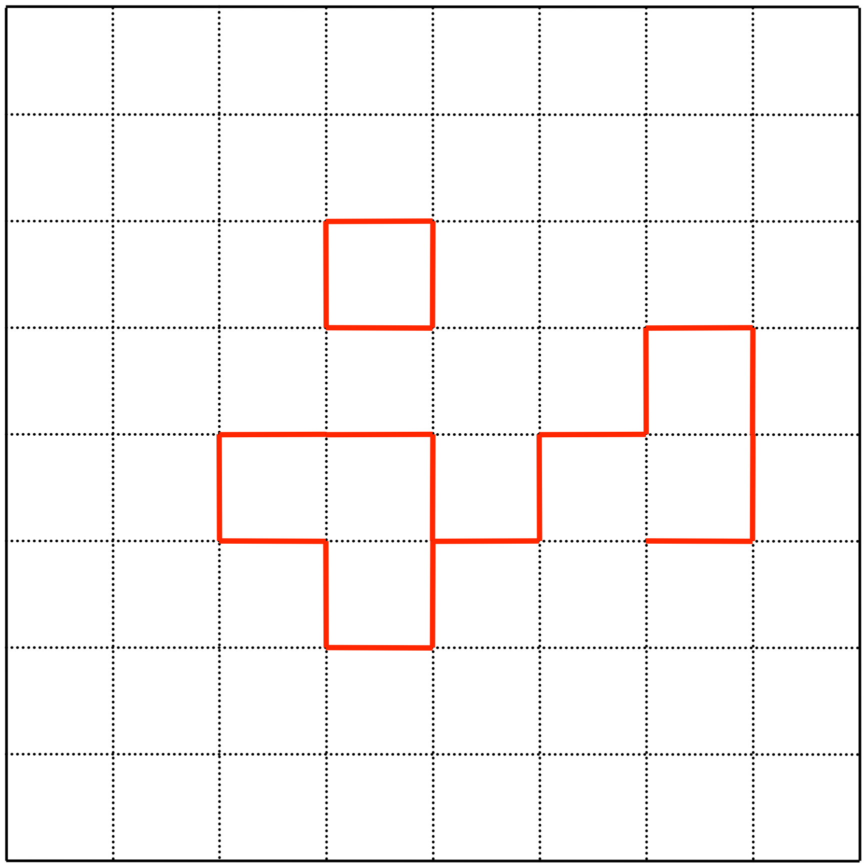
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

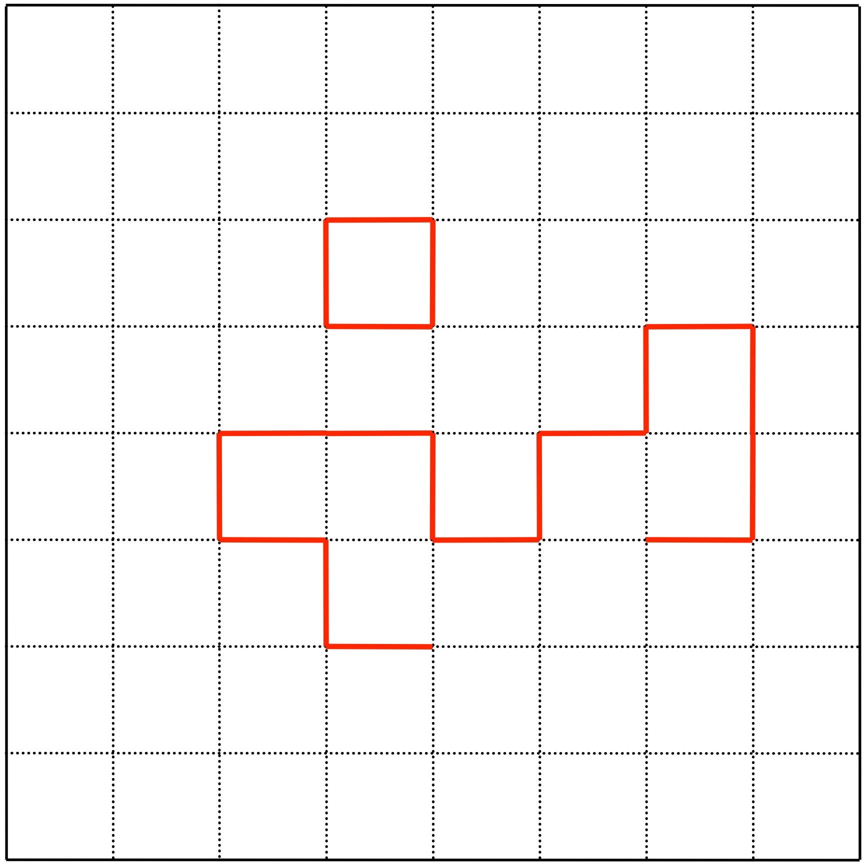
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

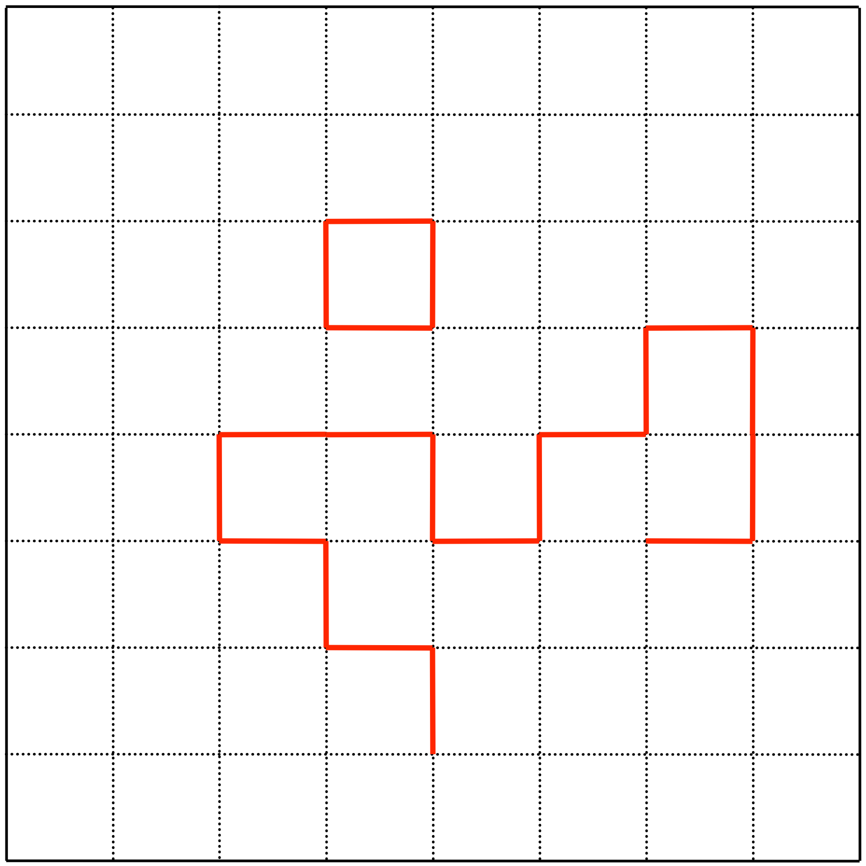
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

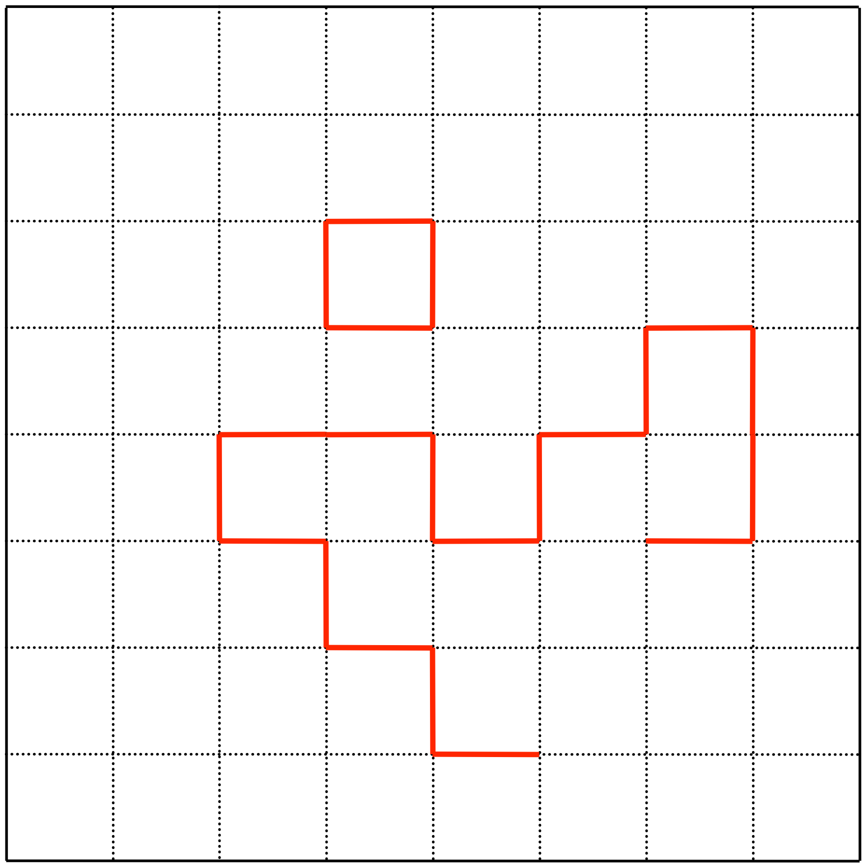
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

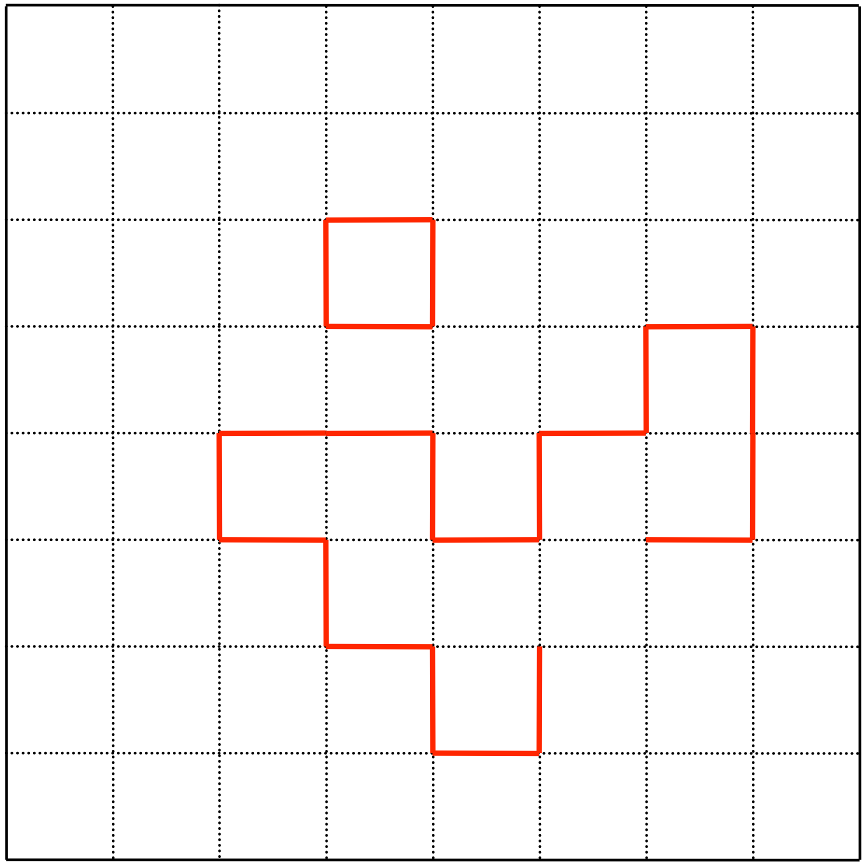
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

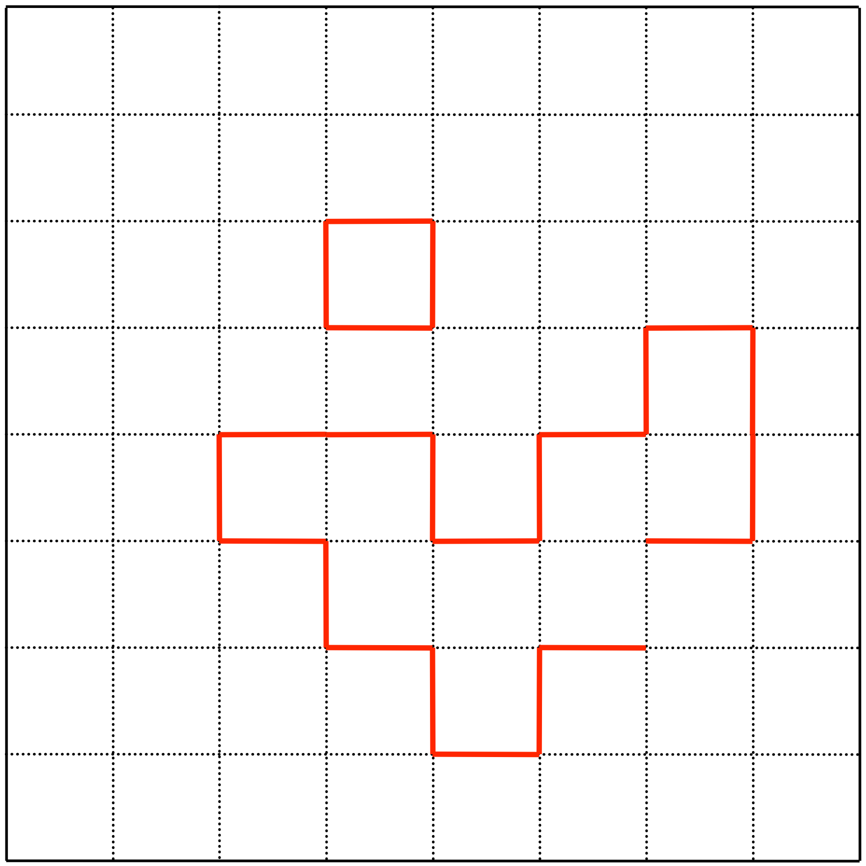
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

faster than cluster updates

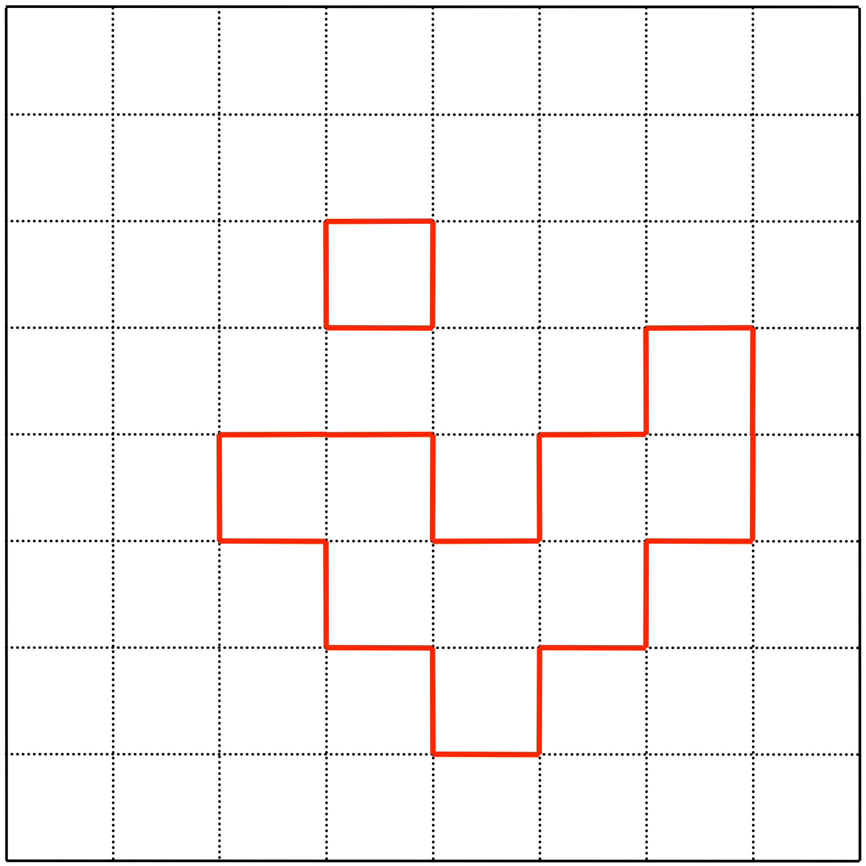
Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

Ordered state: macroscopic entangled loops

# Classical worm algorithm

Prokof'ev and Svistunov, PRL (2001)



Correlator sector

↑↓

Partition function sector

No critical slowing down

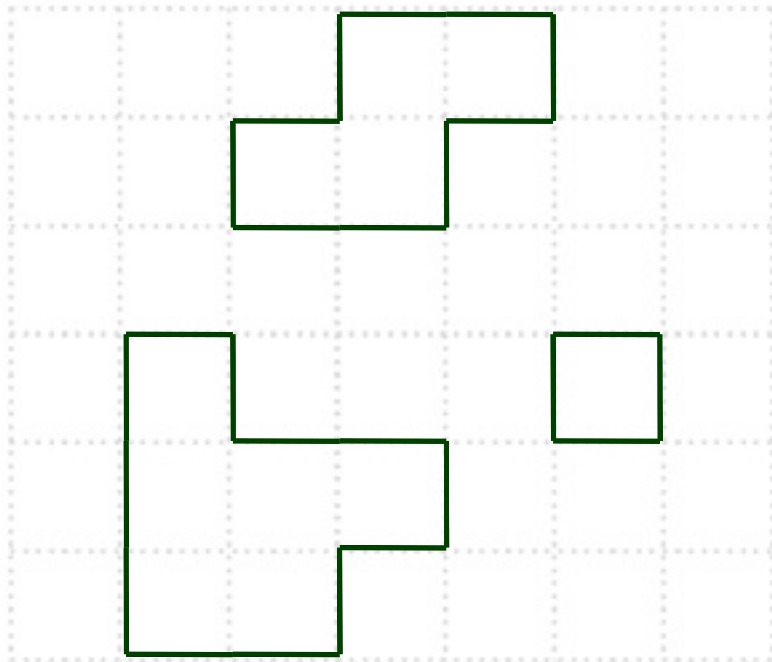
faster than cluster updates

Correlator = distribution function for the ends

Normal state: small loops, short distance between ends

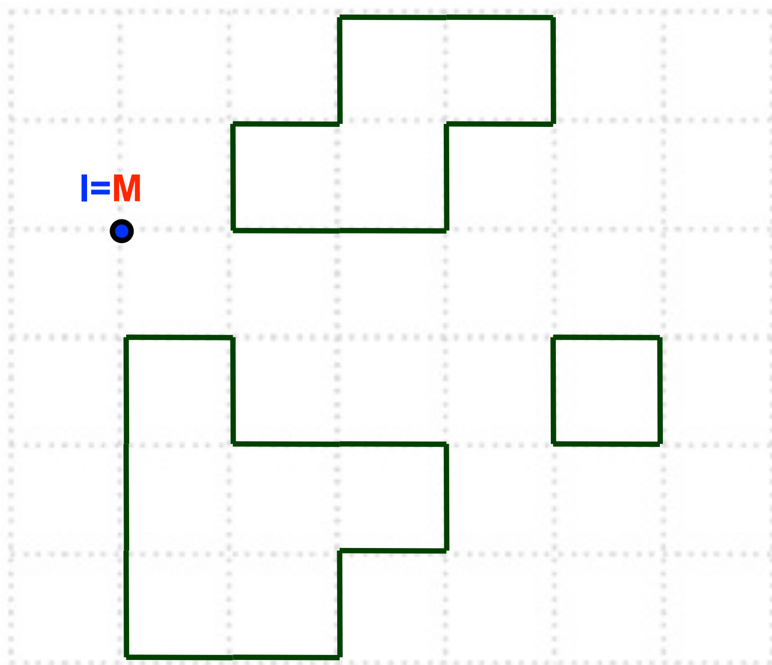
Ordered state: macroscopic entangled loops

# The complete algorithm



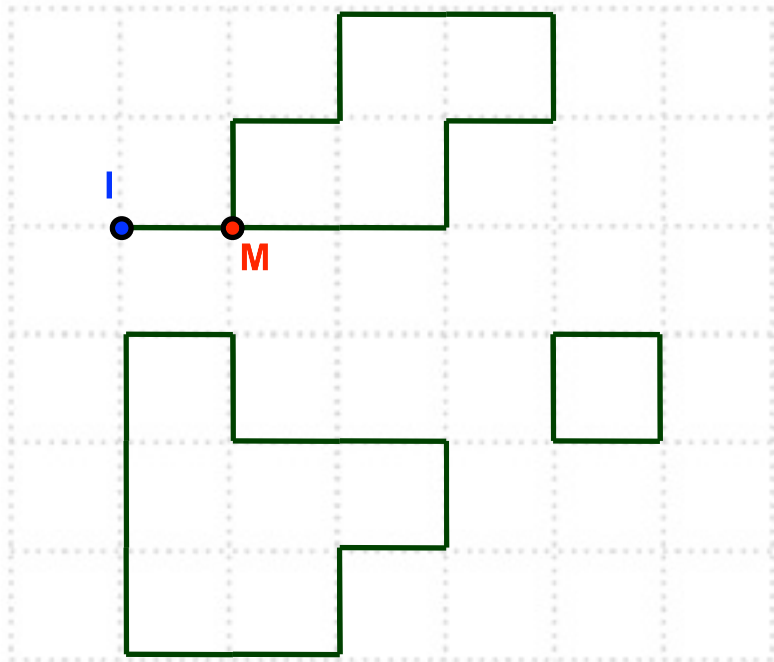
- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm



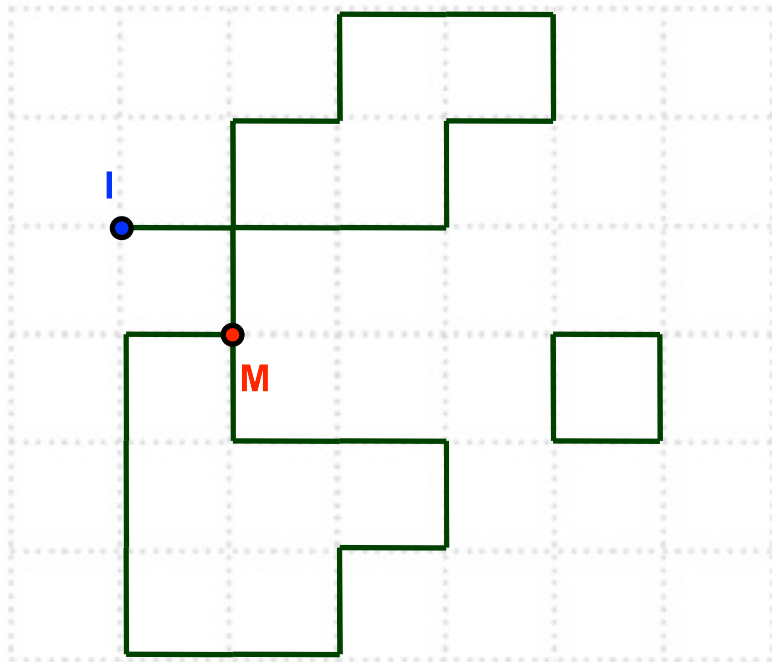

- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm



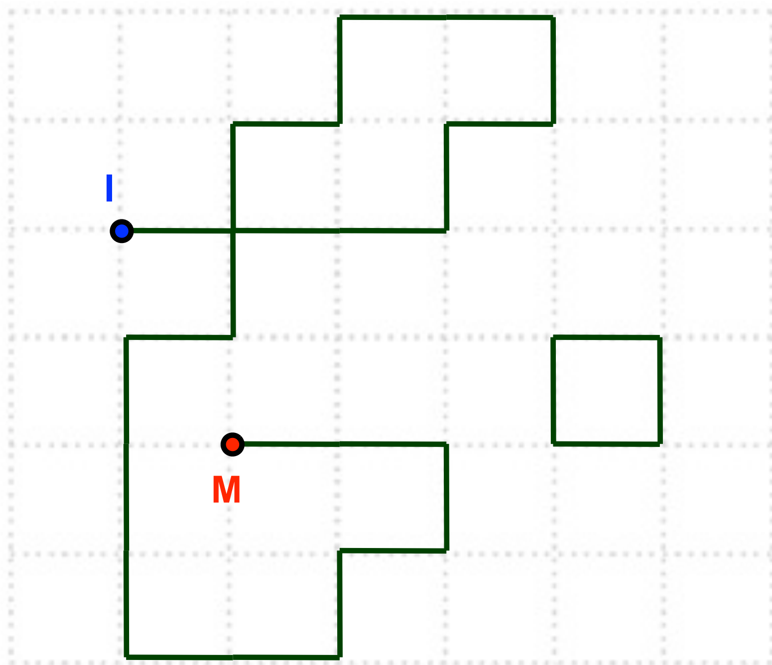

- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm



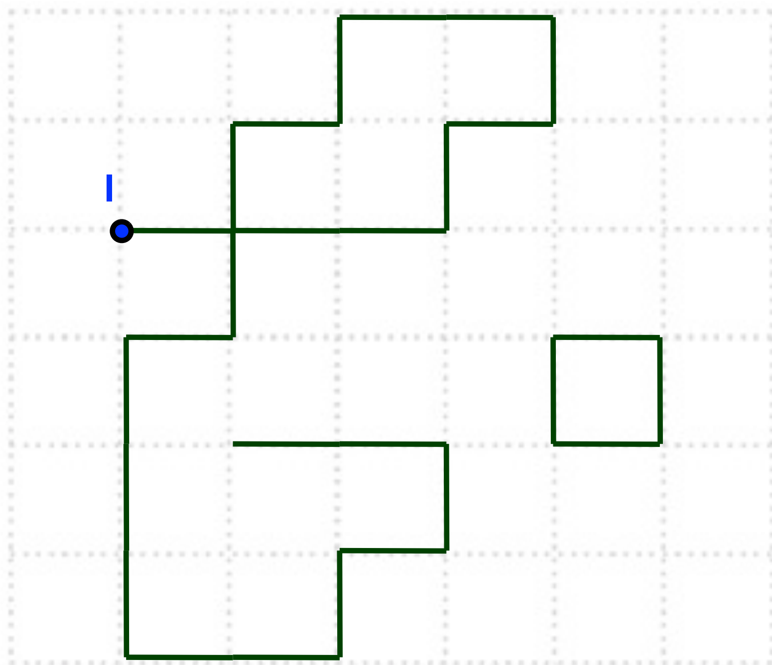

- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm



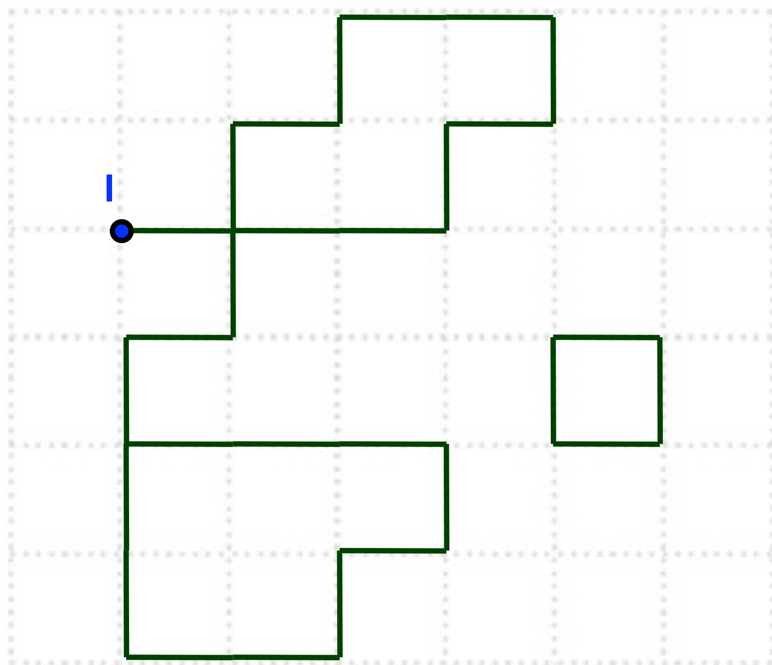

- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm



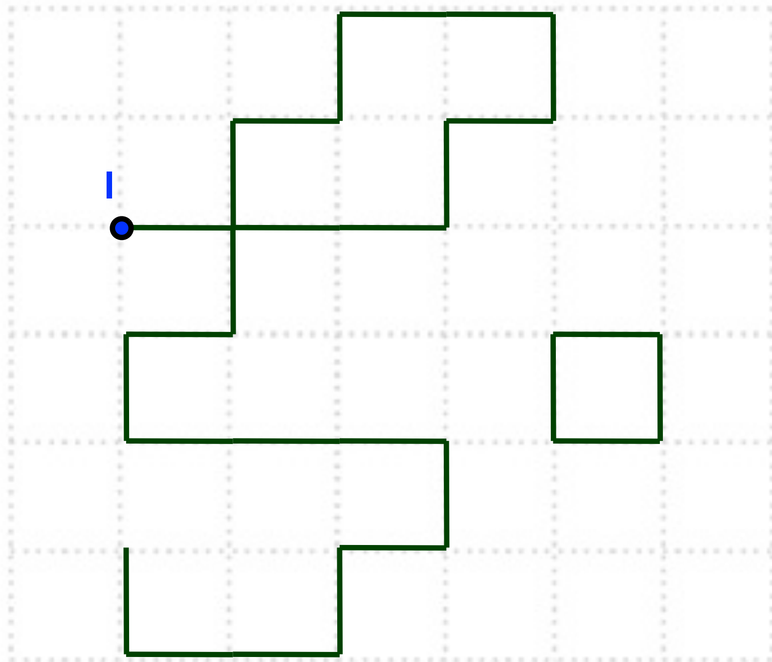

- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm



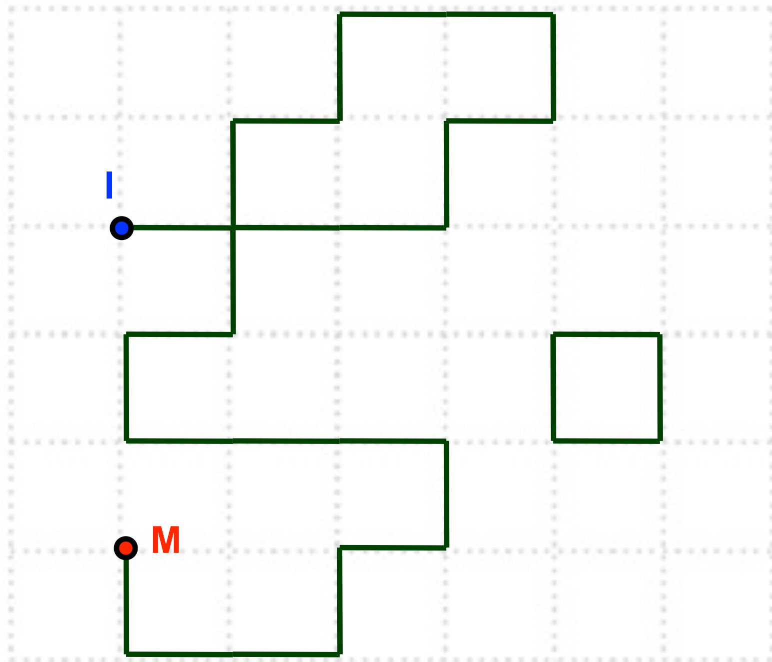

- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm




- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# The complete algorithm




- If I=M, select a new site for both at random
- Otherwise move I or M in a random direction, with acceptance rates
  - min [1,tanh($J/T$)] for n=0 -> n=1
  - min [1,1/tanh($J/T$)] for n=0 -> n=1
- Easier to implement than local updates but faster than cluster updates

# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

$$H \leftarrow H + \eta \sum_i \left( c_i^\dagger + c_i \right) \qquad H \leftarrow H + \eta \sum_i \left( S_i^+ + S_i^- \right)$$

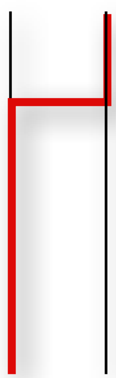
  - move these operators ("Ira" and "Masha") using local moves
  - until Ira and Masha meet

# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

$$H \leftarrow H + \eta \sum_i \left( c_i^\dagger + c_i \right) \qquad H \leftarrow H + \eta \sum_i \left( S_i^+ + S_i^- \right)$$

  - move these operators ("Ira" and "Masha") using local moves
  - until Ira and Masha meet



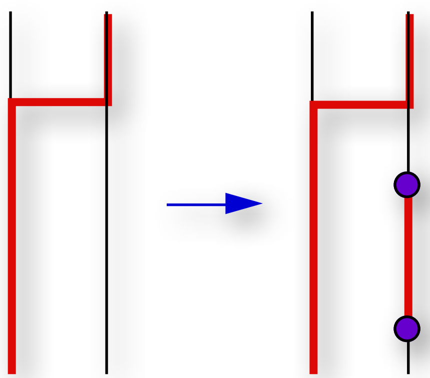
# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

$$H \leftarrow H + \eta \sum_i \left(c_i^\dagger + c_i\right) \qquad H \leftarrow H + \eta \sum_i \left(S_i^+ + S_i^-\right)$$

  - move these operators (“Ira” and “Masha”) using local moves
  - until Ira and Masha meet

insert worm

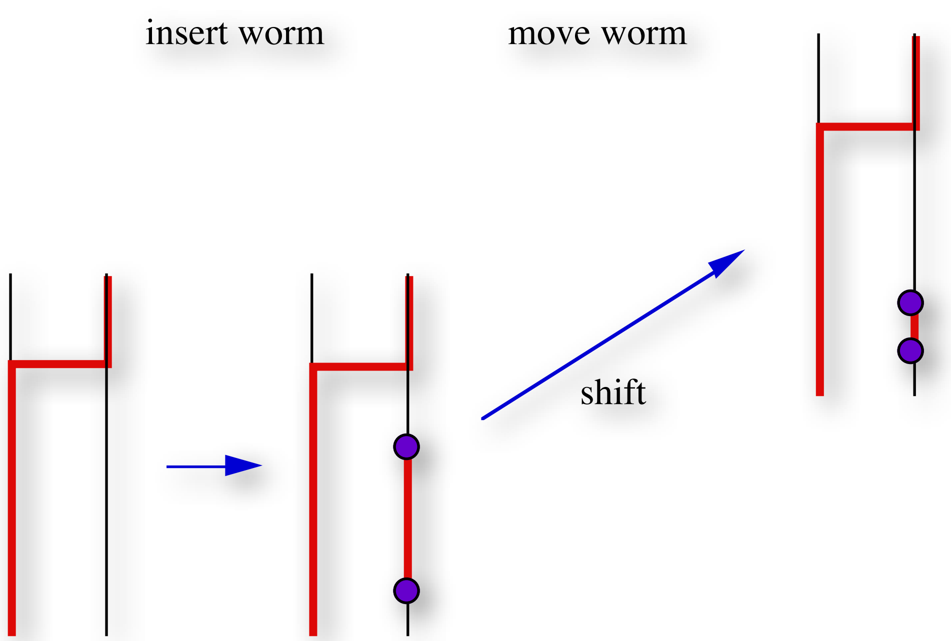


# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

$$H \leftarrow H + \eta \sum_i \left( c_i^\dagger + c_i \right) \qquad H \leftarrow H + \eta \sum_i \left( S_i^+ + S_i^- \right)$$

  - move these operators (“Ira” and “Masha”) using local moves
  - until Ira and Masha meet

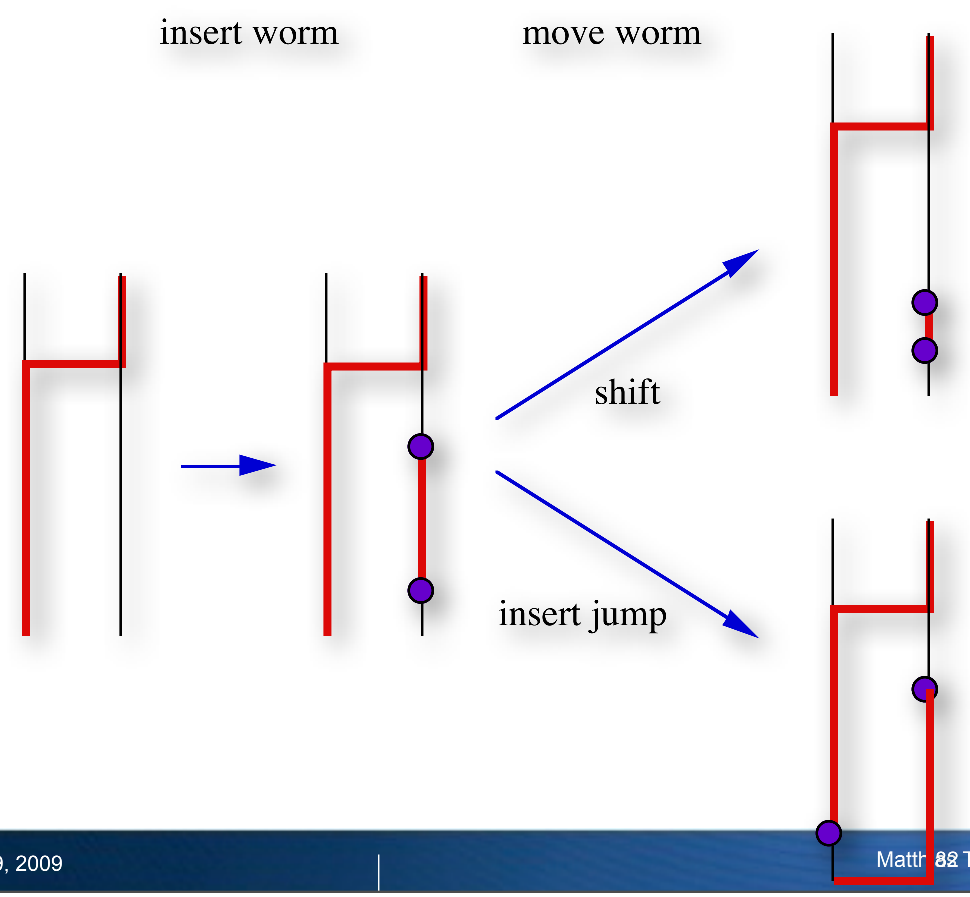


# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

$$H \leftarrow H + \eta \sum_i \left(c_i^\dagger + c_i\right) \qquad H \leftarrow H + \eta \sum_i \left(S_i^+ + S_i^-\right)$$

  - move these operators ("Ira" and "Masha") using local moves
  - until Ira and Masha meet

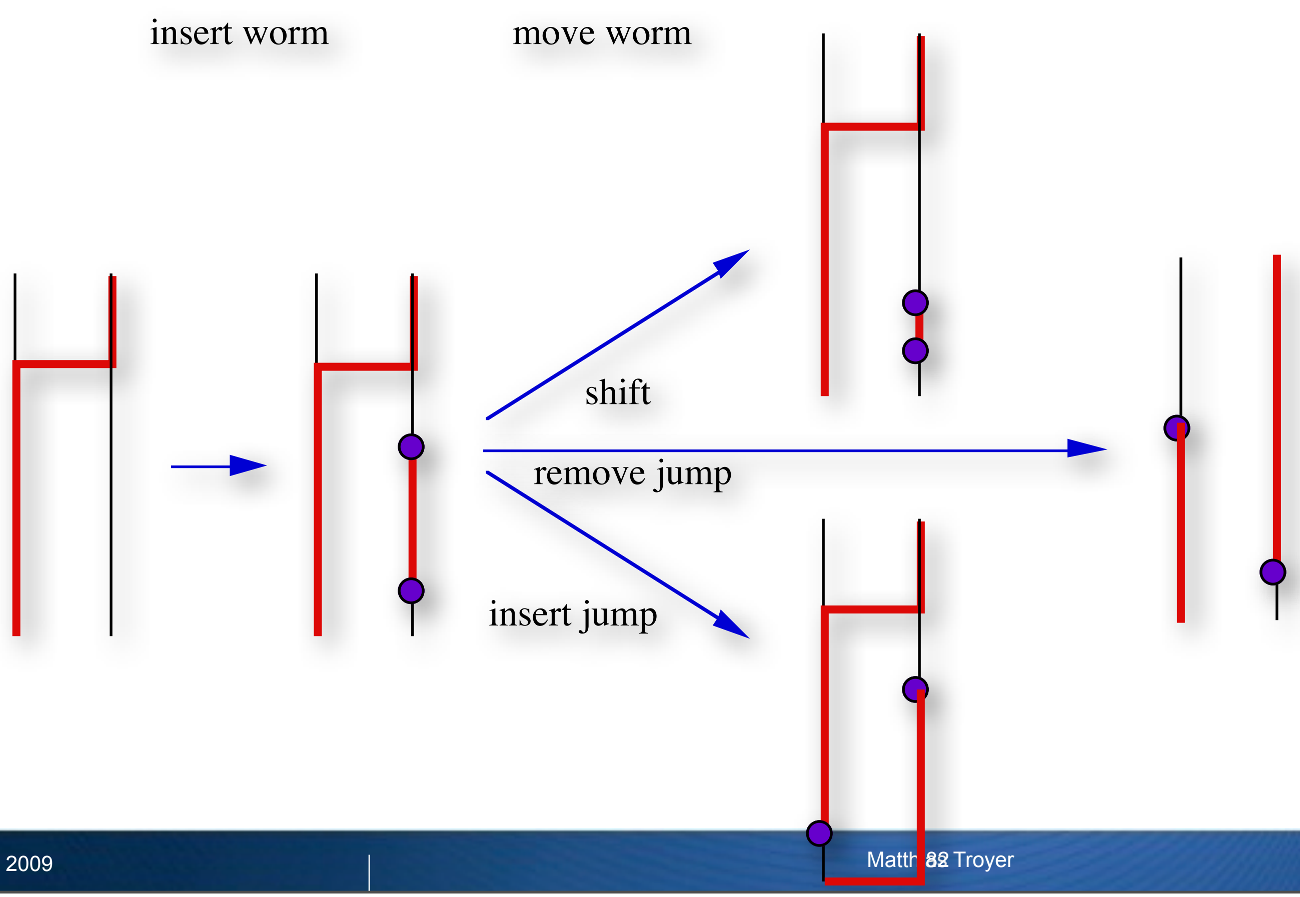


# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

$$H \leftarrow H + \eta \sum_i \left(c_i^\dagger + c_i\right) \qquad H \leftarrow H + \eta \sum_i \left(S_i^+ + S_i^-\right)$$

  - move these operators (“Ira” and “Masha”) using local moves
  - until Ira and Masha meet

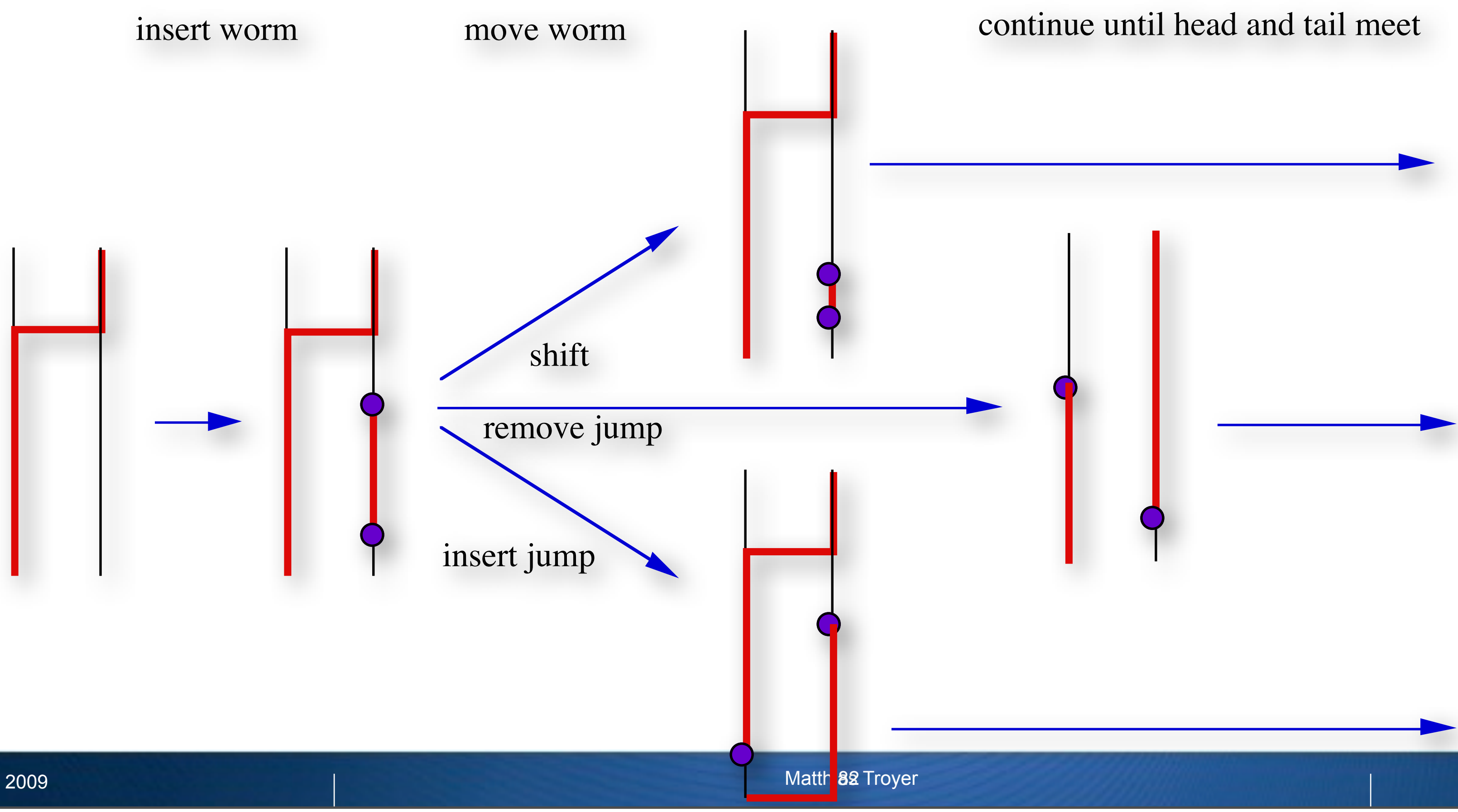


# Worm updates

- Break a world line by inserting a pair of creation/annihilation operators

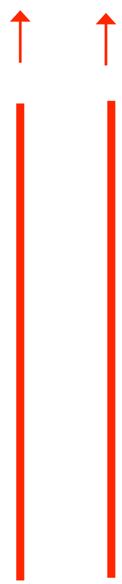
$$H \leftarrow H + \eta \sum_i \left(c_i^\dagger + c_i\right) \qquad H \leftarrow H + \eta \sum_i \left(S_i^+ + S_i^-\right)$$

  - move these operators ("Ira" and "Masha") using local moves
  - until Ira and Masha meet



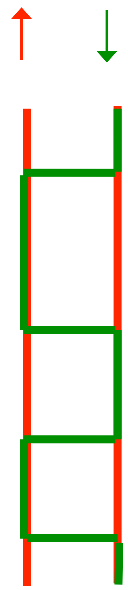
# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

$E = J/4 - h = -3/4$

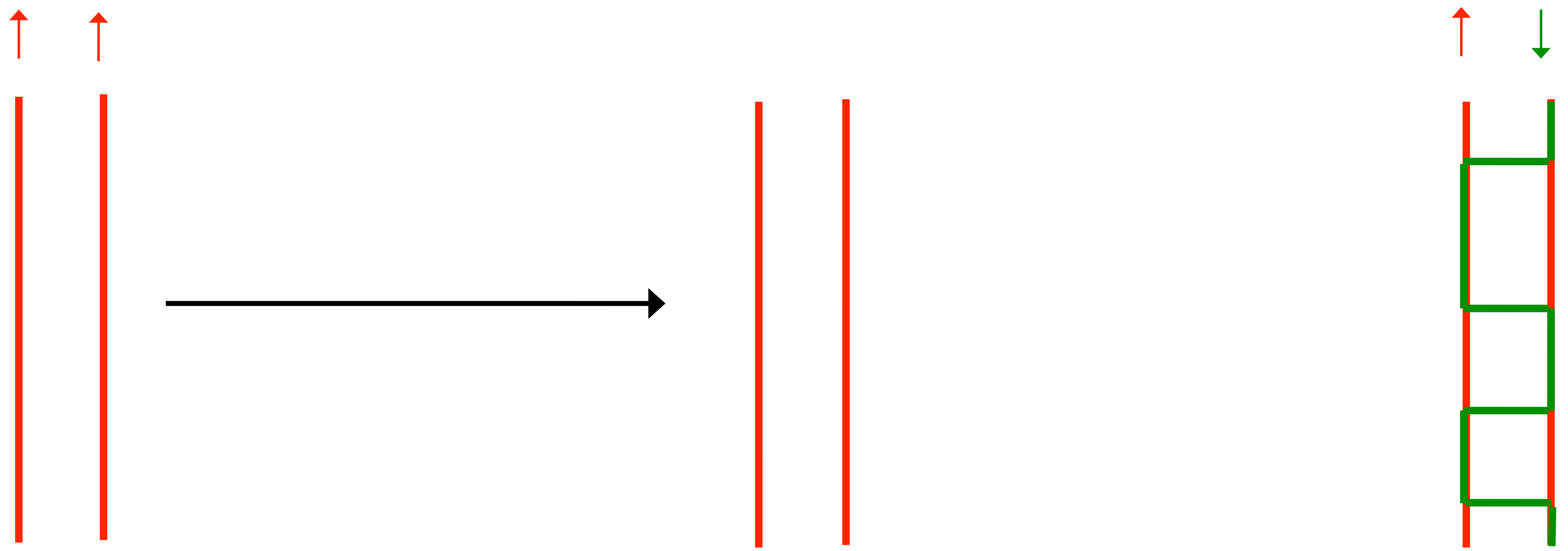


Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in  a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

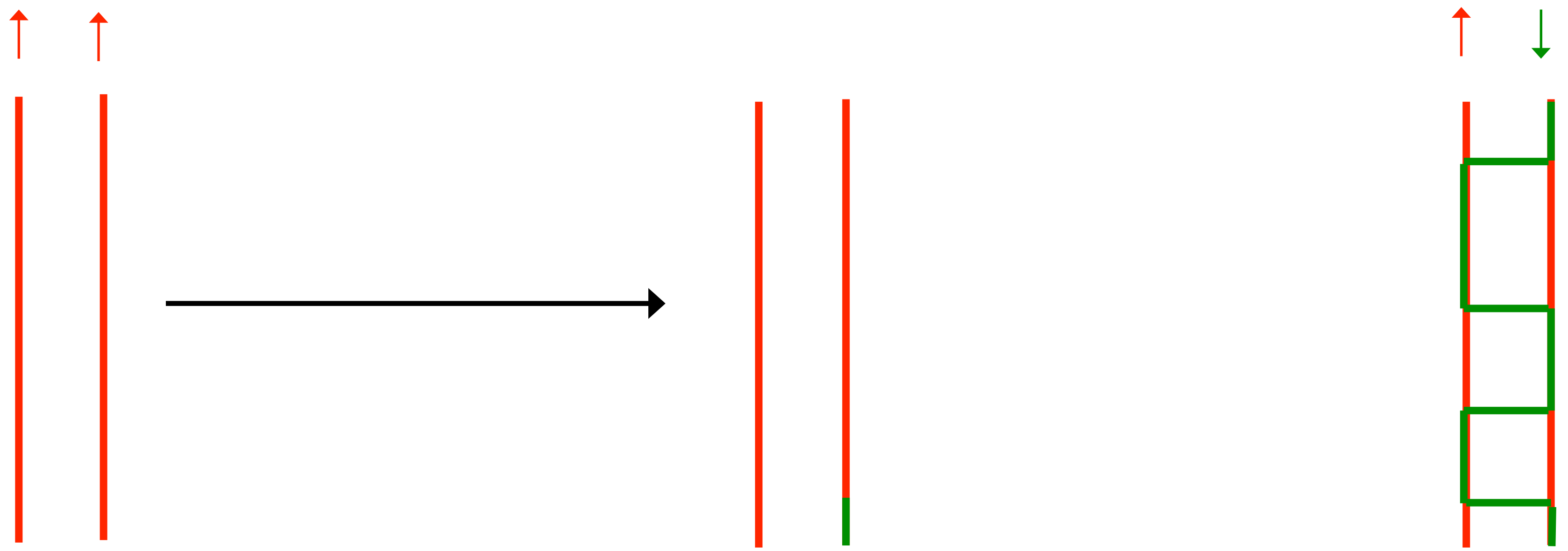
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

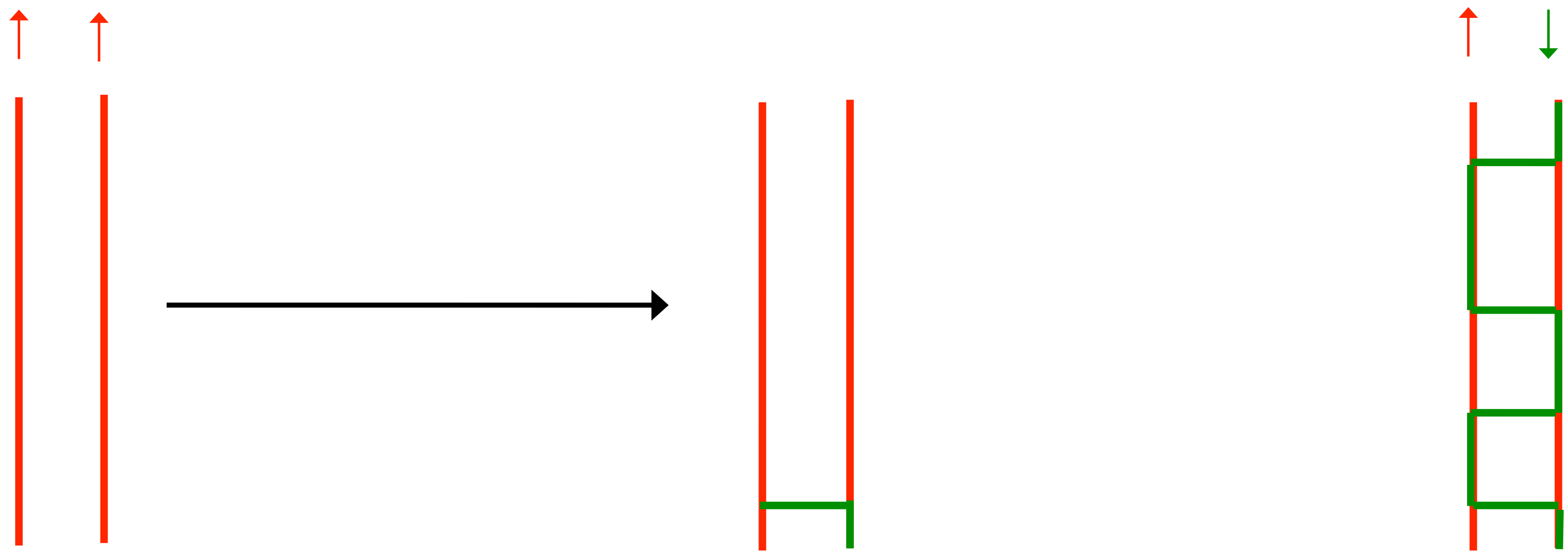
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

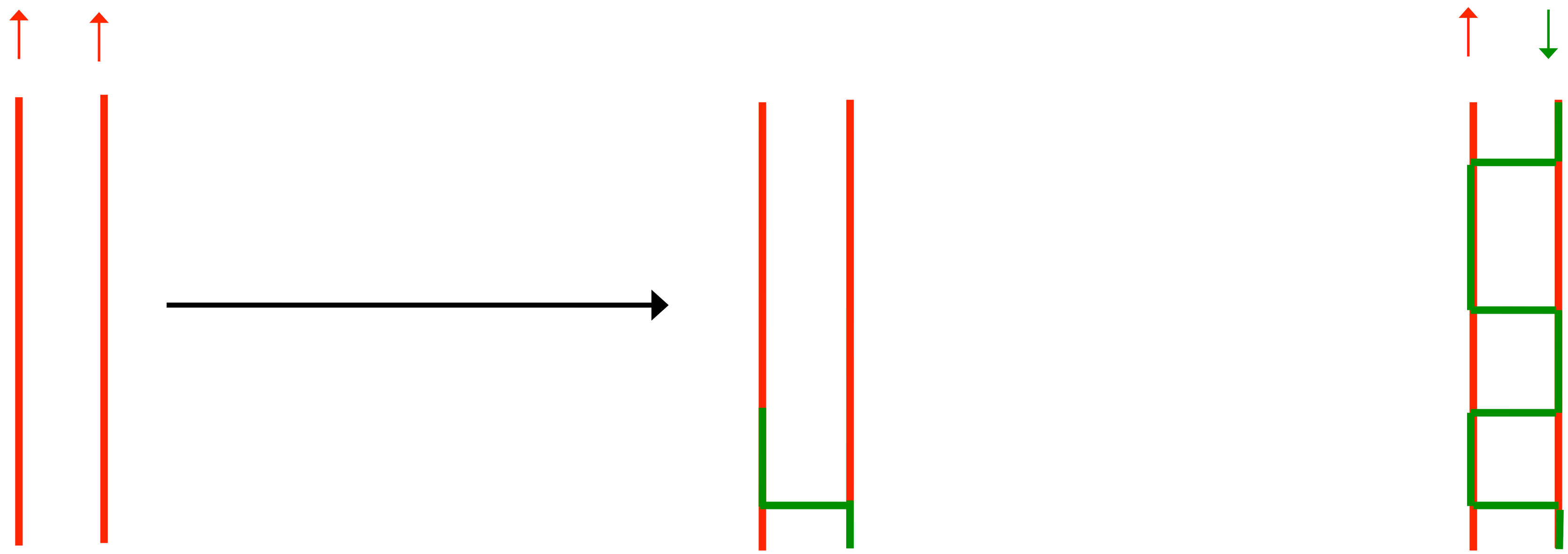
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

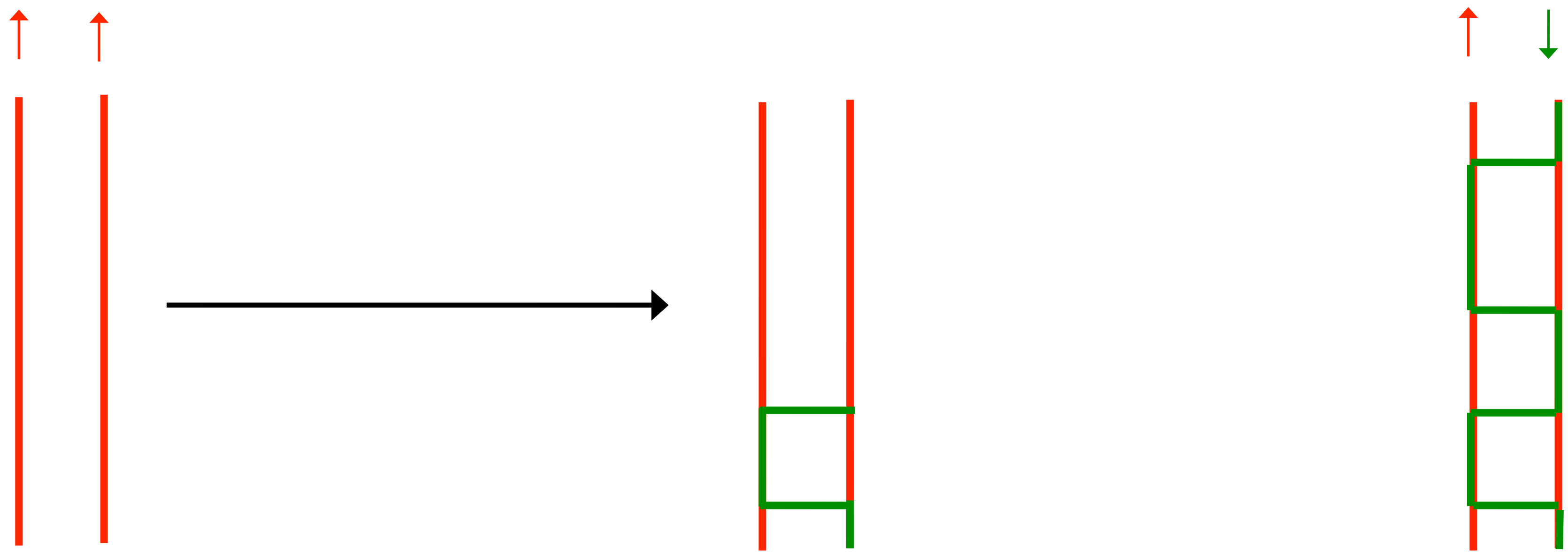
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

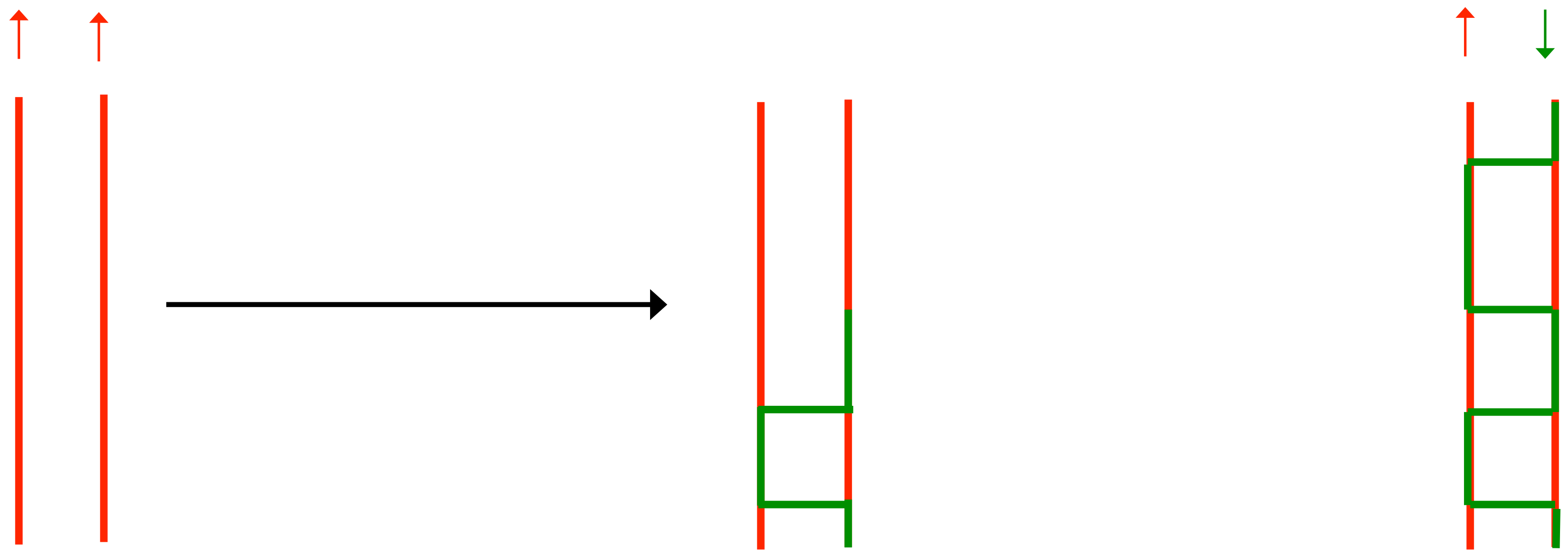
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

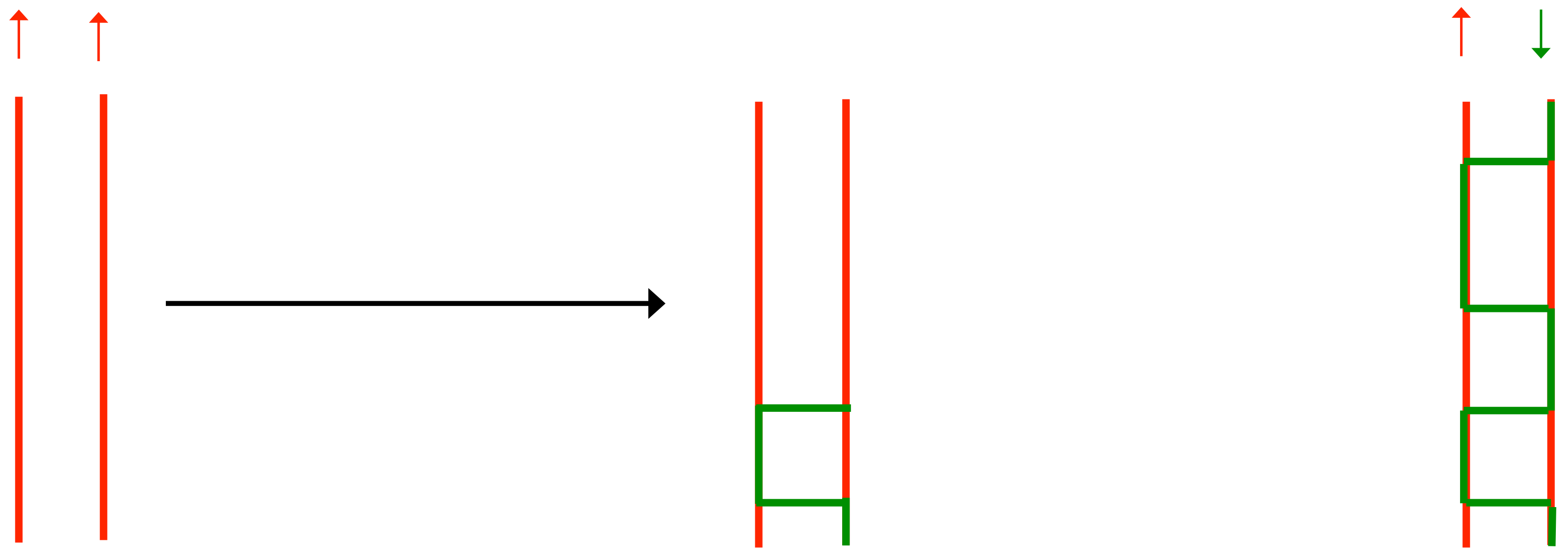
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

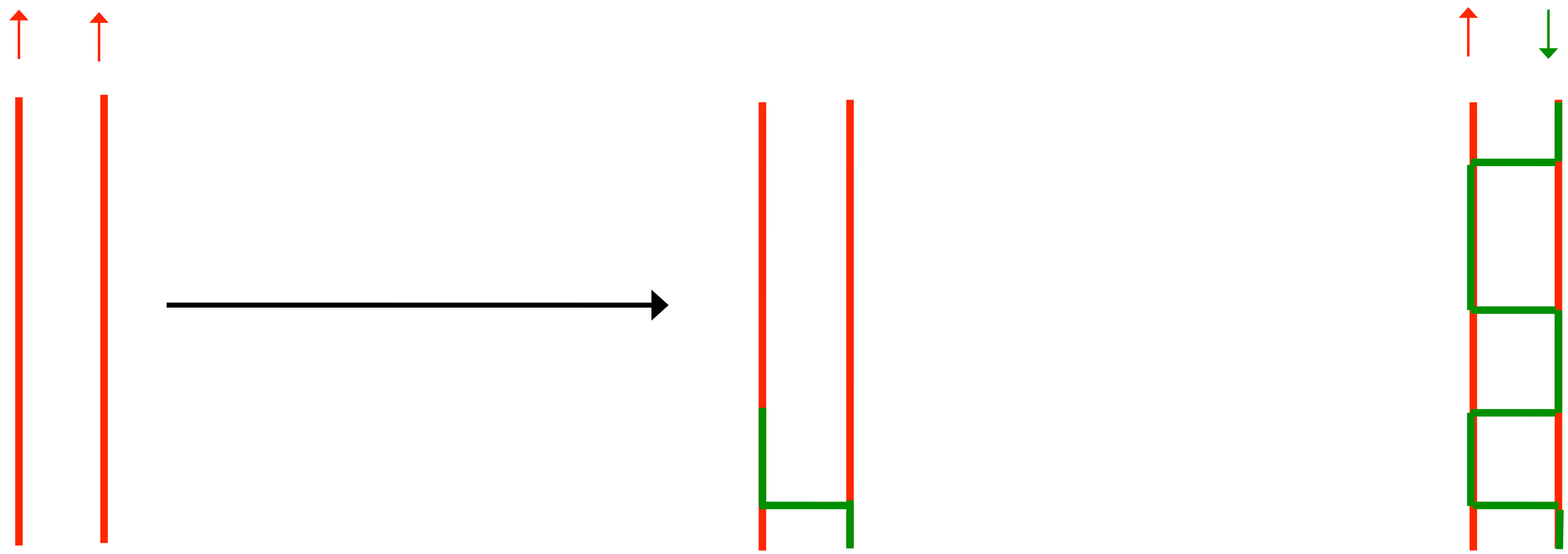
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

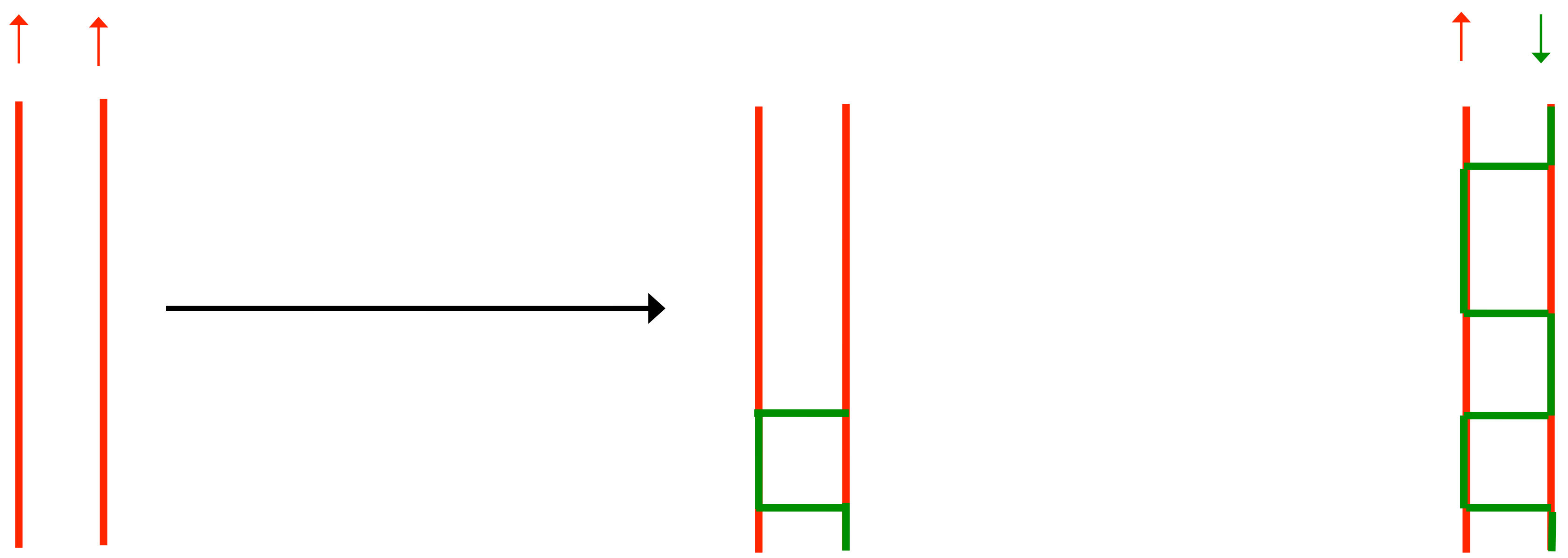
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

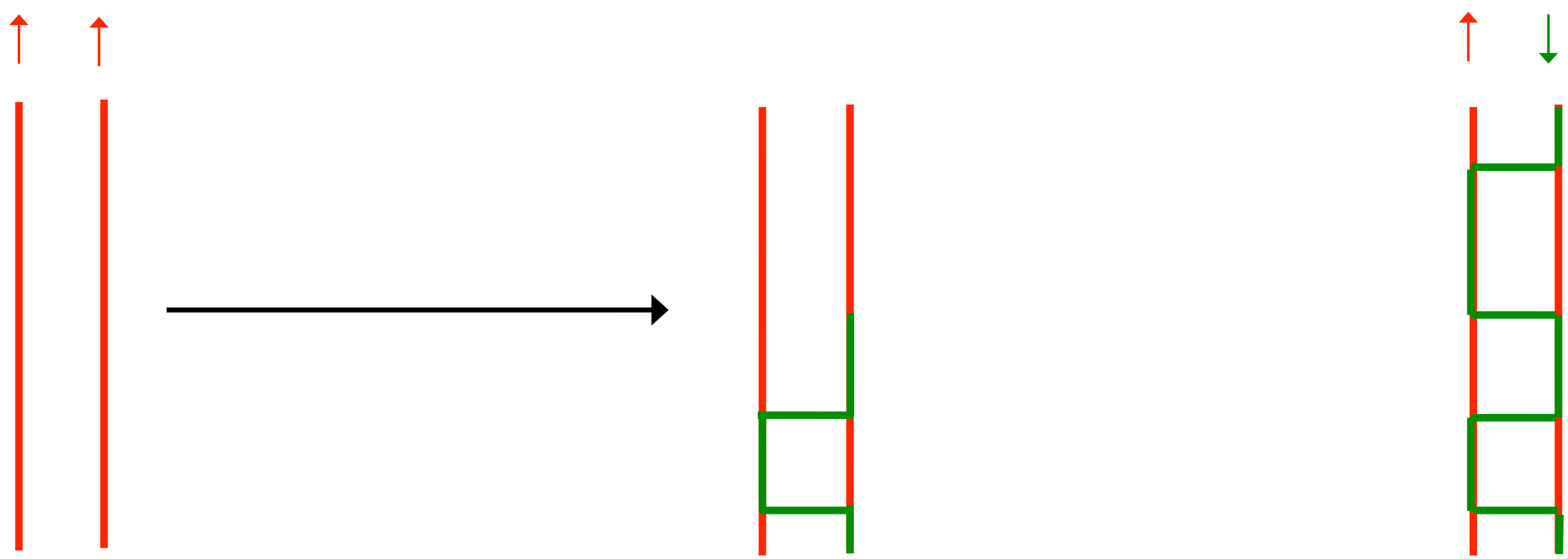
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

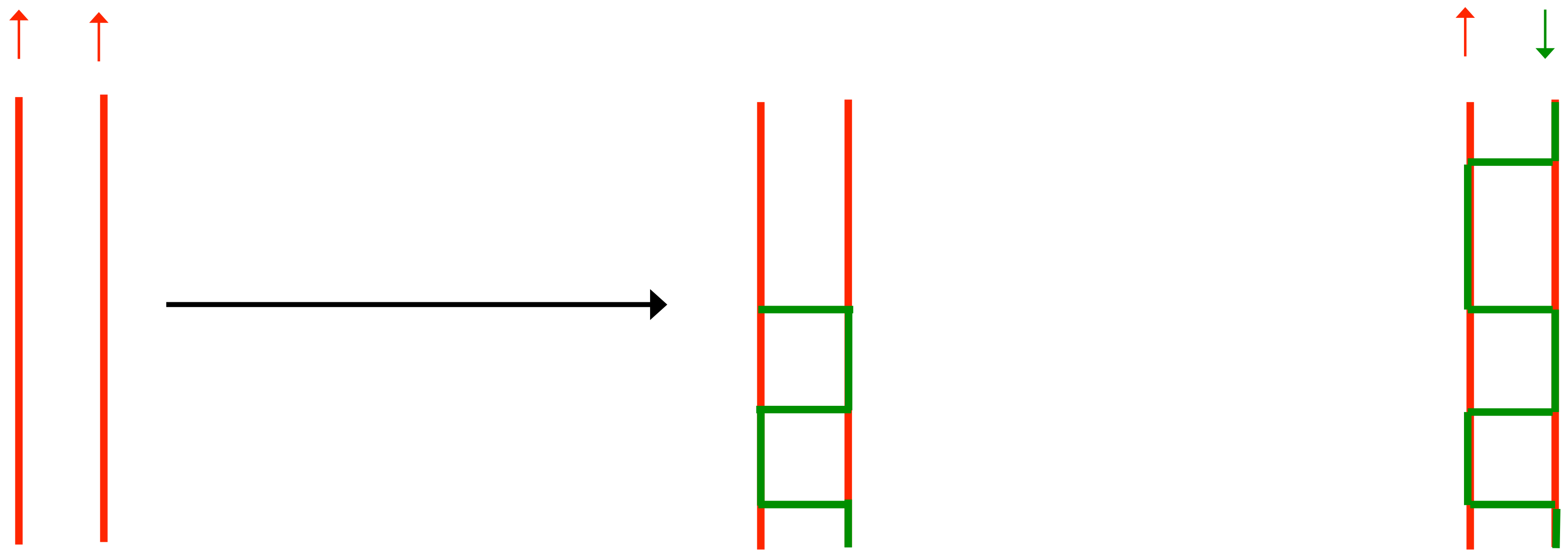
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

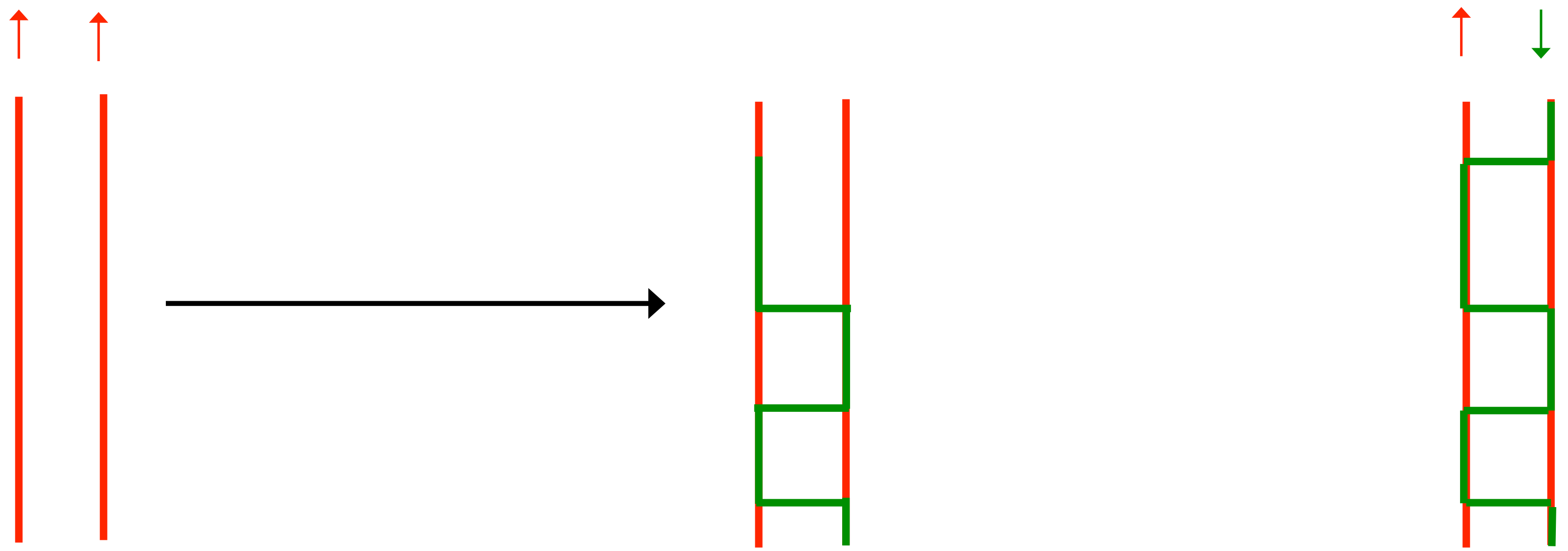
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

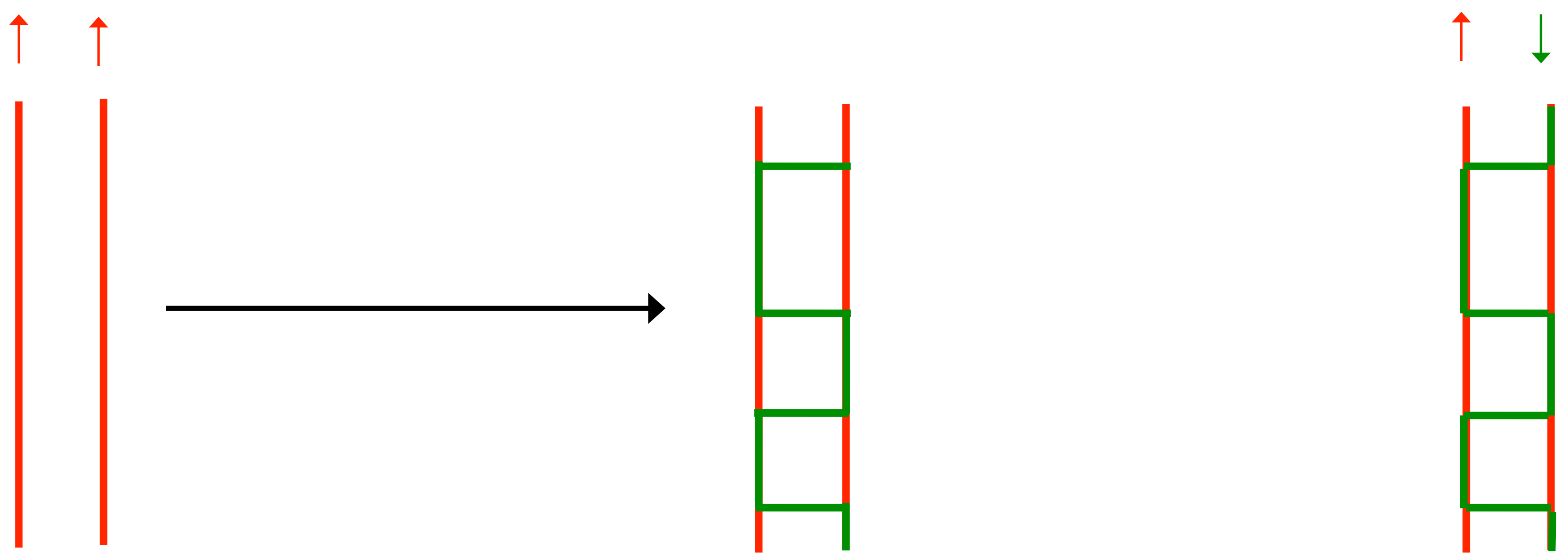
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

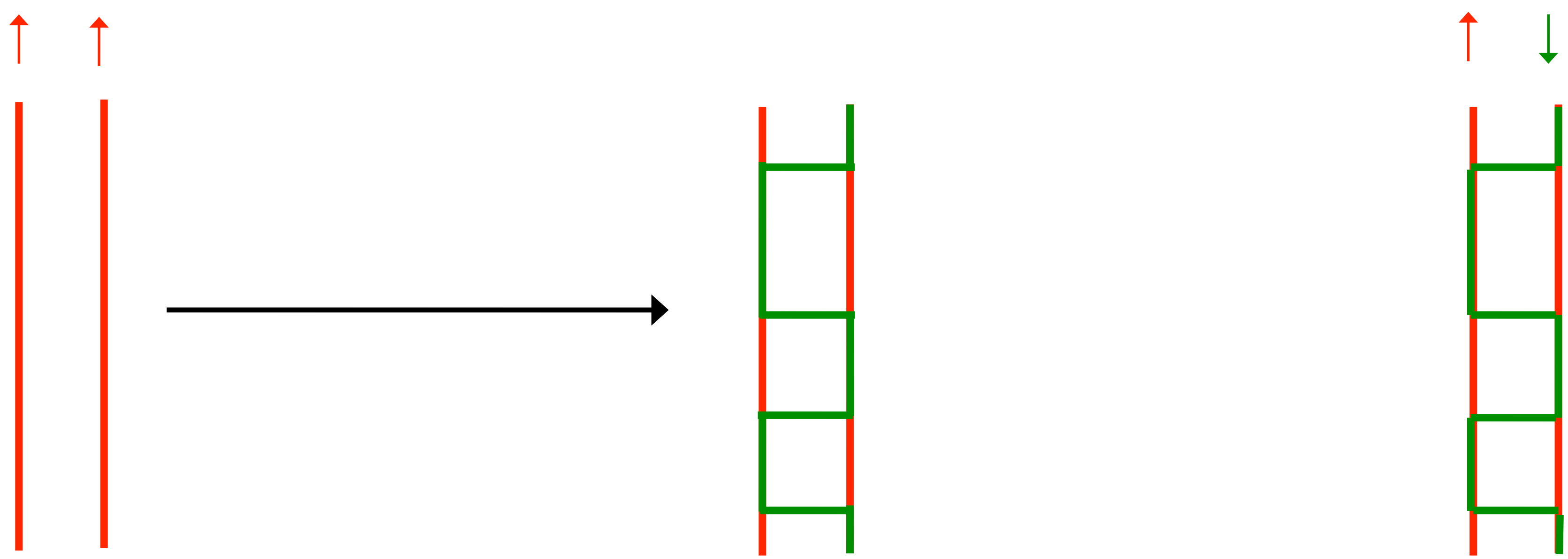
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

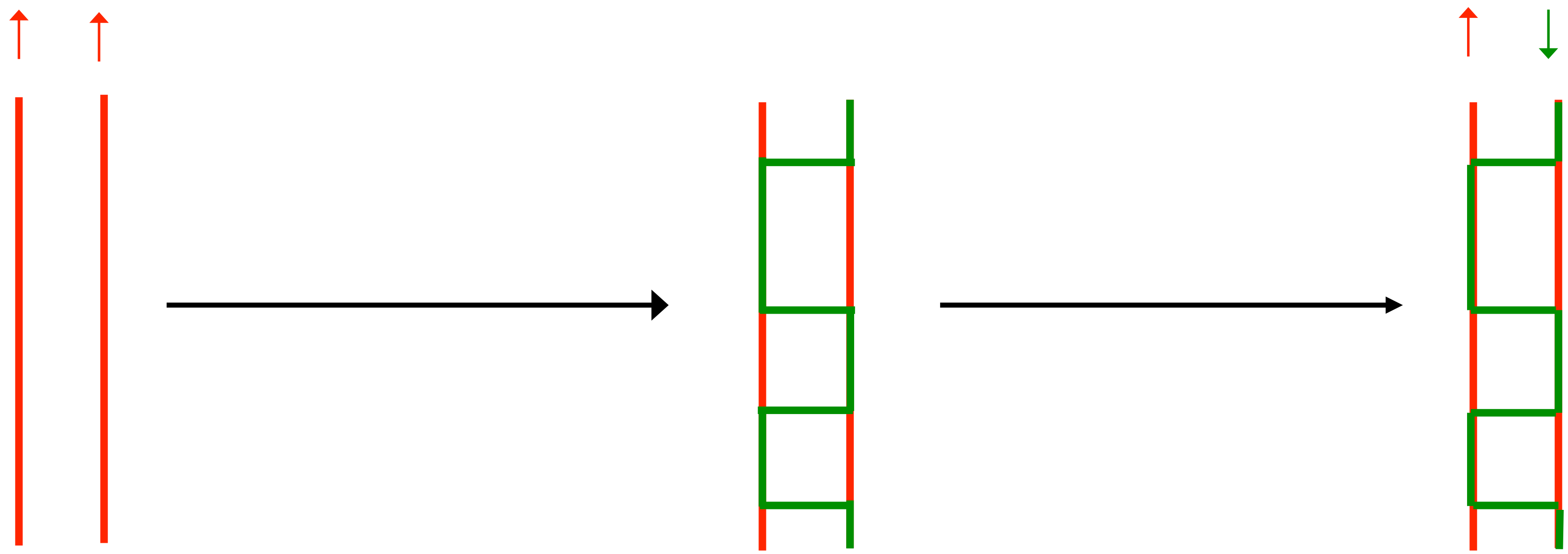
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

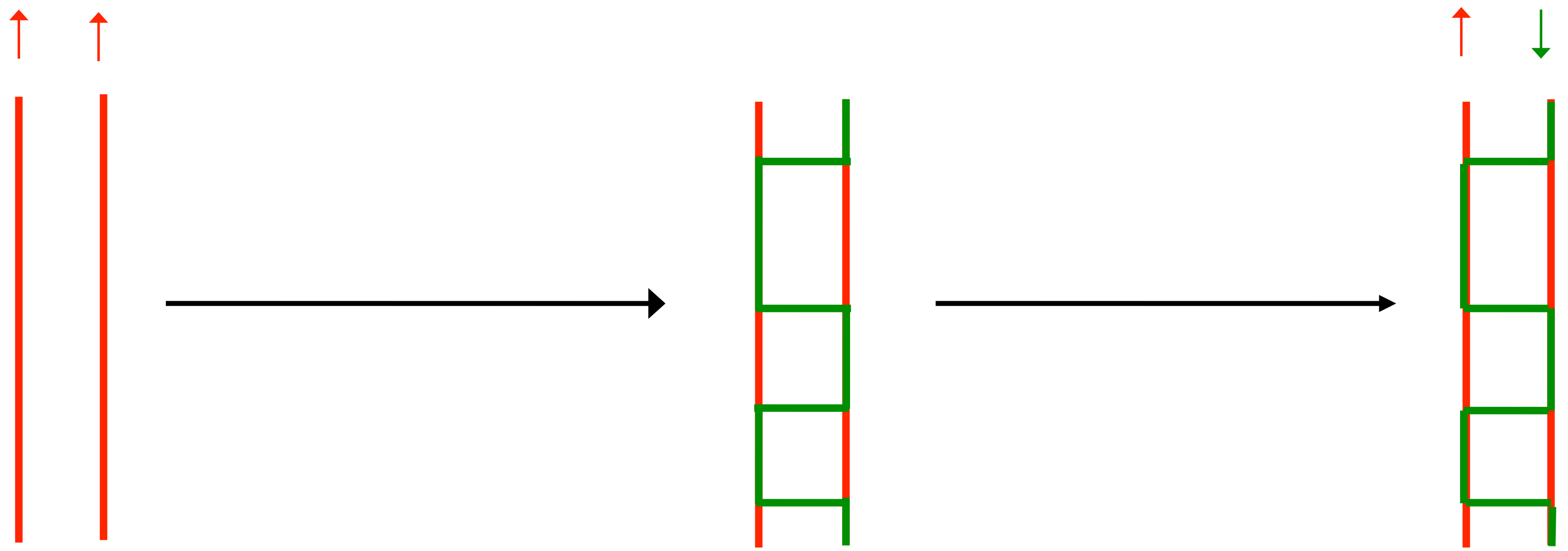
$E = J/4 - h = -3/4$

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1



Triplet

$E = J/4 - h = -3/4$

No high energy intermediate state

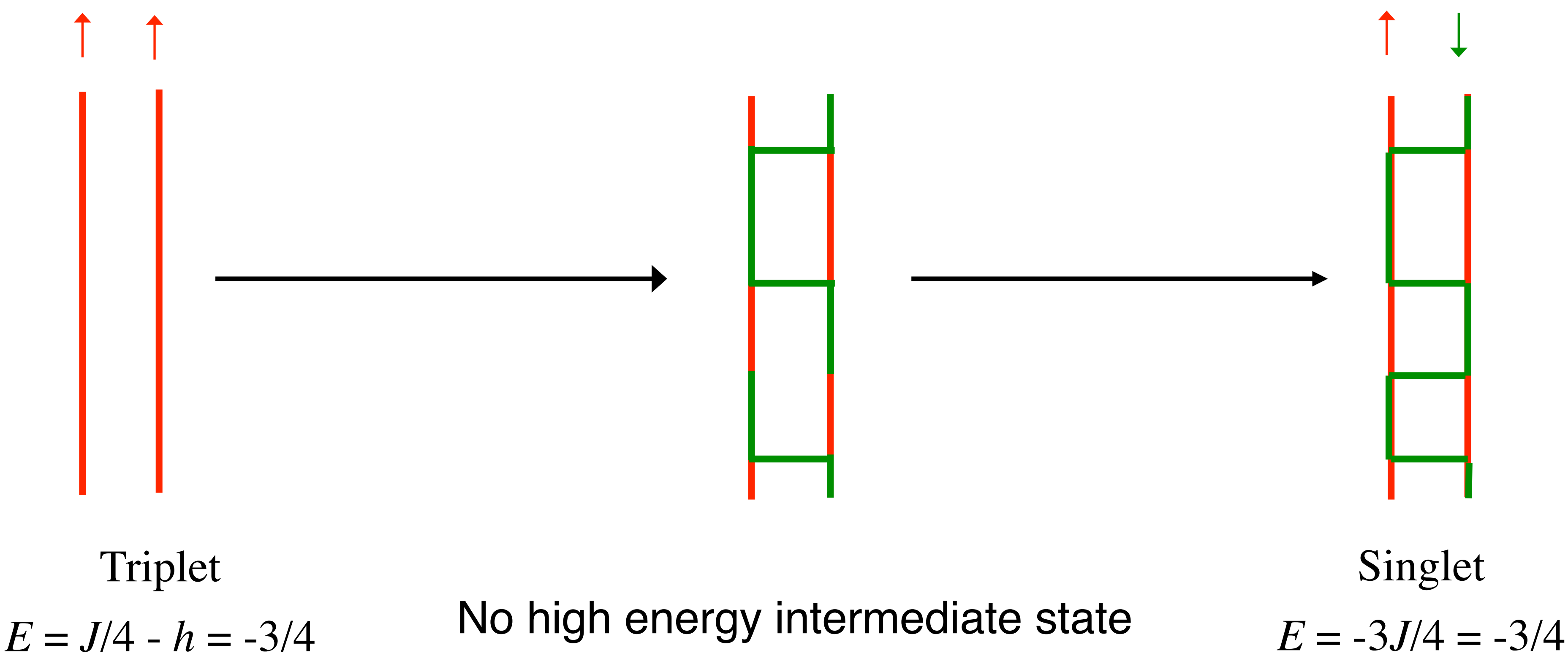
*Efficient update in presence of a magnetic field*

Singlet

$E = -3J/4 = -3/4$

# Worm algorithm in  a magnetic field

- Worm algorithm performs a random walk
  - Change of configuration done in small steps
- Example: spin dimer at J = h =1




No high energy intermediate state

*Efficient update in presence of a magnetic field*

# An earlier attempt

PHYSICAL REVIEW B VOLUME 27, NUMBER 1 1 JANUARY 1983

**Monte Carlo studies of one-dimensional quantum Heisenberg and *XY* models**

John J. Cullen and D. P. Landau
*Department of Physics and Astronomy, University of Georgia, Athens, Georgia 30602*
(Received 20 August 1982)

- Prokof'ev *et al* '98
  - detailed balance at each step of random walk
- Cullen and Landau '83
  - unbiased random walk
  - less efficient since the physics does not enter worm construction

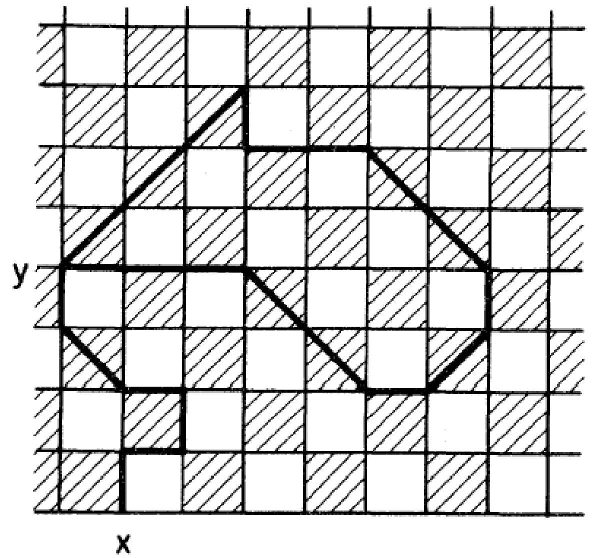



FIG. 5. String of spins generated until it intersects itself, the tail being discarded.



Saturday, January 21, 12

# 9. Overview of modern QMC algorithms

# Modern Monte Carlo algorithms



- Which system sizes can be studied?

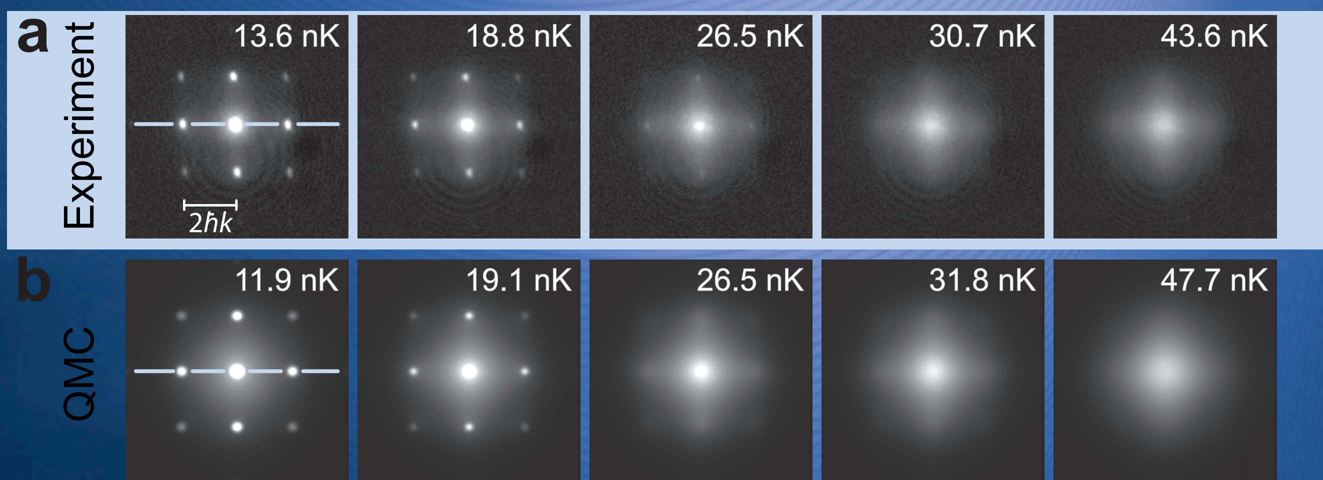
| temperature | local updates | modern algorithms |
|---|---|---|
| 3D Tc | 16’000 spins | 16’000’000 spins |
| 0.1 J | 200 spins | 1’000’000 spins |
| 0.005 J | — | 50’000 spins |
| 3D Tc | 32 bosons | 1’000’000 bosons |
| 0.1 t | 32 bosons | 10’000 bosons |

# When to use which algorithm?

- Stochastic Series Expansion (SSE) is simpler to implement
- Continuous-time path integrals needs lower orders
- Use SSE for local actions with not too large diagonal terms

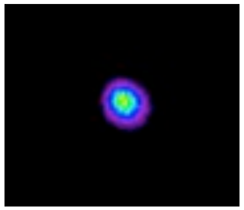
| | SSE | Path Integrals |
|---|---|---|
| **Loop algorithm** | Spin models | Spin models with dissipation |
| **Worm algorithm/ Directed loops** | Spin models in magnetic field | Bose-Hubbard models |

# 10. Simulating optical lattice experiments



# Feynman's quantum simulator

- We are able to control single quantum systems

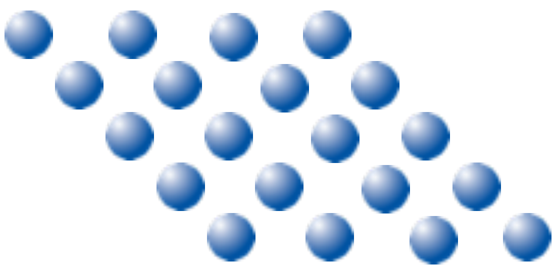


Single Atoms and Ions

Photons

Quantum Dots

- New challenges:
control, engineer and understand complex quantum system



**R. P. Feynman's Vision**

A Quantum Simulator to study the quantum dynamics of another system.

**R.P. Feyman, Int. J. Theo. Phys. (1982)**
**R.P. Feynman, Found. Phys (1986)**

# Quantum simulators

**Strongly correlated materials:**
strong correlation effects in many-electron systems

↑

**Condensed matter models:**
Simple models which capture the relevant mechanism

# Quantum simulators

**Strongly correlated materials:**
strong correlation effects in many-electron systems

no exact solutions
approximations,
impurities,
...

**Condensed matter models:**
Simple models which capture the relevant mechanism

# Quantum simulators

**Strongly correlated materials:**
strong correlation effects in many-electron systems

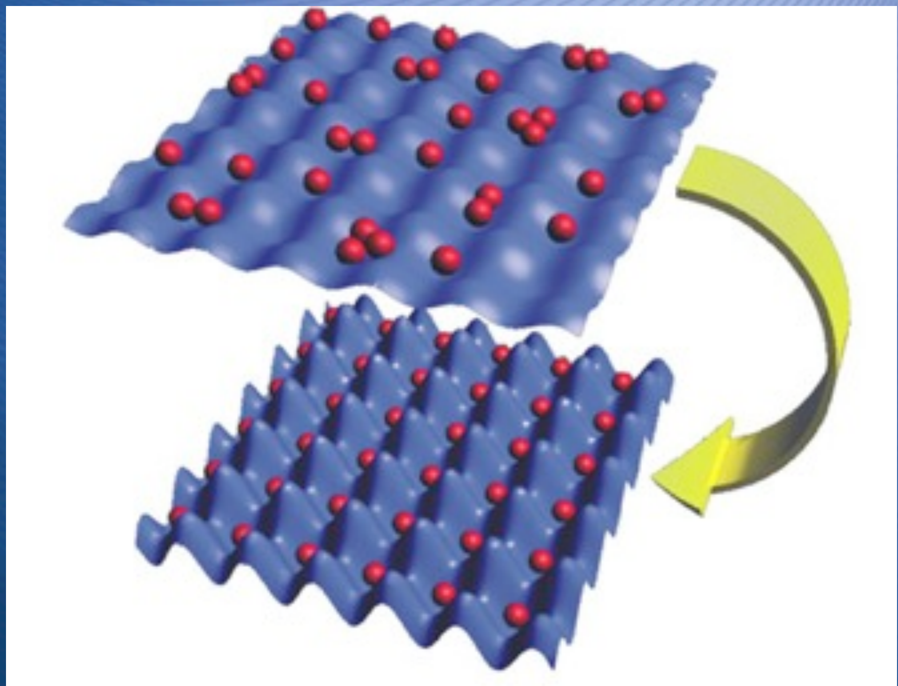
↕

**Quantum simulators:**
Controlled, “simple” systems testing models and verifying concepts

↕

**Condensed matter models:**
Simple models which capture the relevant mechanism

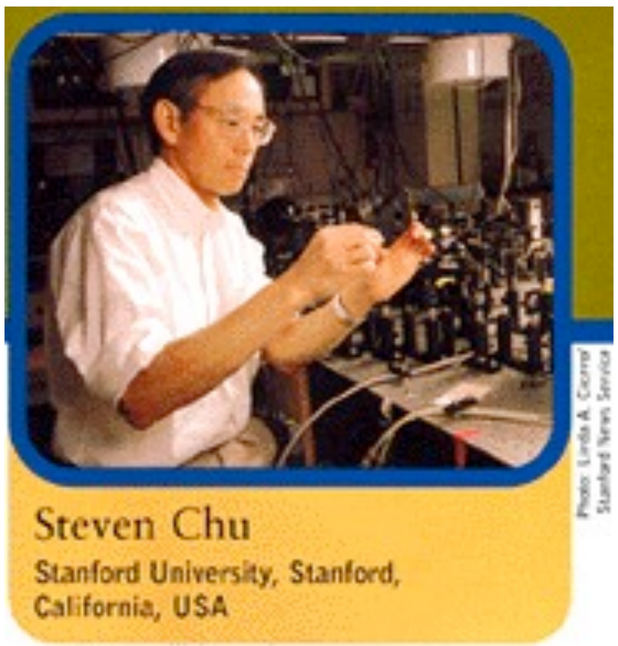
# Ultracold atomic gases as quantum simulators

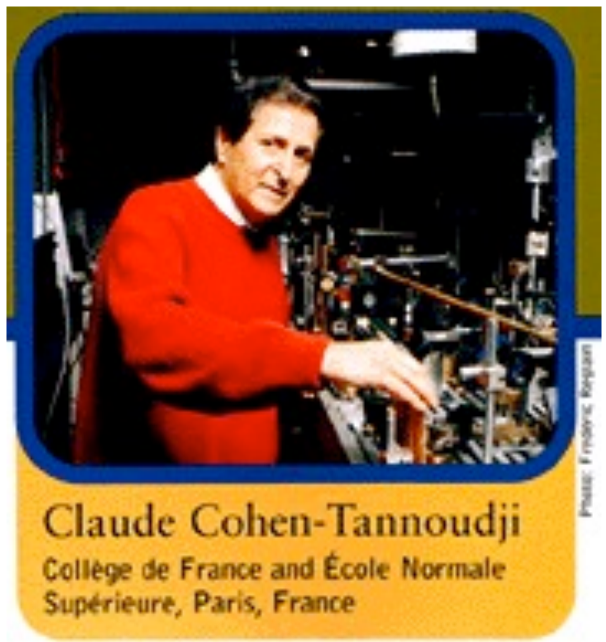


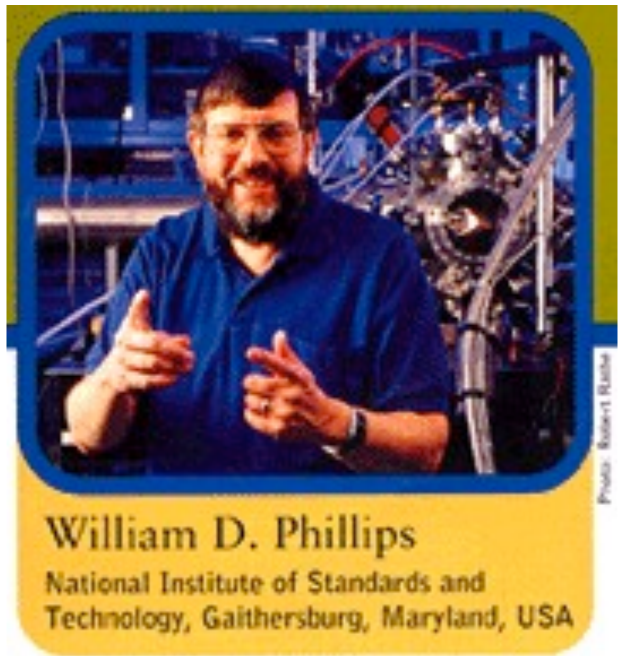
# 1997 Nobel Prize in Physics

**Steven Chu,Claude Cohen-Tannoudji and William D.Phillips**

share 1997 Nobel Prize for the development of methods to cool and trap atoms with laser right.



# 2001 Nobel Prize in Physics

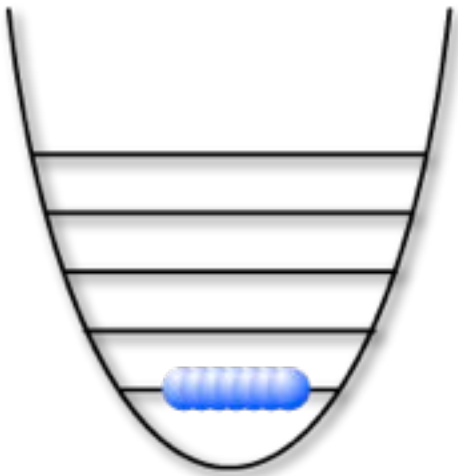
**Carl Wieman , Eric Cornell and Wolfgang Ketterle**

share 2001 Nobel Prize for the achievement of BEC in dilute gases of alkali atoms and for the early fundamental studies of the properties of the condensates .

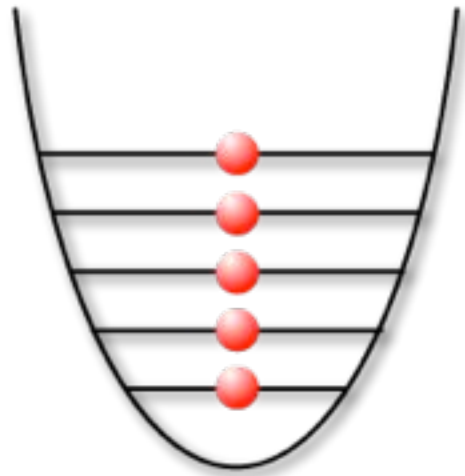
# Our Starting Point – Ultracold Quantum Gases

Parameters:
Densities: $10^{15}$ $cm^{-3}$
Temperatures: Nano Kelvin
Atom Numbers $10^6$

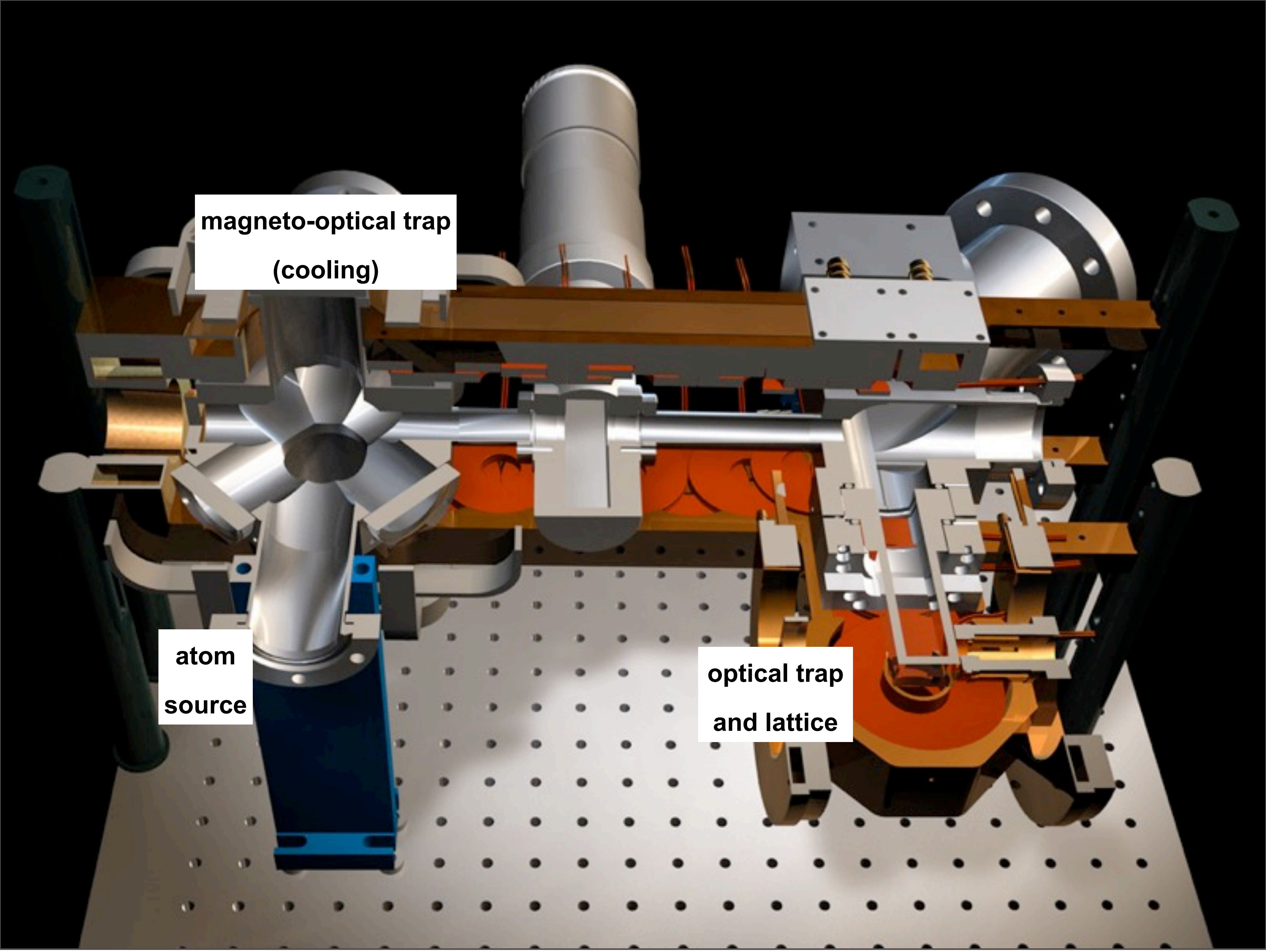
Ground States at T=0



Bose-Einstein
Condensates e.g. $^{87}Rb$

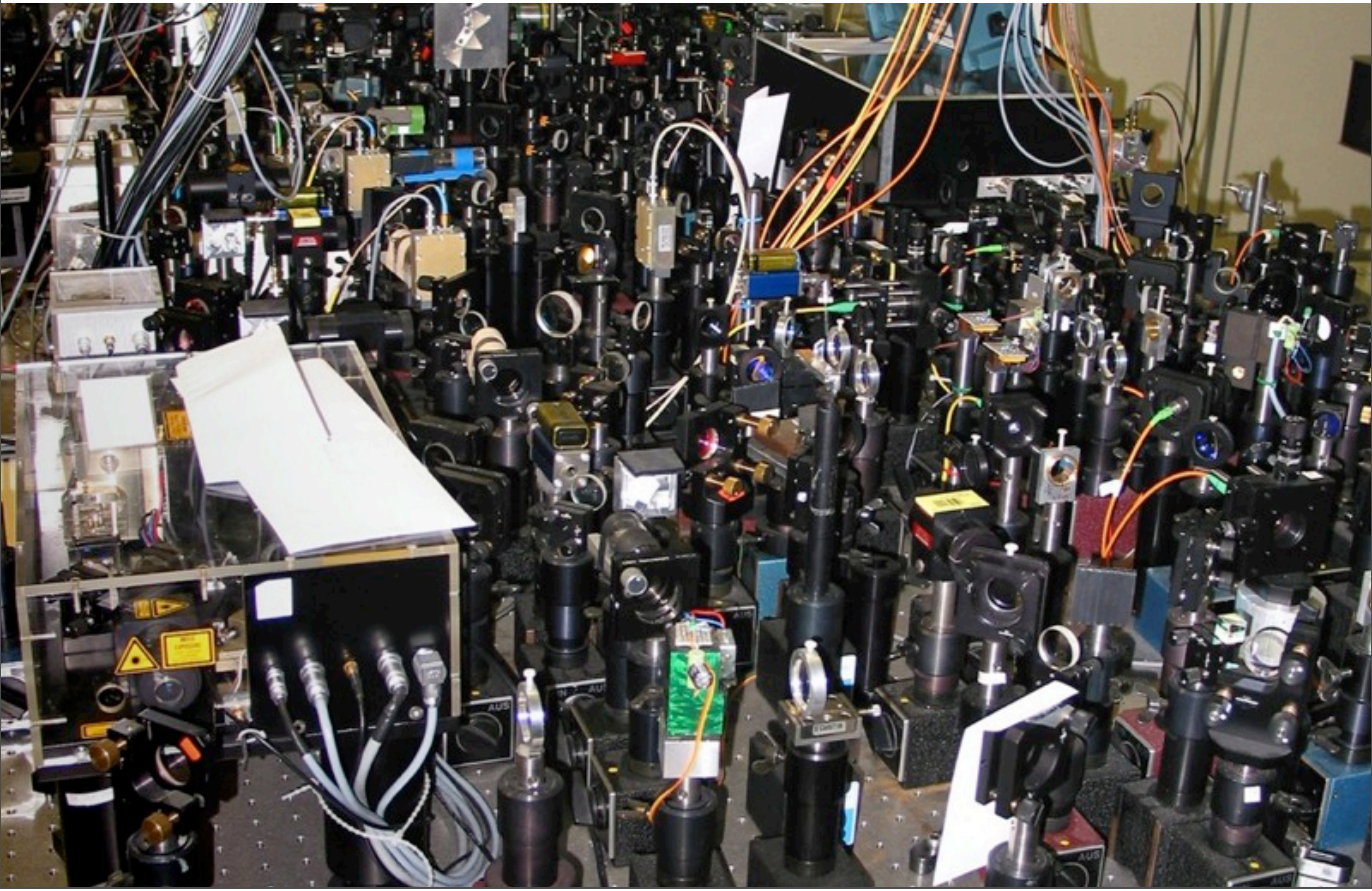


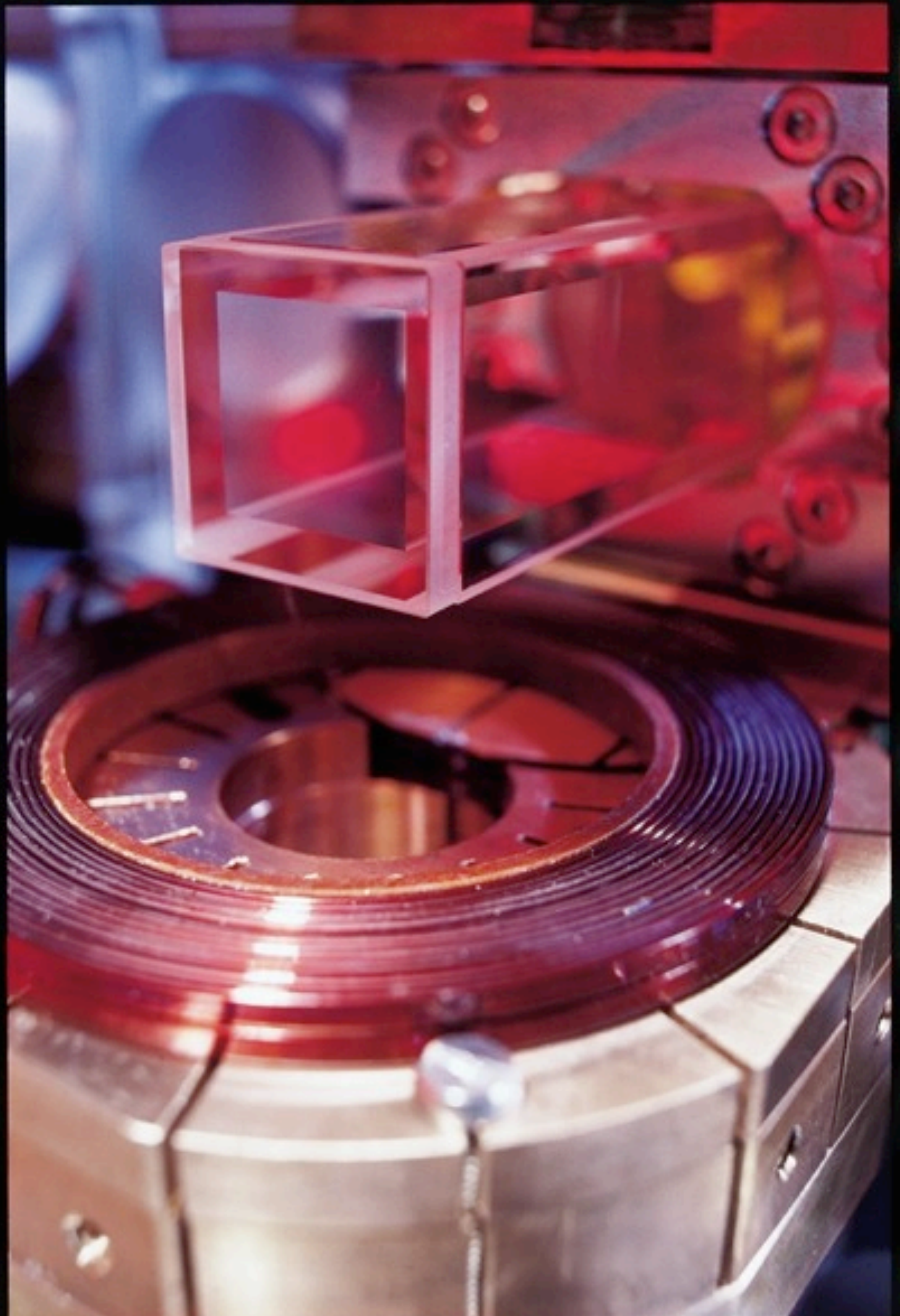
Degenerate Fermi Gases
e.g. $^{40}K$



magneto-optical trap
(cooling)
atom
source
optical trap
and lattice

# And a lot of optics and electronics !





# How do we detect these quantum gases ?

# How do we detect these quantum gases ?

release the atoms

## How do we detect these quantum gases ?

release the atoms

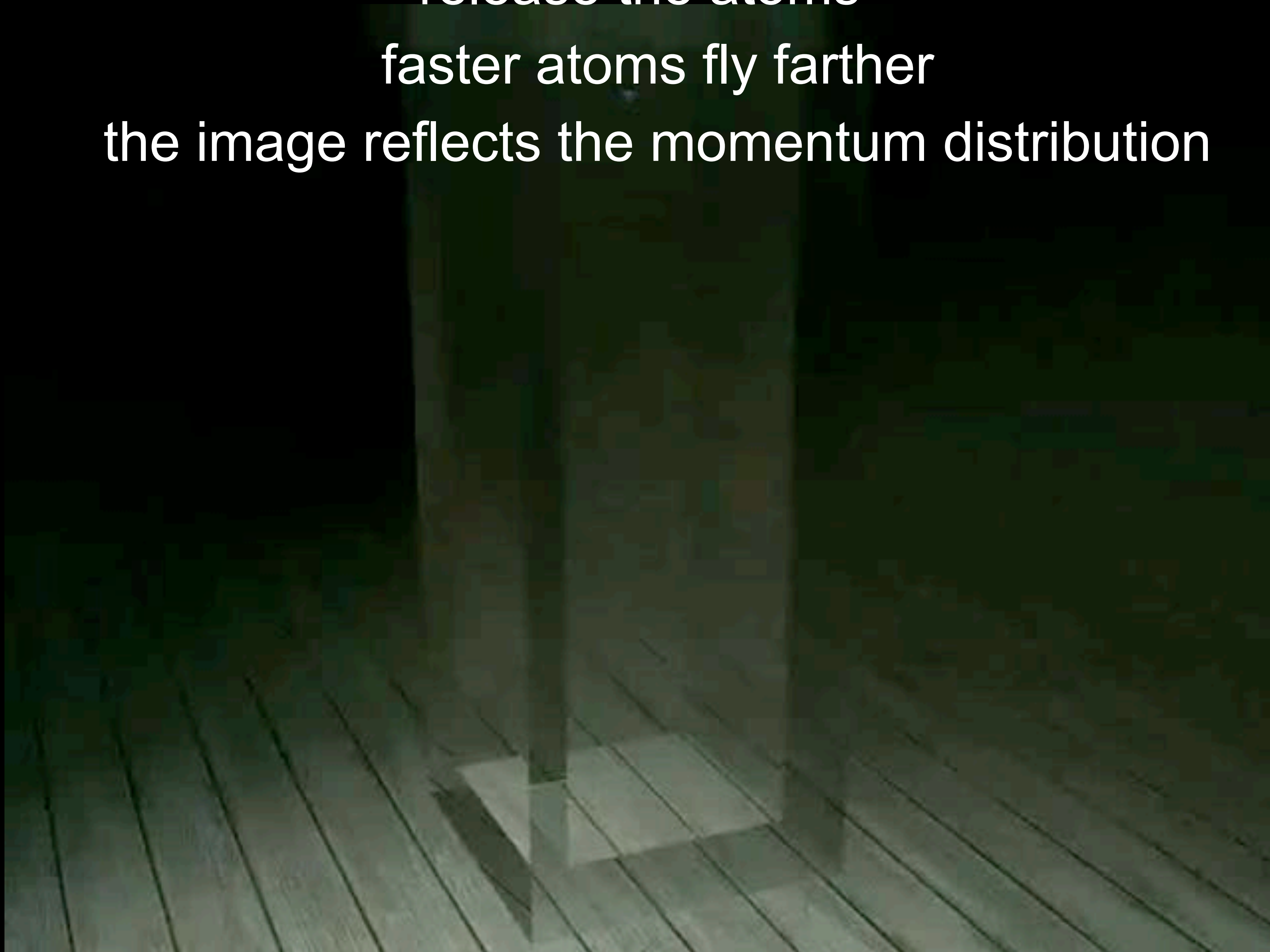
faster atoms fly farther

## How do we detect these quantum gases ?

release the atoms

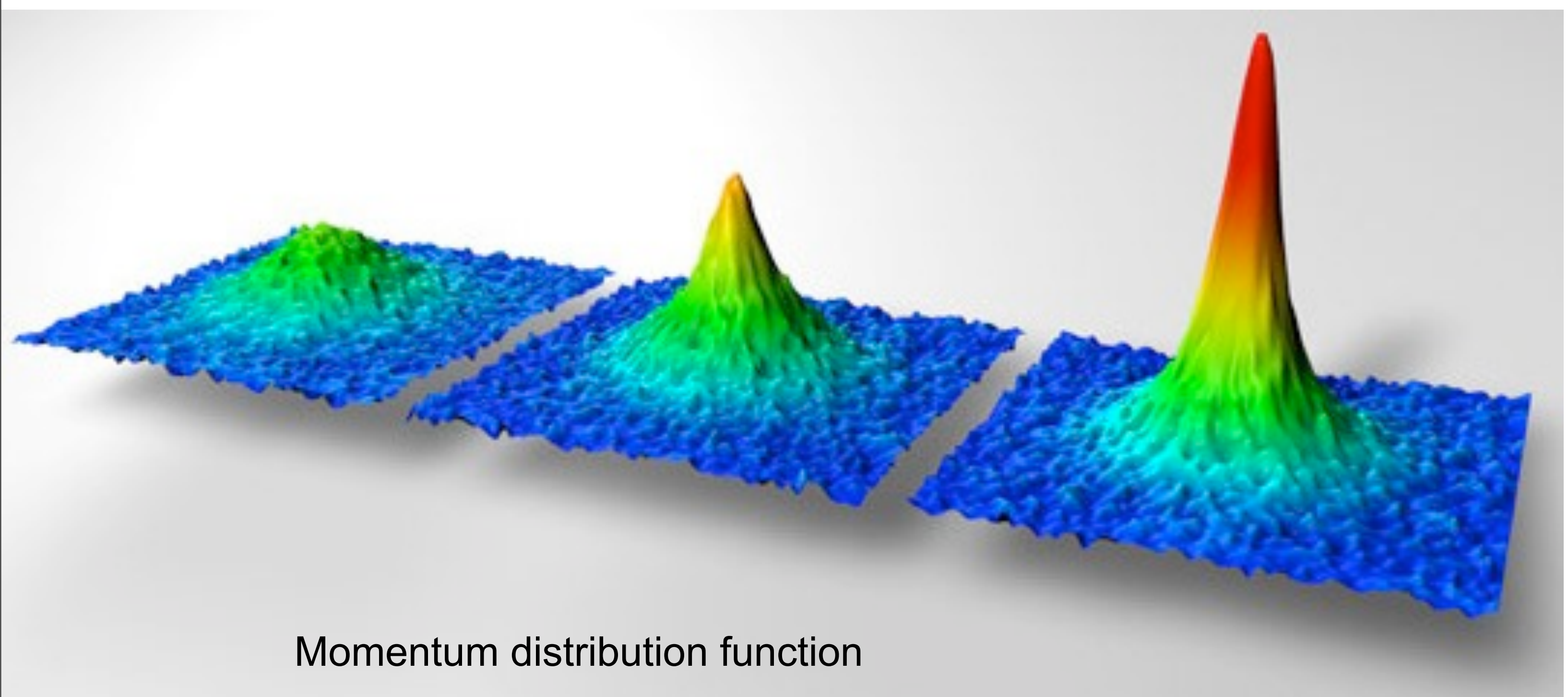
faster atoms fly farther

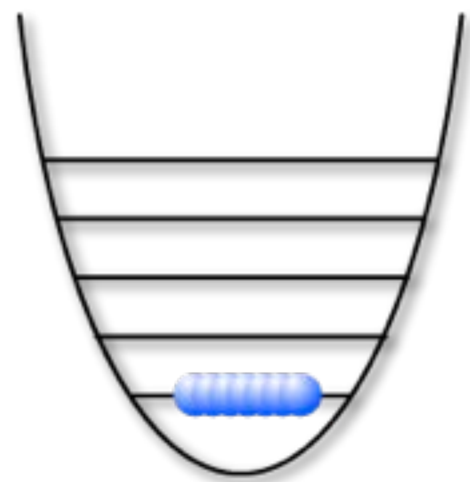
the image reflects the momentum distribution



# Bose-Einstein condensation in cold atomic gases

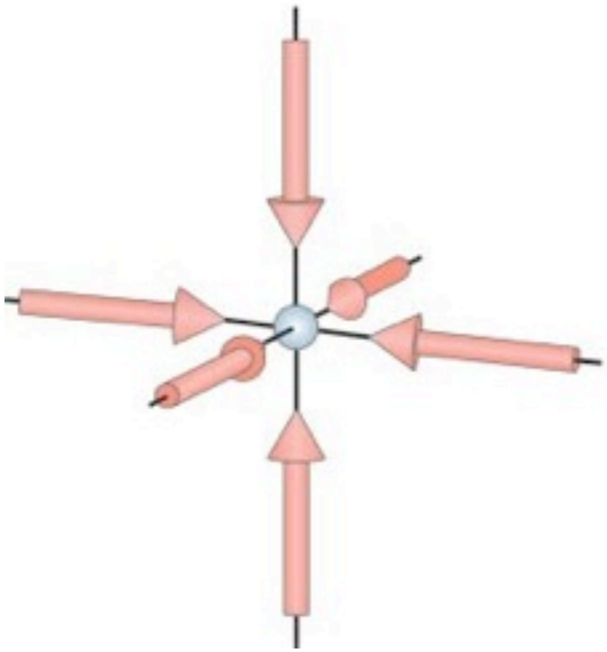
- At close to zero temperatures, a macroscopic fraction of all atoms in a Bose gas occupy the same quantum state
- A diverging occupation of the zero momentum state

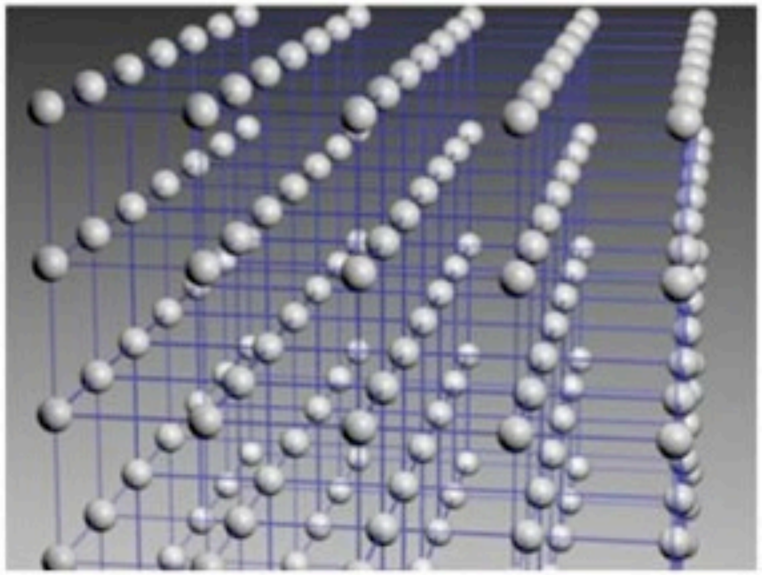


# Optical lattices

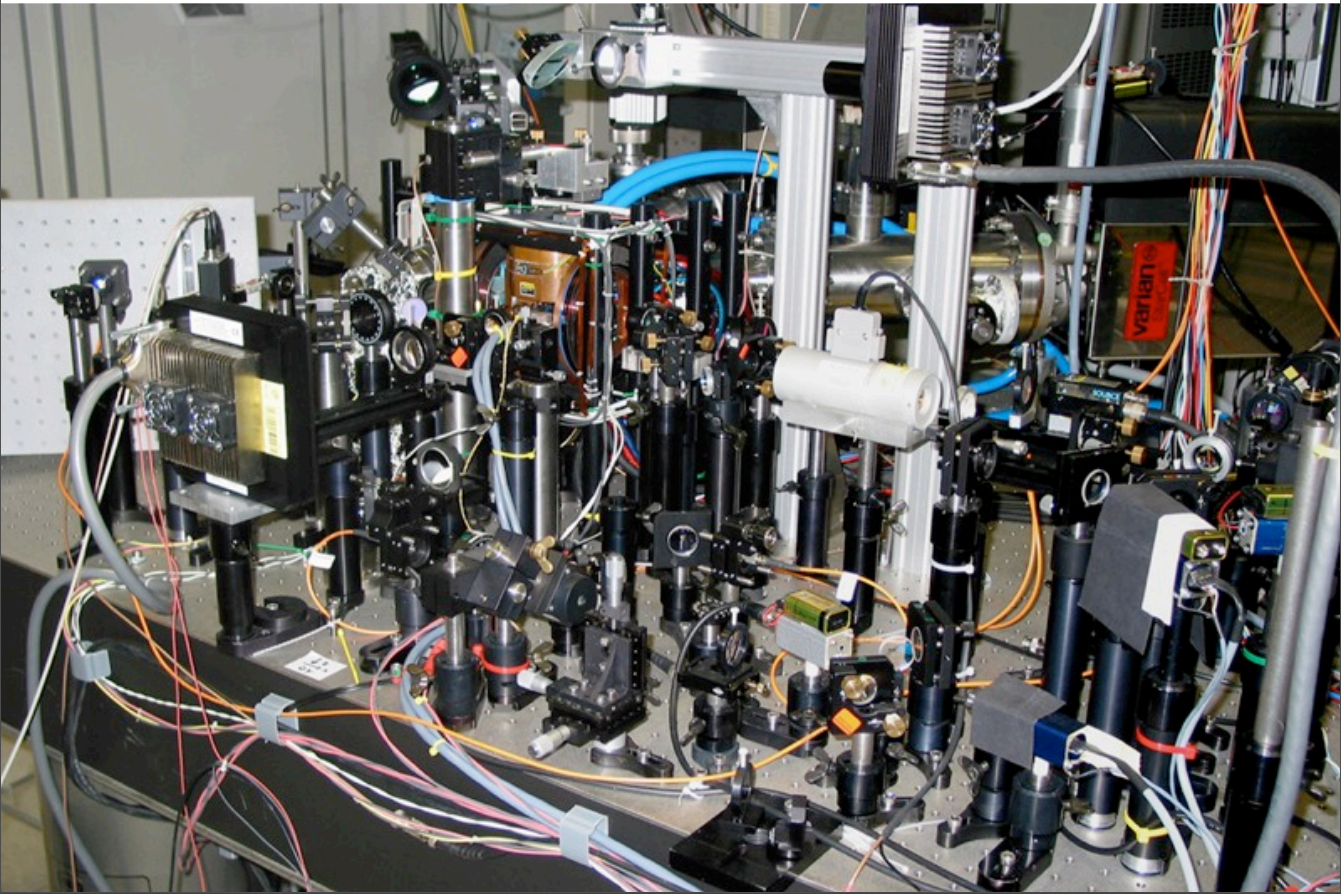
- formed by standing waves  from three pairs of laser beams



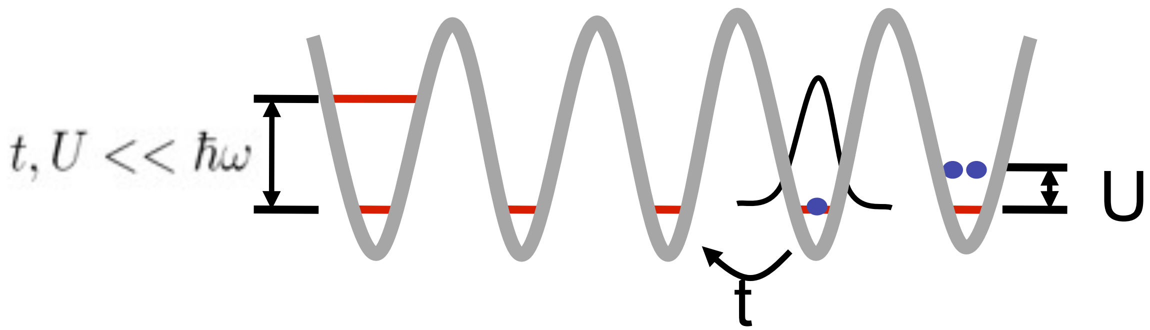


- realize quantum **lattice** models of fermions or bosons

# Table 2

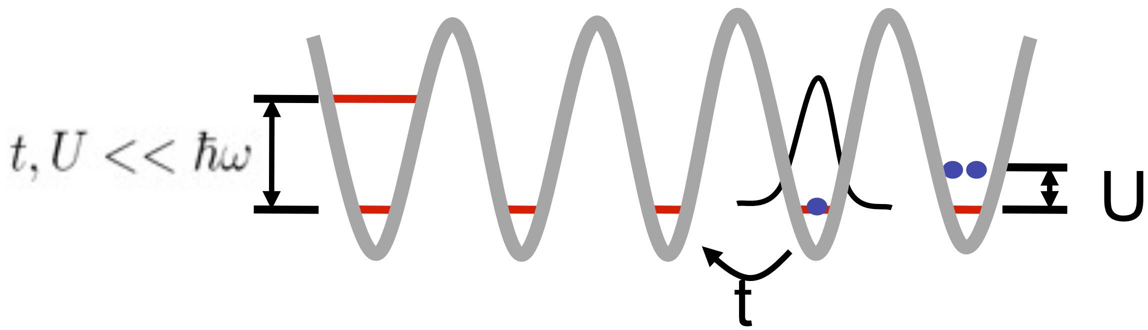


# Optical lattices and the Hubbard model



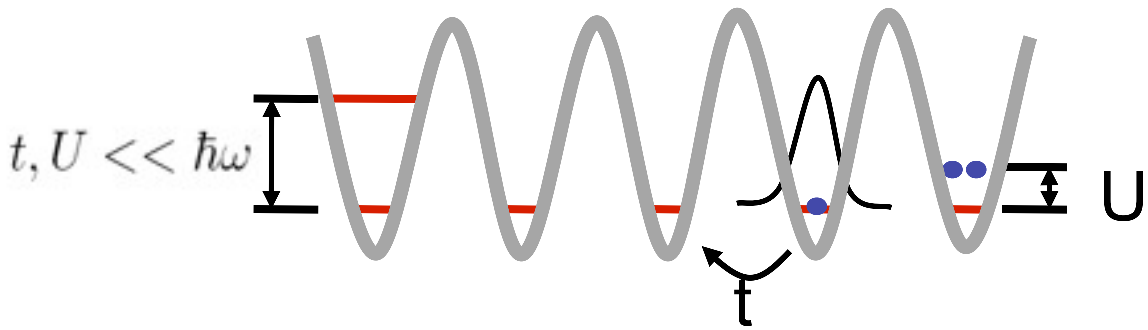
# Optical lattices and the Hubbard model

- Lasers couple to the dipole moment of the atoms
  - atoms prefer to sit at the amplitude maxima (AC Stark effect)
  - a periodic potential with periodicity half of the wave length
  - obtain a Hubbard model for the lowest band



# Optical lattices and the Hubbard model

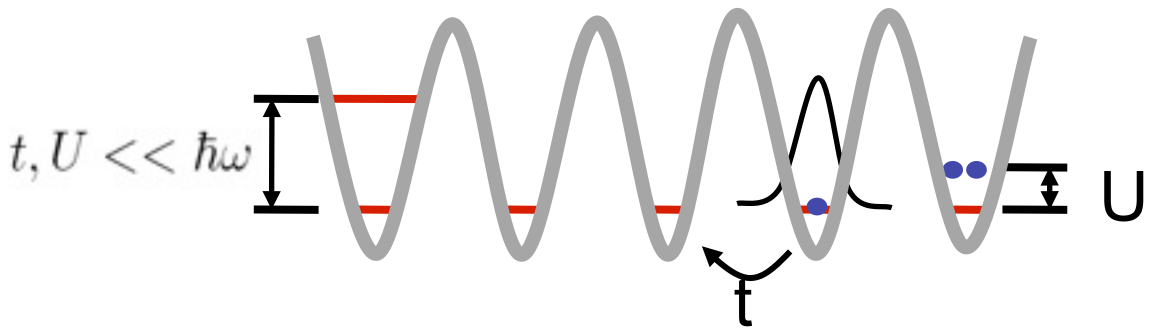
- Lasers couple to the dipole moment of the atoms
  - atoms prefer to sit at the amplitude maxima (AC Stark effect)
  - a periodic potential with periodicity half of the wave length
  - obtain a Hubbard model for the lowest band




- Tunable and controlled
  - Laser amplitude determines *U* and *t*
  - Spatially varying couplings using optical superlattices

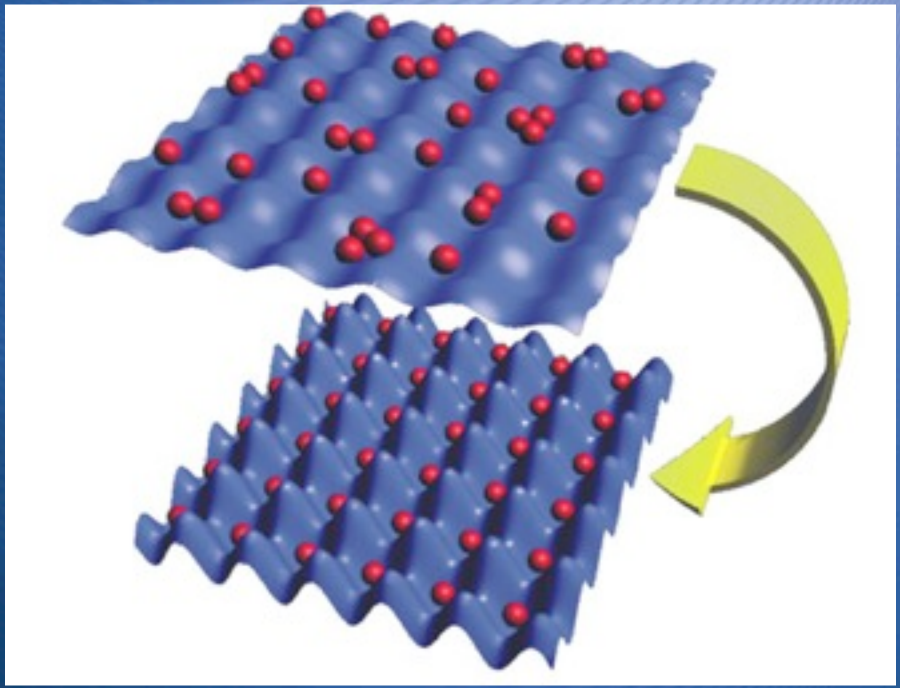
# Optical lattices and the Hubbard model

- Lasers couple to the dipole moment of the atoms
  - atoms prefer to sit at the amplitude maxima (AC Stark effect)
  - a periodic potential with periodicity half of the wave length
  - obtain a Hubbard model for the lowest band




- Tunable and controlled
  - Laser amplitude determines *U* and *t*
  - Spatially  varying couplings using optical superlattices
- Flexible
  - fermionic or bosonic atoms or mixtures are possible

# Validating a quantum simulator: does it really work?

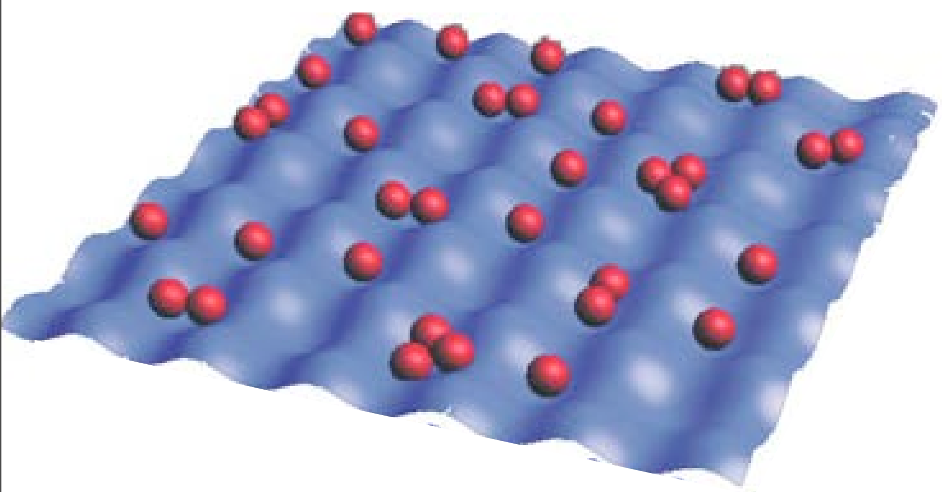


# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\text{opt}}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r})\psi^\dagger(\vec{r})\psi(\vec{r})\psi(\vec{r})$$

$$V_{\text{opt}}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$





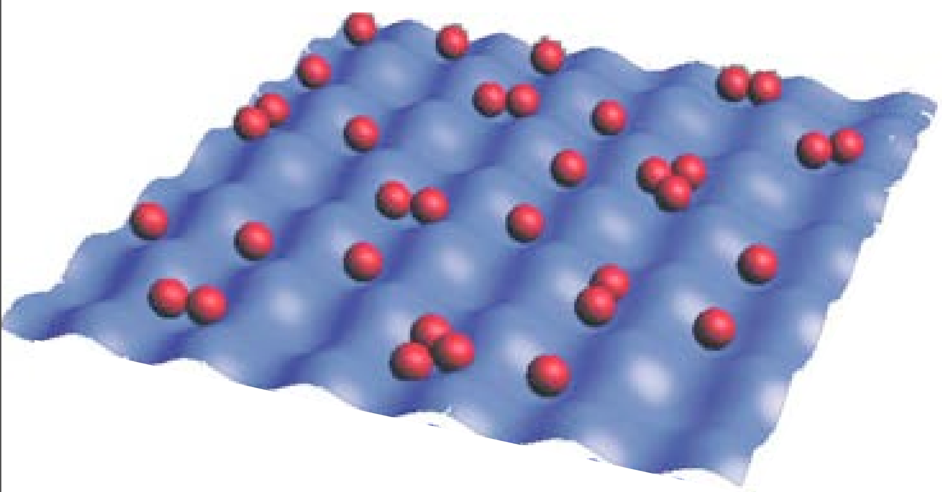
Saturday, January 21, 12

# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\text{opt}}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r})\psi^\dagger(\vec{r})\psi(\vec{r})\psi(\vec{r})$$

$$V_{\text{opt}}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$

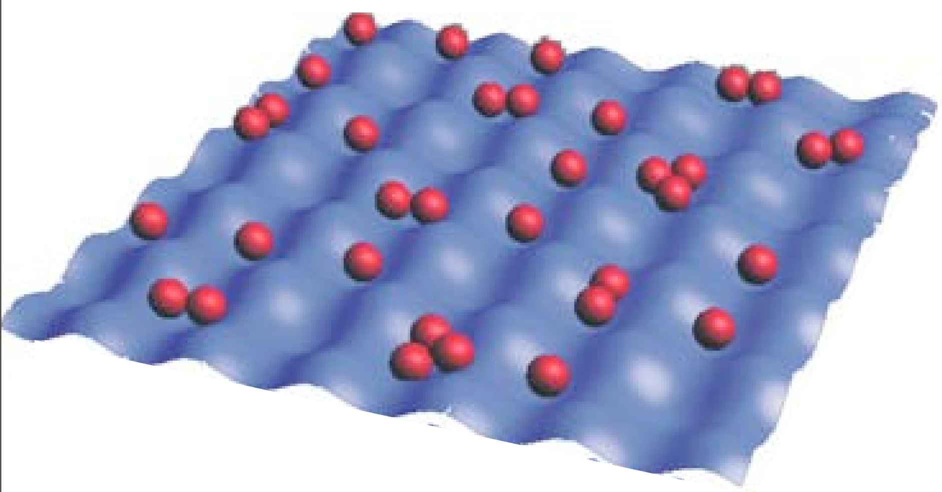


# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\text{opt}}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r}) \psi^\dagger(\vec{r}) \psi(\vec{r}) \psi(\vec{r})$$

$$V_{\text{opt}}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$

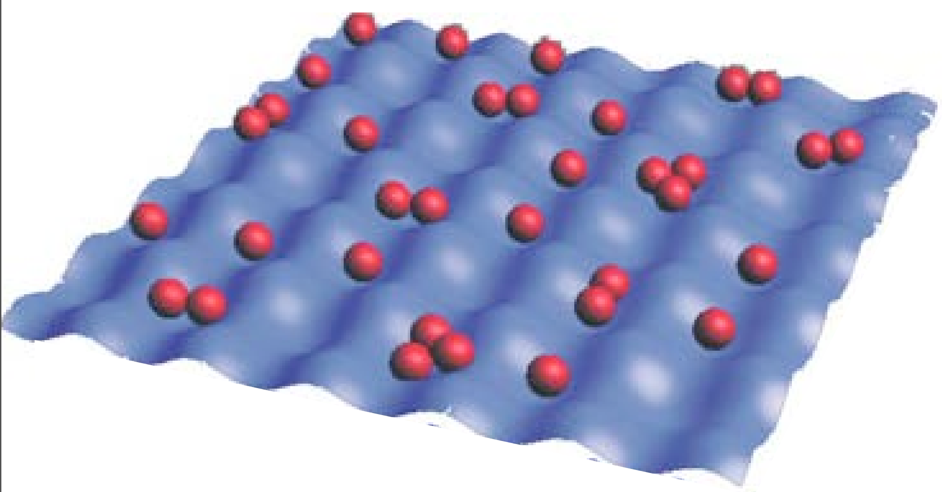


# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\rm opt}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r})\psi^\dagger(\vec{r})\psi(\vec{r})\psi(\vec{r})$$

$$V_{\rm opt}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$

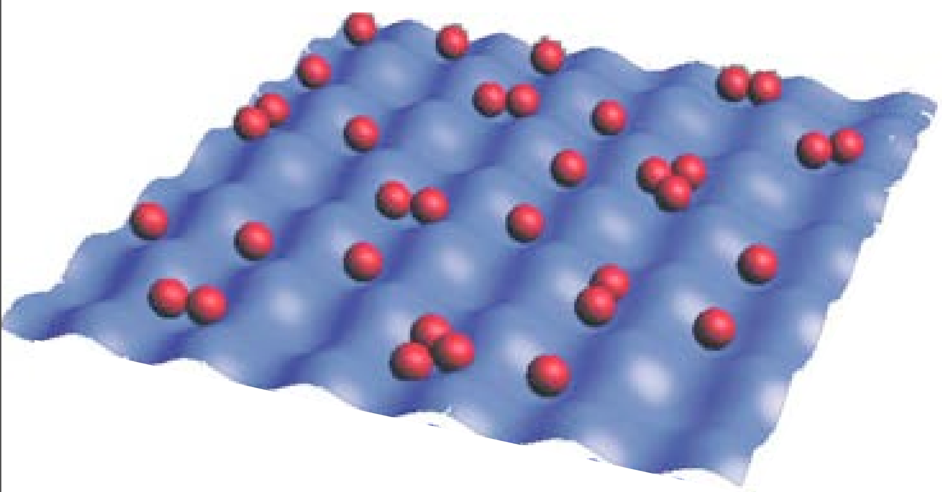


# The ab-initio microscopic model

$$H = \int d^3 r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\rm opt}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3 r \psi^\dagger(\vec{r}) \psi^\dagger(\vec{r}) \psi(\vec{r}) \psi(\vec{r})$$

$$V_{\rm opt}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$

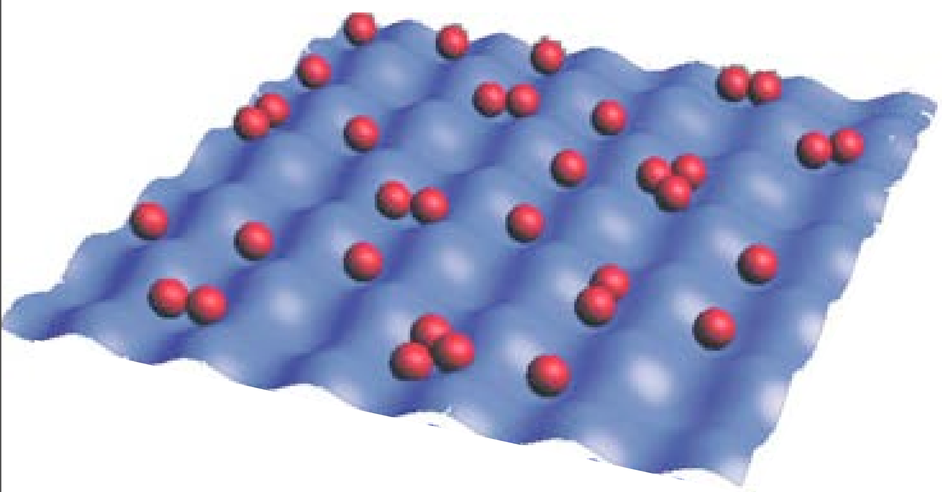


# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\rm opt}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r}) \psi^\dagger(\vec{r}) \psi(\vec{r}) \psi(\vec{r})$$

$$V_{\rm opt}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$

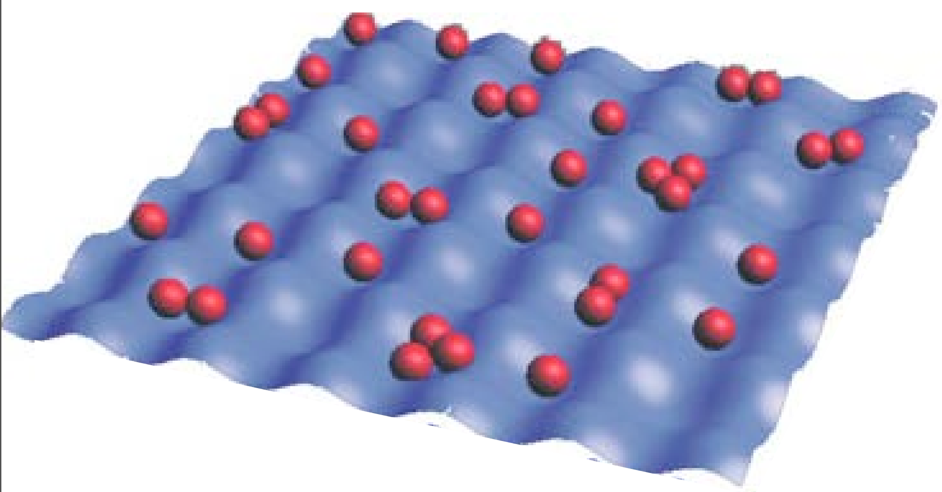


# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\text{opt}}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r})\psi^\dagger(\vec{r})\psi(\vec{r})\psi(\vec{r})$$

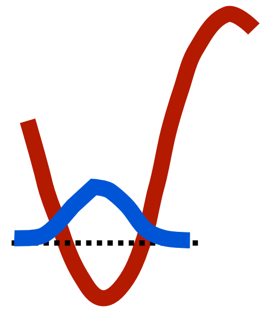
$$V_{\text{opt}}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$



$$\psi(\vec{r}) = \sum_i w(\vec{r} - \vec{r}_i) b_i$$

express the bosonic field operator in terms of Wannier functions

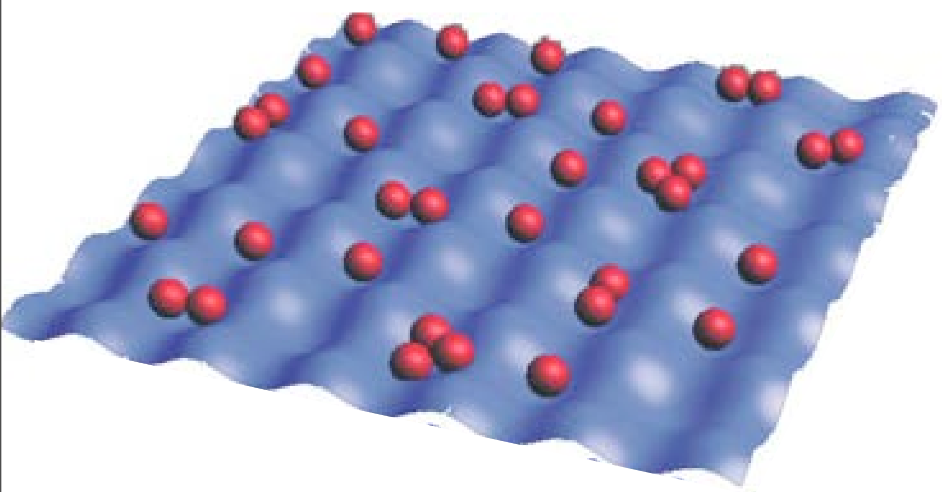


# The ab-initio microscopic model

$$H = \int d^3r \psi^\dagger(\vec{r}) \left( -\frac{\hbar^2}{2m}\Delta + V_{\text{opt}}(\vec{r}) \right) \psi(\vec{r}) + \frac{g}{2} \int d^3r \psi^\dagger(\vec{r})\psi^\dagger(\vec{r})\psi(\vec{r})\psi(\vec{r})$$

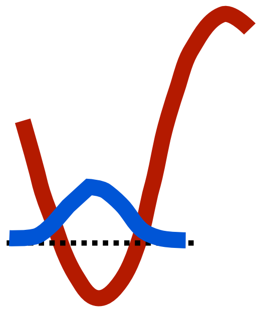
$$V_{\text{opt}}(r,z) = -V_0 e^{-2r^2/w^2(z)} \sin^2(kz)$$

$$g = \frac{4\pi\hbar^2 a_s}{m}$$



$$\psi(\vec{r}) = \sum_i w(\vec{r} - \vec{r}_i) b_i$$

express the bosonic field operator in terms of Wannier functions



$$H = -t \sum_{\langle ij \rangle} \left( b_i^\dagger b_j + \text{h.c.} \right) + U \sum_i n_i(n_i - 1)/2 - \mu \sum_i n_i + V \sum_i r_i^2 n_i$$

# Limitations of the Hubbard model

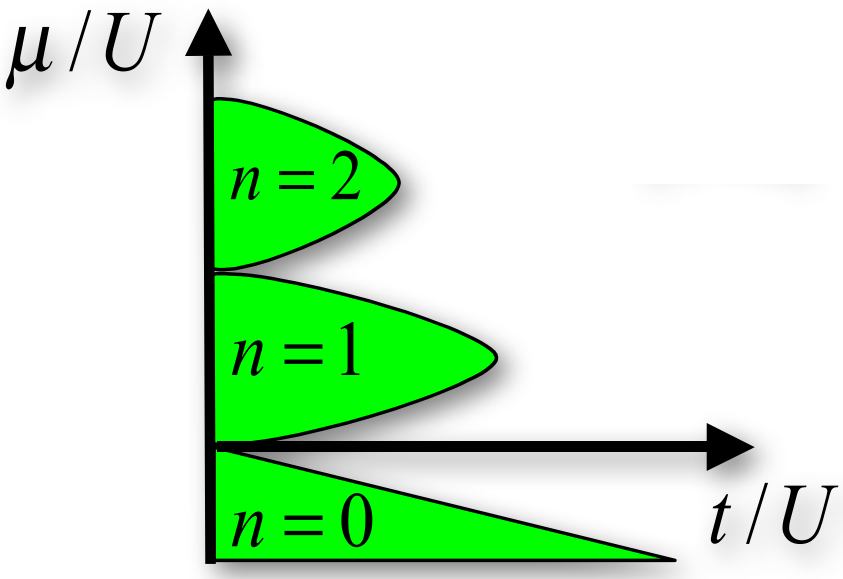
- Only valid in deep lattices where one band is enough
- Shallow lattices require alternative approaches
- Corrections to naïve calculations of *U* are hard
  - H. P. Büchler Phys. Rev. Lett. 104, 090402 (2010)
- Equilibration is an open issue in the experiments

# Bose-Hubbard model

Fisher *et al*, PRB 1989

- Use bosonic atoms for validation

$$H = -t \sum_{\langle i,j \rangle} \left( b_i^\dagger b_j + b_j^\dagger b_i \right) + U \sum_i n_i(n_i - 1)/2 - \mu \sum_i n_i$$



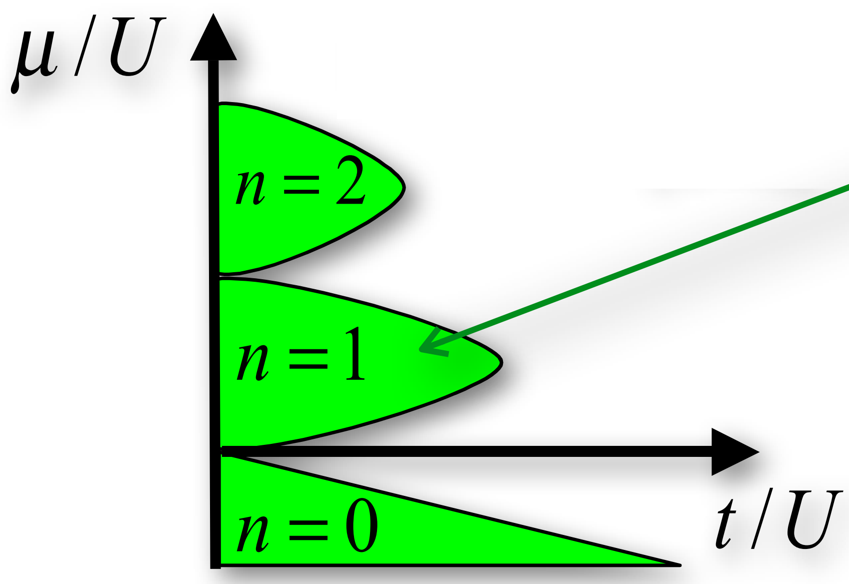
# Bose-Hubbard model

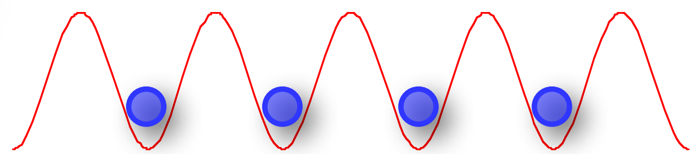
Fisher *et al*, PRB 1989

- Use bosonic atoms for validation

$$H = -t \sum_{\langle i,j \rangle} \left( b_i^\dagger b_j + b_j^\dagger b_i \right) + U \sum_i n_i(n_i - 1)/2 - \mu \sum_i n_i$$

Large *U*: incompressible Mott-insulator at Integer filling





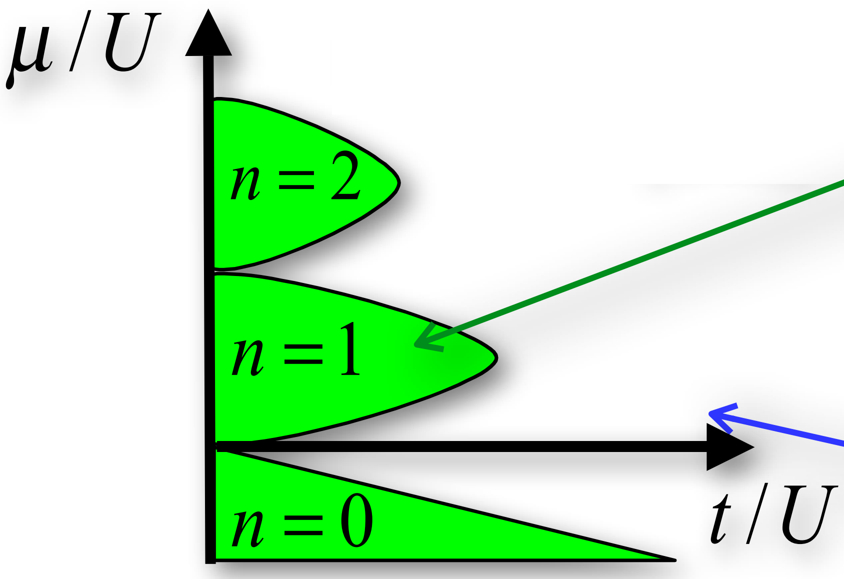
Saturday, January 21, 12

# Bose-Hubbard model

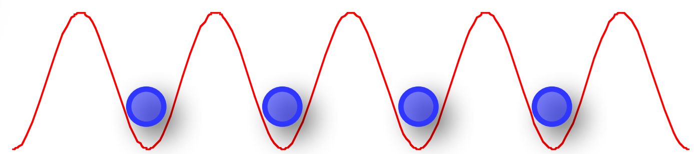
Fisher *et al*, PRB 1989

- Use bosonic atoms for validation

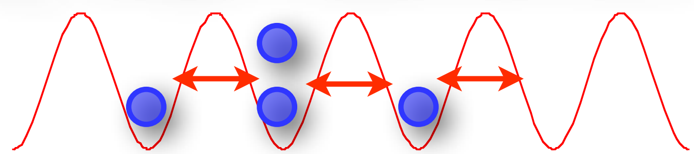
$$H = -t\sum_{\langle i,j\rangle}\left(b_i^\dagger b_j + b_j^\dagger b_i\right) + U\sum_i n_i(n_i-1)/2 - \mu\sum_i n_i$$




Large *U*: incompressible Mott-insulator at Integer filling



Large *t*: superfluid BEC

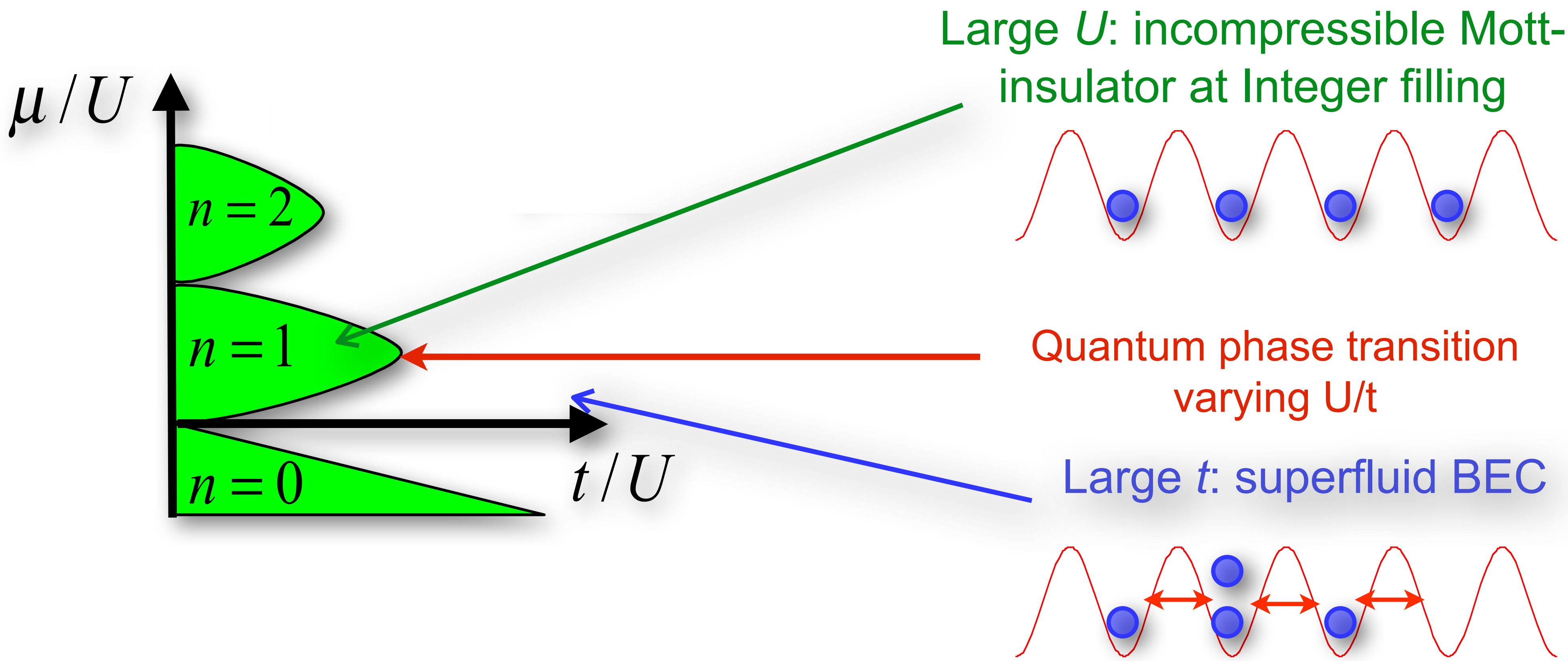


# Bose-Hubbard model

Fisher *et al*, PRB 1989

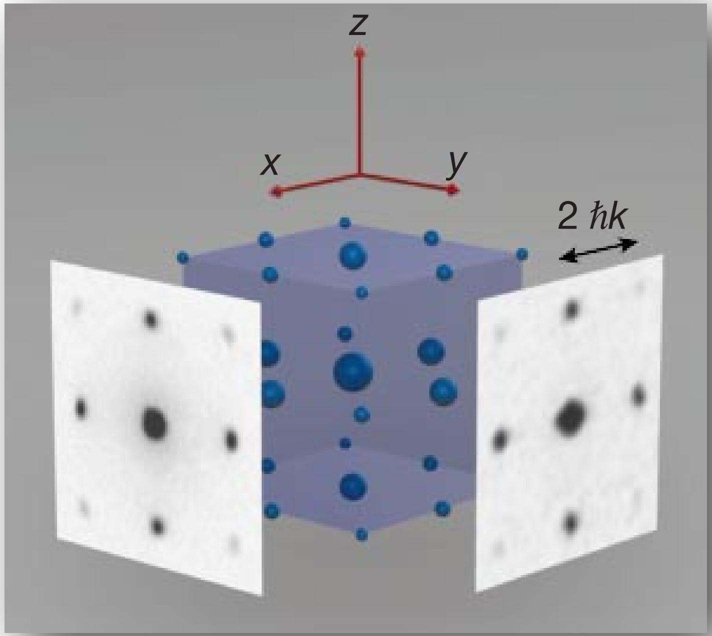
- Use bosonic atoms for validation

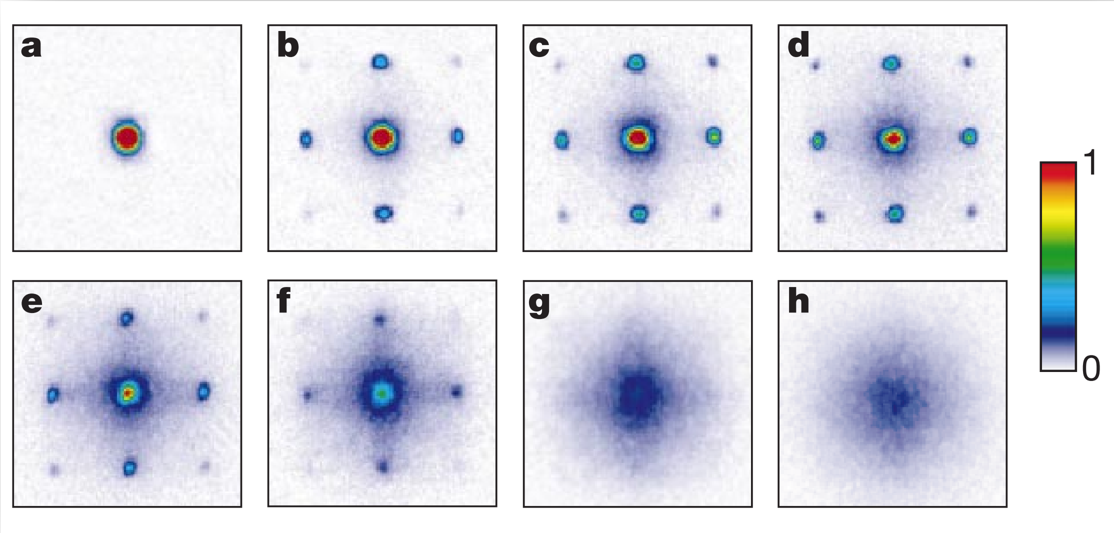
$$H = -t \sum_{\langle i,j \rangle} \left( b_i^\dagger b_j + b_j^\dagger b_i \right) + U \sum_i n_i(n_i - 1)/2 - \mu \sum_i n_i$$



# The first optical lattice experiments

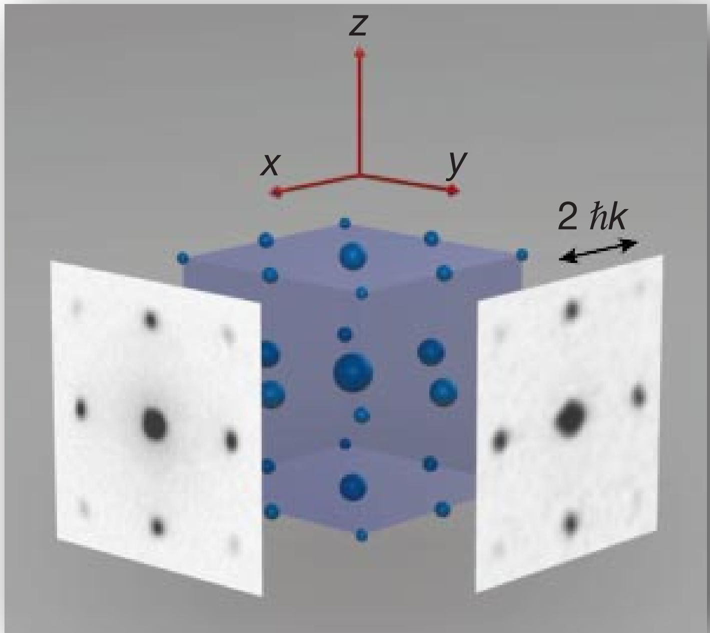
- Quantum phase transition as lattice depth is increased
  - Greiner et al, Nature (2002)
  - measuring the momentum distribution function in time-of-flight images



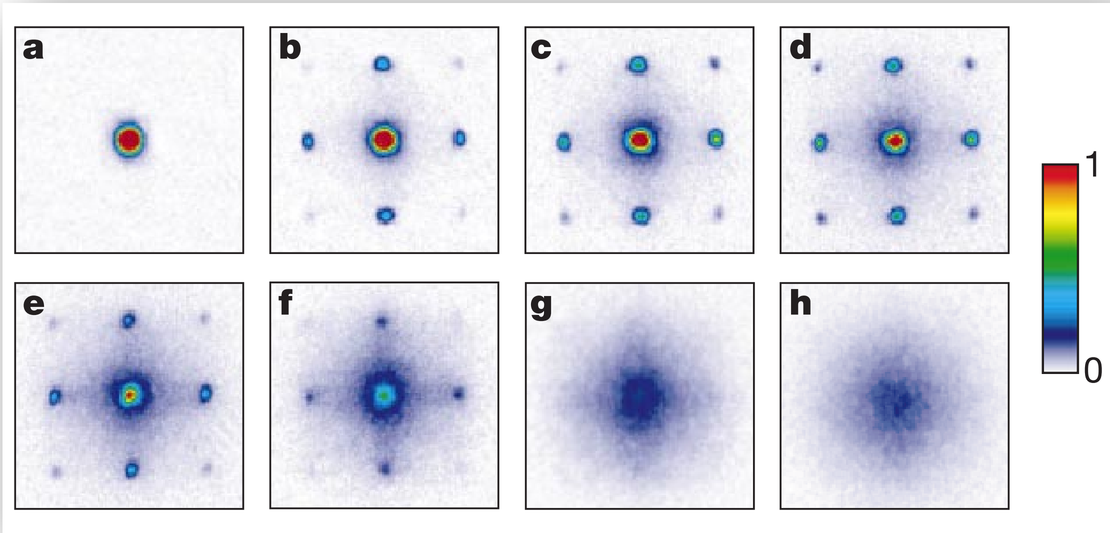



# The first optical lattice experiments

- Quantum phase transition as lattice depth is increased
  - Greiner et al, Nature (2002)
  - measuring the momentum distribution function in time-of-flight images

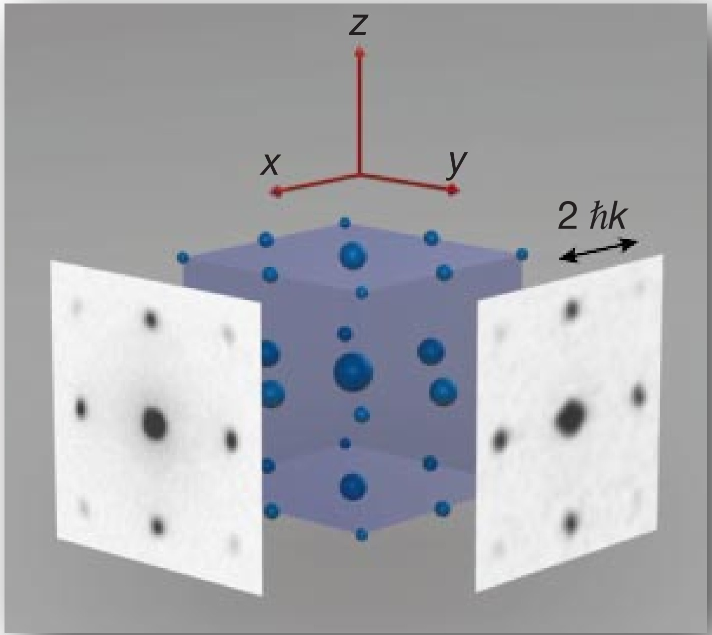



small *U*/*t*: condensate

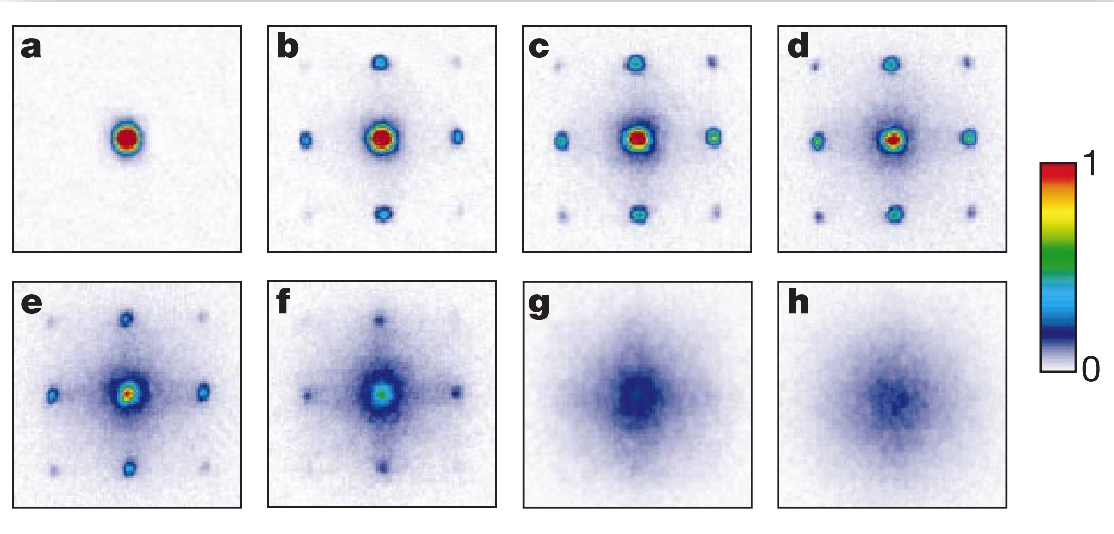


# The first optical lattice experiments

- Quantum phase transition as lattice depth is increased
  - Greiner et al, Nature (2002)
  - measuring the momentum distribution function in time-of-flight images



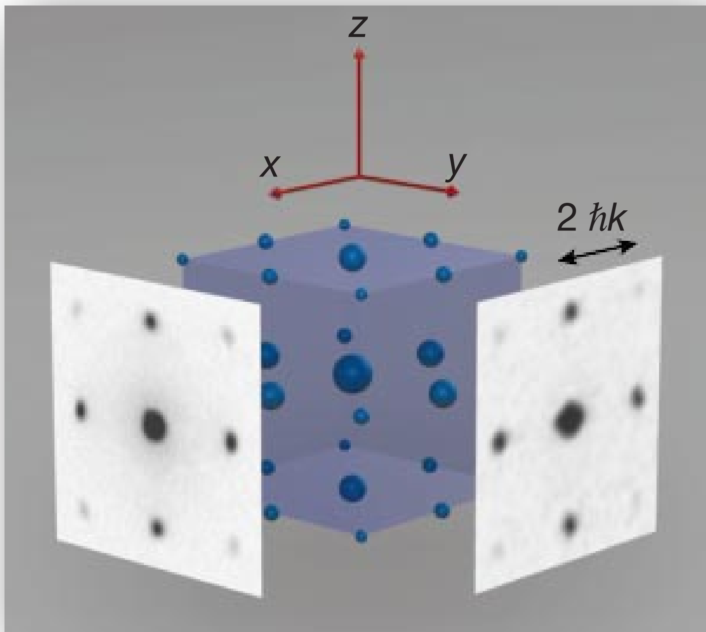

small *U*/*t*: condensate



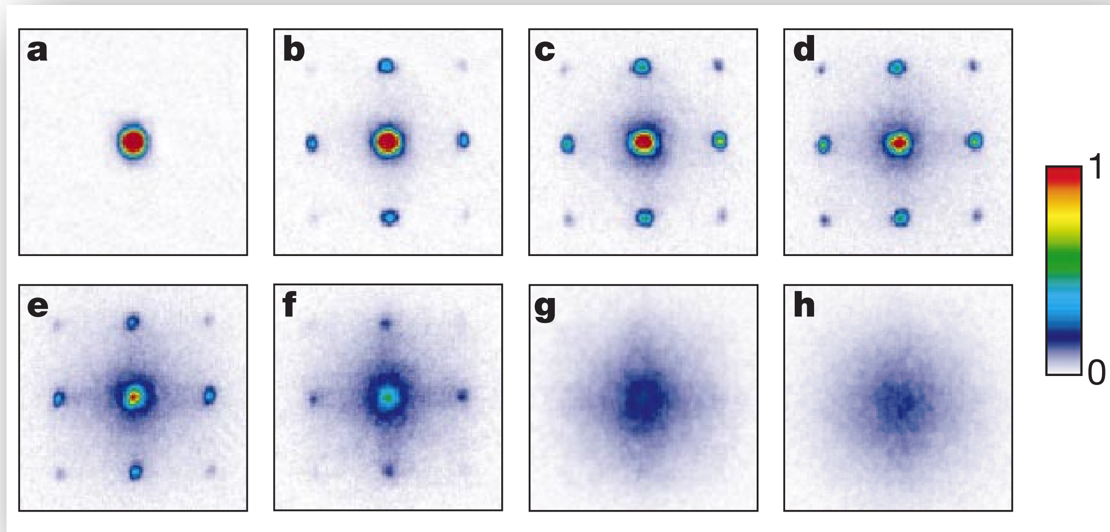

large *U*/*t*: Mott insulator

# The first optical lattice experiments

- Quantum phase transition as lattice depth is increased
  - Greiner et al, Nature (2002)
  - measuring the momentum distribution function in time-of-flight images




small *U*/*t*: condensate




large *U*/*t*: Mott insulator

Can this be made more quantitative?



Saturday, January 21, 12

# Validation by Quantum Monte Carlo simulations

# Validation by Quantum Monte Carlo simulations

- Approximation-free QMC simulations
  - worm algorithm, Prokof'ev, Svistunov and Tupitsyn, (1998)
  - up to 500,000 atoms, 220 x 220 x 200 ≈ 10 million sites
  - a single simulation takes only 10 hours on one CPU core

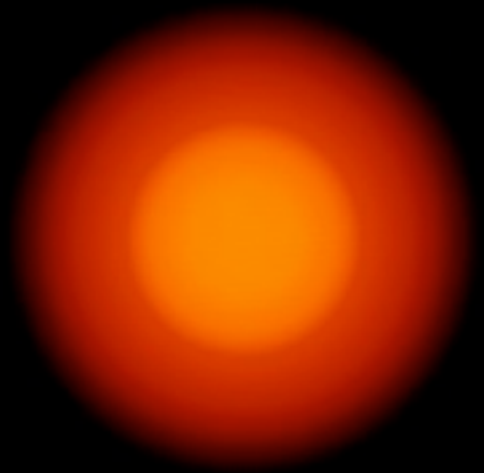
# Validation by Quantum Monte Carlo simulations

- Approximation-free QMC simulations
  - worm algorithm, Prokof'ev, Svistunov and Tupitsyn, (1998)
  - up to 500,000 atoms, 220 x 220 x 200 ≈ 10 million sites
  - a single simulation takes only 10 hours on one CPU core
- We can model all important details of the experiment
  - accurate microscopic model
  - same system size, particle numbers
  - temperature and entropy matched to experiment
  - measure quantities as observed in experiment

# QMC “images” of the boson cloud

# QMC “images” of the boson cloud

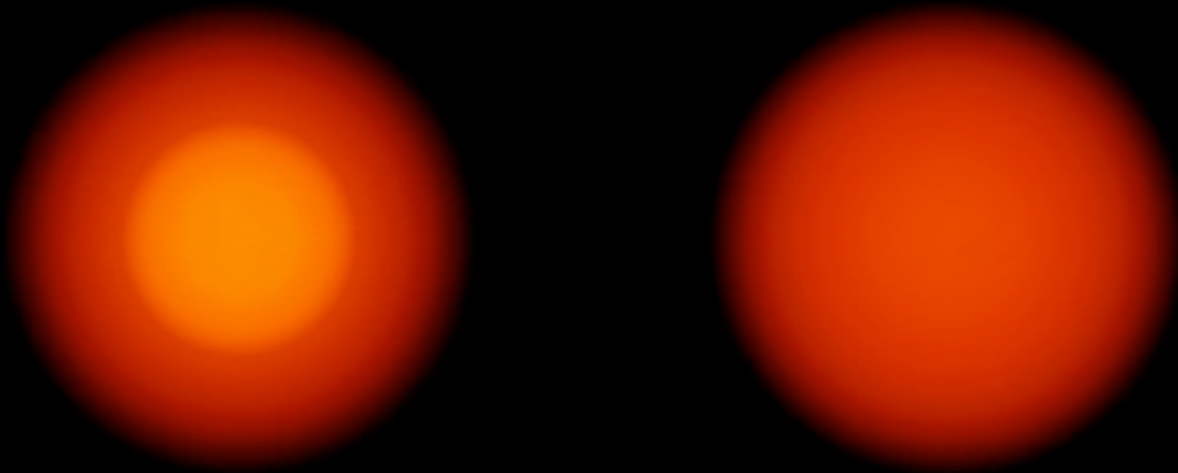
U/t = 10



# QMC “images” of the boson cloud

U/t = 10

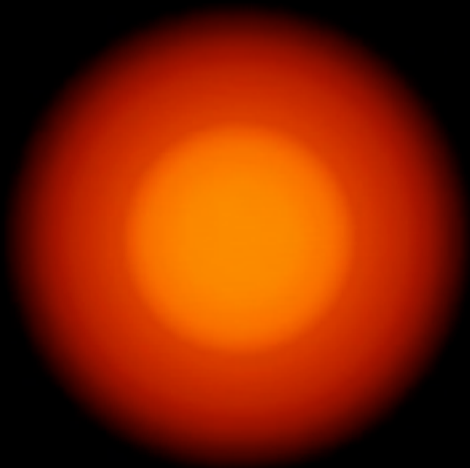
U/t = 25



# QMC “images” of the boson cloud

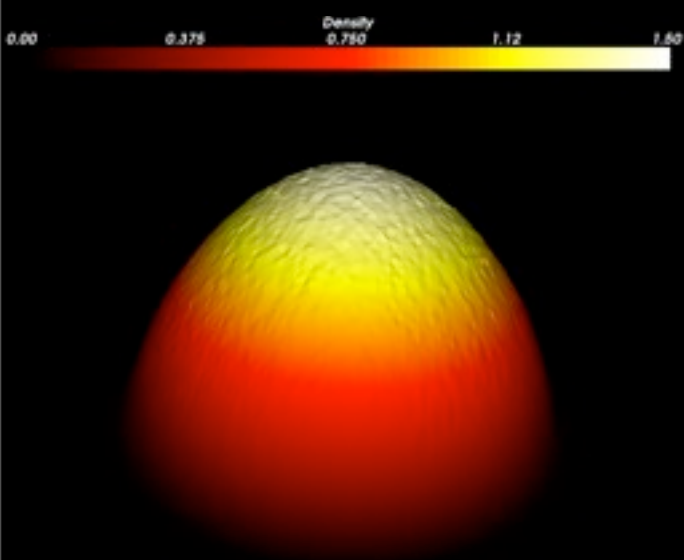
# QMC "images" of the boson cloud

U/t = 10



U/t = 25

U/t = 50

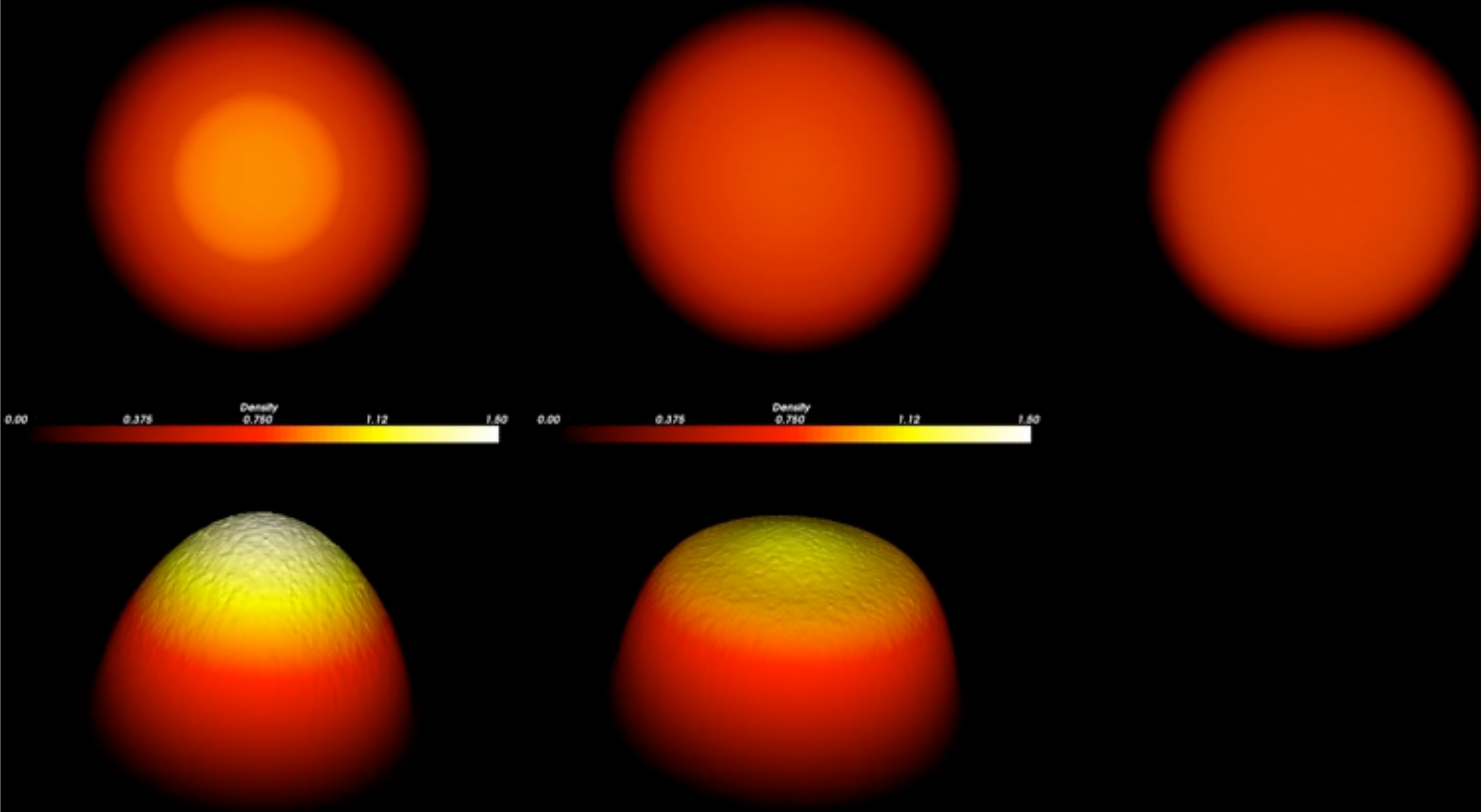


# QMC “images” of the boson cloud

U/t = 10

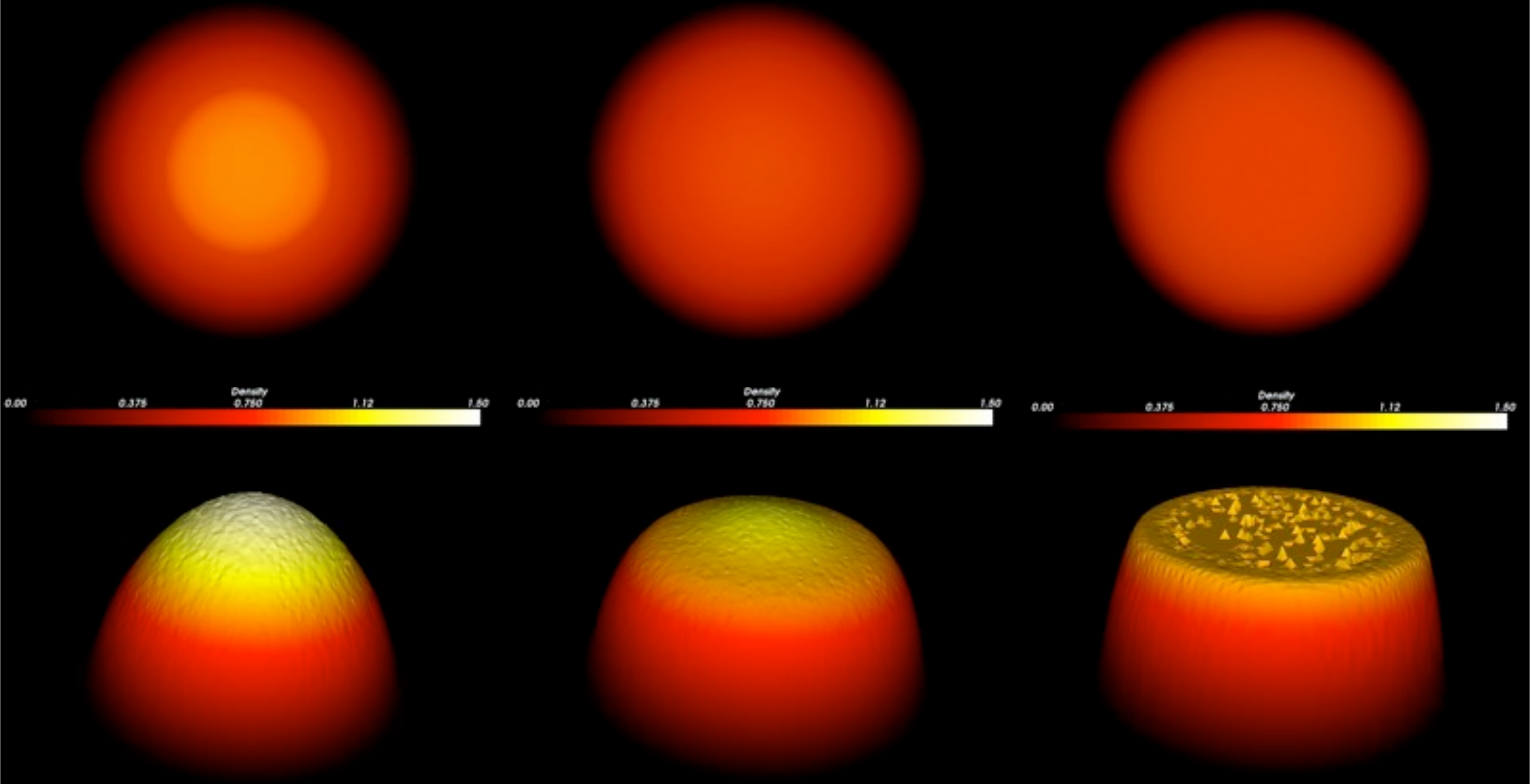
U/t = 25

U/t = 50



# QMC “images” of the boson cloud

**U/t = 10** **U/t = 25** **U/t = 50**

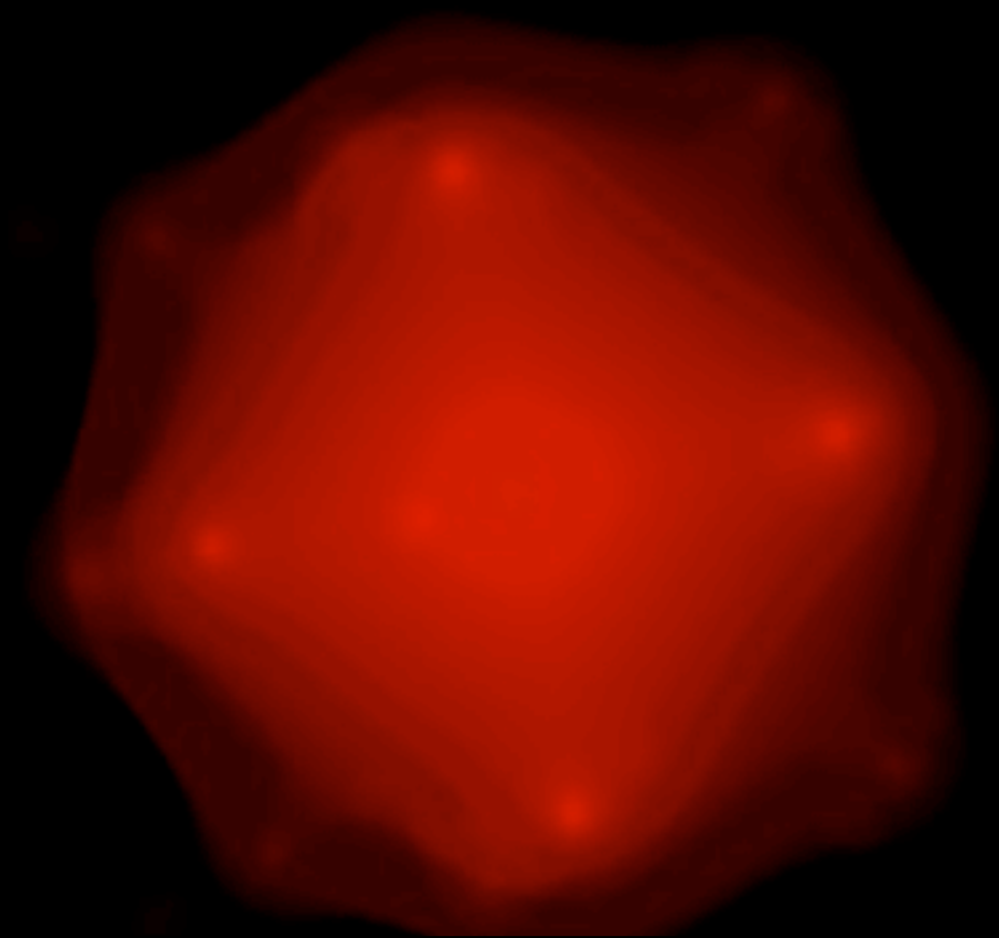


# Image after expansion – momentum distribution

## 3D image

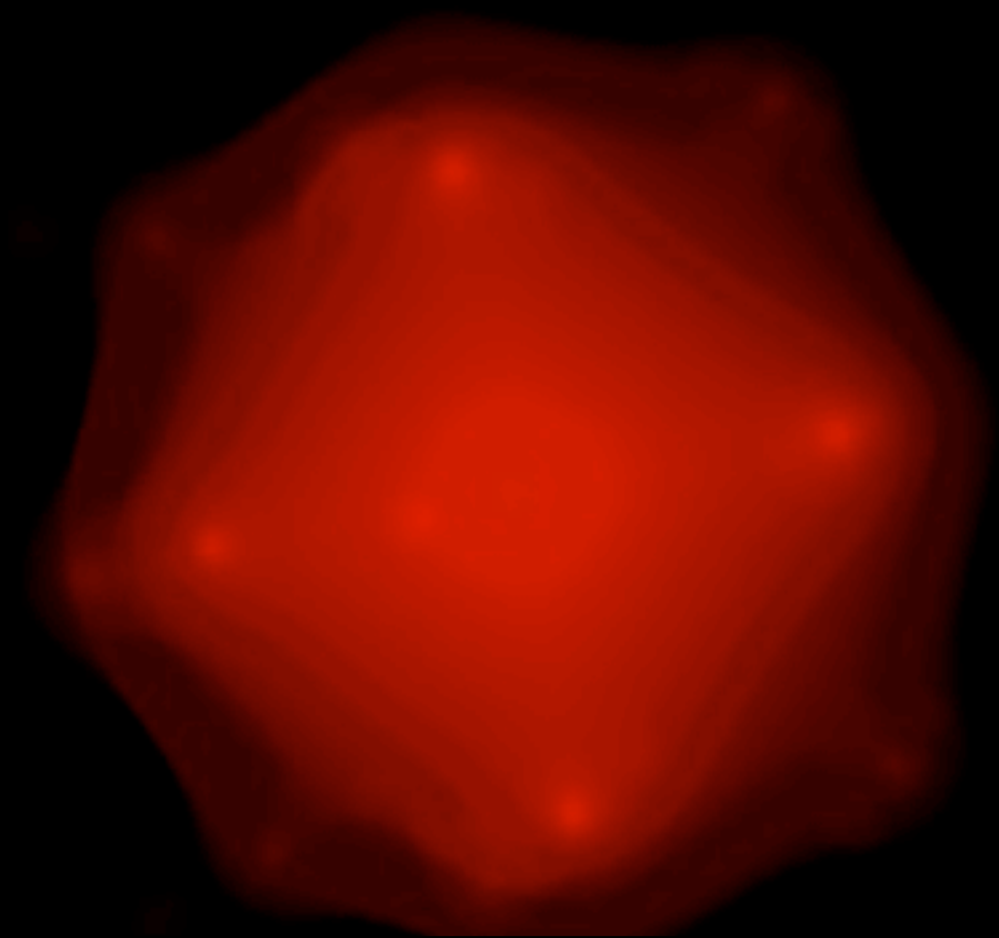
# Image after expansion – momentum distribution

3D image

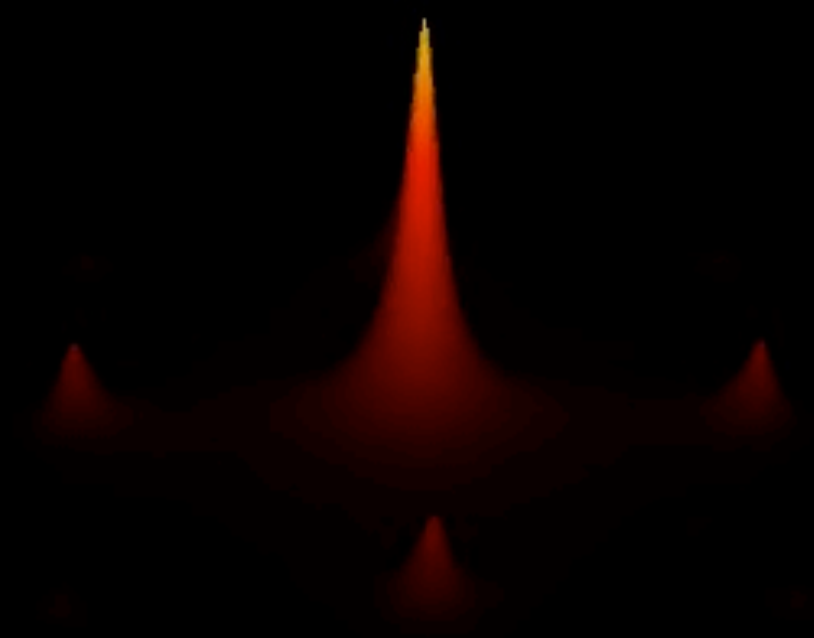


# Image after expansion – momentum distribution

## 3D image

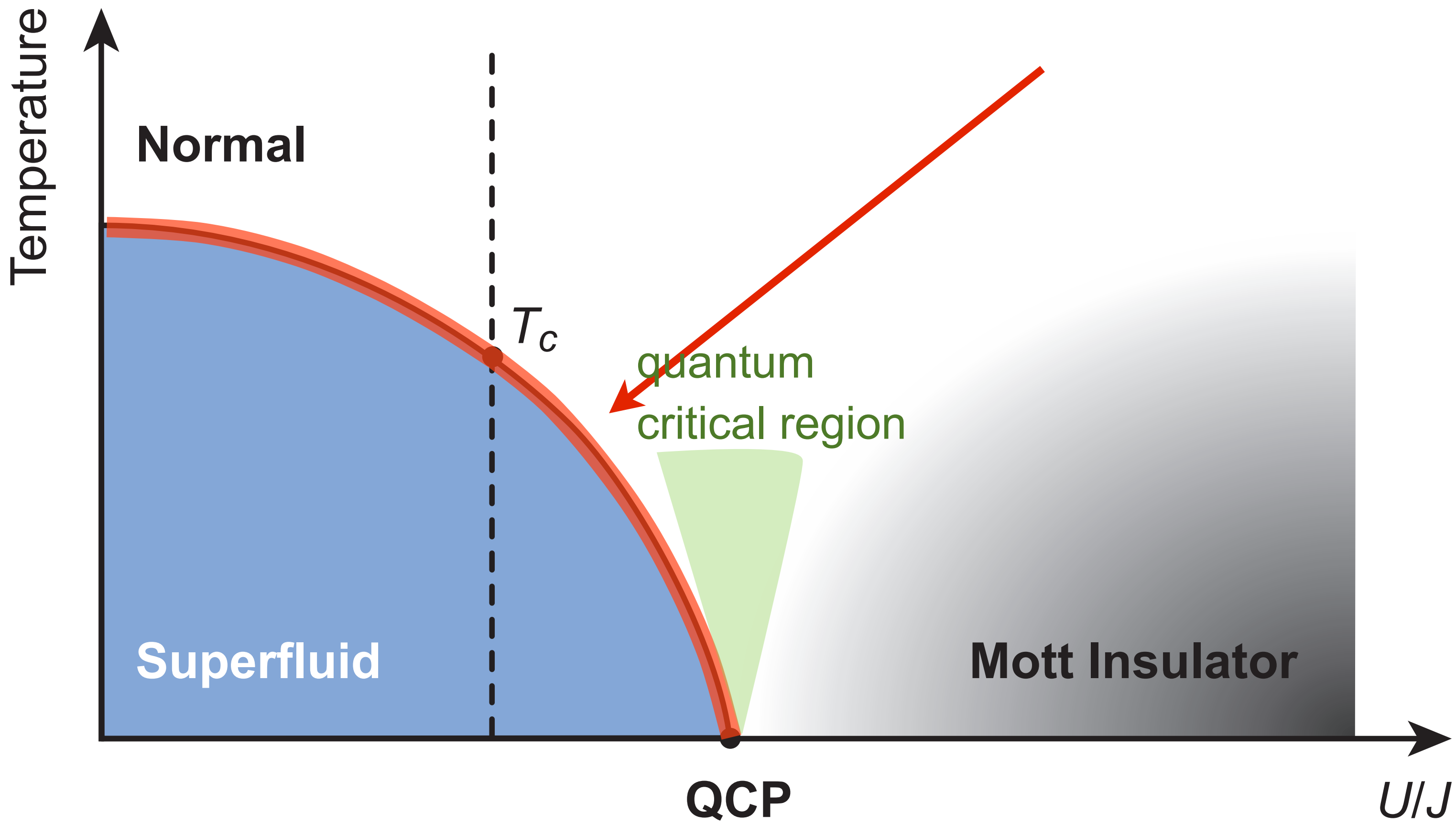


## Crosssection



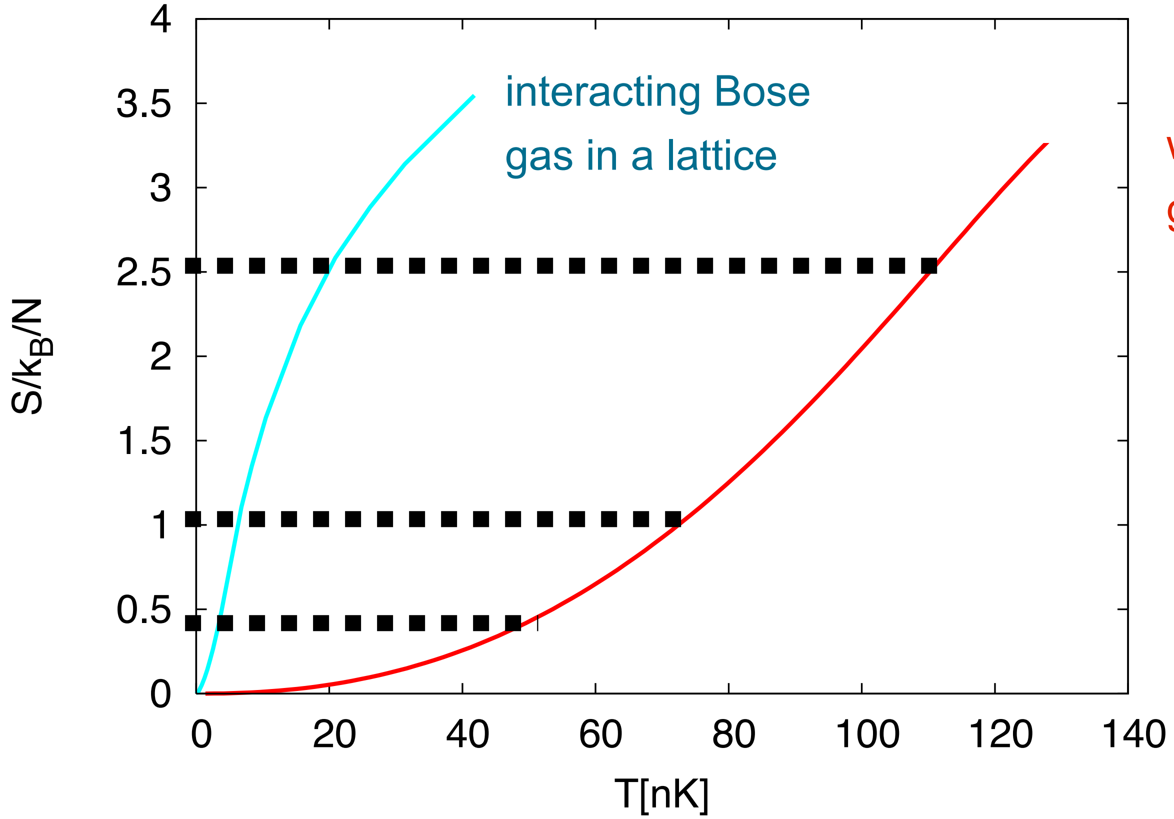
# Quantitative validation: the phase diagram

- Bosons in a 3D optical lattice at filling $n = 1$
- Measure suppression of $T_c$ close to the Mott insulator
- Particle number required to achieve $n = 1$ obtained from QMC

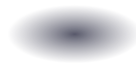


# Use QMC simulations for thermometry

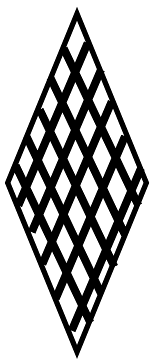
- Experiments work (ideally) at constant entropy!
  - Measure the momentum distribution before loading the gas into the lattice
  - Get its temperature and entropy fitting to a dilute Bose gas
  - Use QMC simulations to find the temperature for that entropy once loaded into an optical lattice (non-trivial simulations!)



# Accurately model time of flight (TOF) images

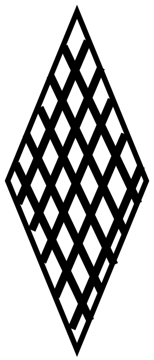


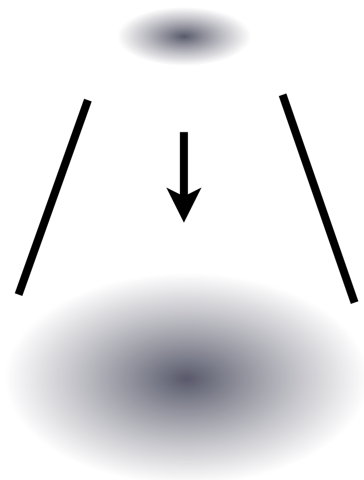
**CCD camera**



# Accurately model time of flight (TOF) images

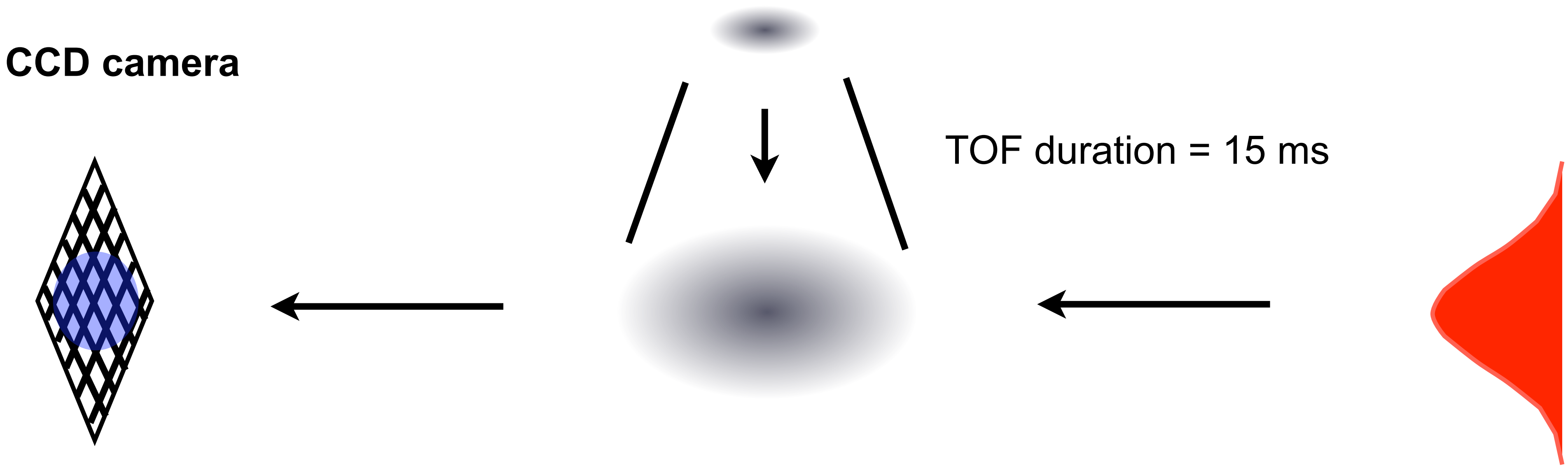
**CCD camera**





TOF duration = 15 ms

# Accurately model time of flight (TOF) images



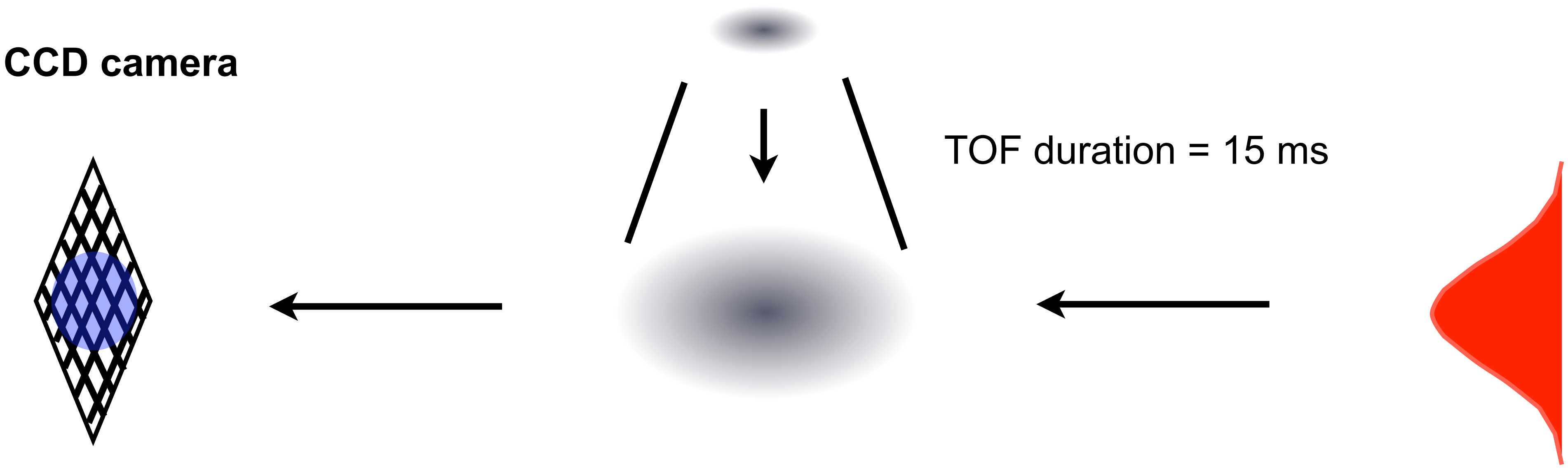

faster atoms fly farther

records the momentum distribution



Saturday, January 21, 12

# Accurately model time of flight (TOF) images



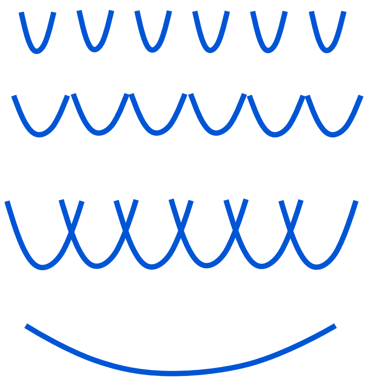

pixel size : 4.4 micron

further broadening
by optical elements

faster atoms fly farther

records the momentum distribution

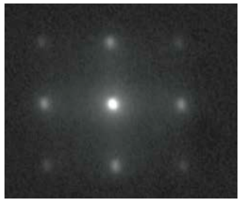
# Time-of-flight images: momentum distribution?



$$\hat{\Psi}(r,t) = \sum_\nu w_\nu(r,t)\hat{a}_\nu$$

$$w_\nu(r,t) = \langle r | e^{-i\frac{\hat{p}^2 t}{2m\hbar}} | w_\nu(t)\rangle$$

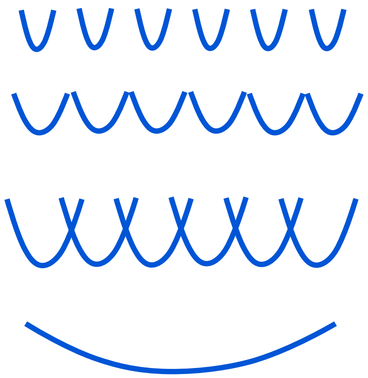
$$w_\nu(r,t) = \int \frac{d^3k}{(2\pi)^3} e^{-i\frac{\hbar k^2 t}{2m} + ik\cdot(r-r_\nu)}\tilde{w}(k)$$



$$\tilde{w}(k) \sim e^{-\frac{a_0^2 k^2}{2}}$$

$$n(r,t) = \sum_{\mu,\nu} w_\mu^*(r,t) w_\nu(r,t) \langle \hat{a}_\mu^\dagger \hat{a}_\nu \rangle$$

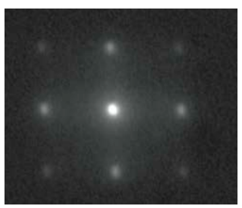
# Time-of-flight images: momentum distribution?



$$\hat{\Psi}(r,t) = \sum_{\nu} w_\nu(r,t)\hat{a}_\nu$$

$$w_\nu(r,t) = \langle r | e^{-i\frac{\hat{p}^2 t}{2m\hbar}} | w_\nu(t)\rangle$$

$$w_\nu(r,t) = \int \frac{d^3k}{(2\pi)^3} e^{-i\frac{\hbar k^2 t}{2m} + ik\cdot(r-r_\nu)}\tilde{w}(k)$$

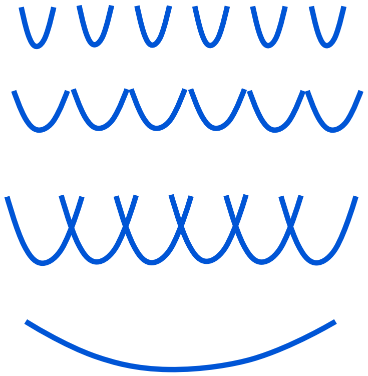


$$\tilde{w}(k) \sim e^{-\frac{a_0^2 k^2}{2}}$$

$$n(r,t) = \sum_{\mu,\nu} w_\mu^*(r,t) w_\nu(r,t)\langle \hat{a}_\mu^\dagger \hat{a}_\nu\rangle$$

Intensity

# Time-of-flight images: momentum distribution?

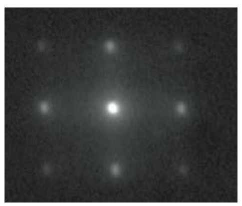


$$\hat{\Psi}(r,t) = \sum_\nu w_\nu(r,t)\hat{a}_\nu$$

contribution from all sites

$$w_\nu(r,t) = \langle r| e^{-i\frac{\hat{p}^2 t}{2m\hbar}} |w_\nu(t)\rangle$$

$$w_\nu(r,t) = \int \frac{d^3k}{(2\pi)^3} e^{-i\frac{\hbar k^2 t}{2m} + ik\cdot(r-r_\nu)} \tilde{w}(k)$$

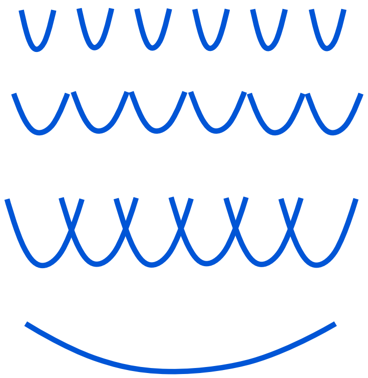


$$\tilde{w}(k) \sim e^{-\frac{a_0^2 k^2}{2}}$$

$$n(r,t) = \sum_{\mu,\nu} w_\mu^*(r,t) w_\nu(r,t) \langle \hat{a}_\mu^\dagger \hat{a}_\nu \rangle$$

Intensity

# Time-of-flight images: momentum distribution?



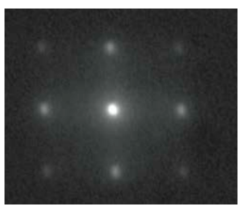
$$\hat{\Psi}(r,t) = \sum_{\nu} w_\nu(r,t)\hat{a}_\nu$$

contribution from all sites

$$w_\nu(r,t) = \langle r | e^{-i\frac{\hat{p}^2 t}{2m\hbar}} | w_\nu(t)\rangle$$

ballistic expansion

$$w_\nu(r,t) = \int \frac{d^3k}{(2\pi)^3} e^{-i\frac{\hbar k^2 t}{2m} + ik\cdot(r-r_\nu)}\tilde{w}(k)$$

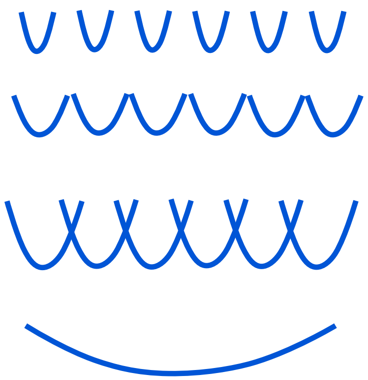


$$\tilde{w}(k) \sim e^{-\frac{a_0^2 k^2}{2}}$$

$$n(r,t) = \sum_{\mu,\nu} w_\mu^*(r,t) w_\nu(r,t) \langle \hat{a}_\mu^\dagger \hat{a}_\nu \rangle$$

Intensity

# Time-of-flight images: momentum distribution?



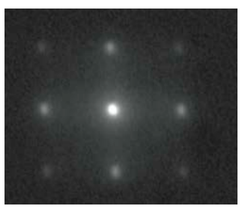
$$\hat{\Psi}(r,t) = \sum_\nu w_\nu(r,t)\hat{a}_\nu$$

contribution from all sites

$$w_\nu(r,t) = \langle r|e^{-i\frac{\hat{p}^2 t}{2m\hbar}}|w_\nu(t)\rangle$$

ballistic expansion

$$w_\nu(r,t) = \int \frac{d^3k}{(2\pi)^3} e^{-i\frac{\hbar k^2 t}{2m}+ik\cdot(r-r_\nu)}\tilde{w}(k)$$



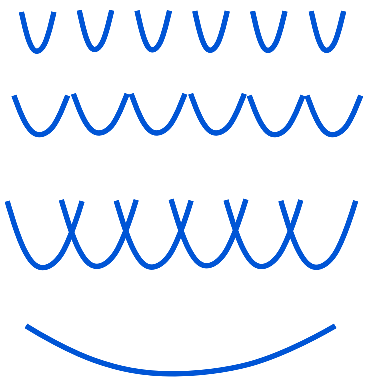
$$\tilde{w}(k) \sim e^{-\frac{a_0^2 k^2}{2}}$$

Gaussian approximation for Wannier function

$$n(r,t) = \sum_{\mu,\nu} w_\mu^*(r,t)w_\nu(r,t)\langle\hat{a}_\mu^\dagger\hat{a}_\nu\rangle$$

Intensity

# Time-of-flight images: momentum distribution?



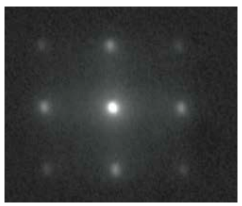
$$\hat{\Psi}(r,t) = \sum_{\nu} w_{\nu}(r,t)\hat{a}_{\nu}$$

contribution from all sites

$$w_{\nu}(r,t) = \langle r | e^{-i\frac{\hat{p}^2 t}{2m\hbar}} | w_{\nu}(t)\rangle$$

ballistic expansion

$$w_{\nu}(r,t) = \int \frac{d^3k}{(2\pi)^3} e^{-i\frac{\hbar k^2 t}{2m} + ik\cdot(r-r_{\nu})}\tilde{w}(k)$$



$$\tilde{w}(k) \sim e^{-\frac{a_0^2 k^2}{2}}$$

Gaussian approximation for Wannier function

$$n(r,t) = \sum_{\mu,\nu} w_{\mu}^{*}(r,t) w_{\nu}(r,t)\langle \hat{a}_{\mu}^{\dagger}\hat{a}_{\nu}\rangle$$

density matrix from QMC

Intensity

# Time of flight (TOF) images

Do we really measure n(k) in experiment?

F. Gerbier *et al*, PRL (2008)

$$n(\mathbf{r},t) \sim e^{-i\frac{\mathbf{K}(\mathbf{r}_\mu-\mathbf{r}_\nu)}{1+\delta^2}-i\frac{m(r_\nu^2-r_\mu^2)}{2\hbar t(1+\delta^2)}-\frac{a_0K^2}{1+\delta^2}}$$
$$\times \quad e^{\frac{\delta\mathbf{K}(\mathbf{r}_\mu+\mathbf{r}_\nu)}{1+\delta^2}-\frac{m\delta(\mathbf{r}_\mu^2+\mathbf{r}_\nu^2)}{2\hbar t(1+\delta^2)}}\langle\hat{a}_\mu^\dagger\hat{a}_\nu\rangle.$$

$a_0$ : width of the initial Gaussian Wannier function

$$\mathbf{K}=\frac{m\mathbf{r}}{\hbar t},$$ "quasi-momentum"

$$K_y=\frac{r_y m}{\hbar t}+gmt$$

$$\delta=\frac{ma_0^2}{\hbar t}=2\frac{m\lambda^2}{8\hbar t}\left(\frac{a_0}{\lambda/2}\right)^2\approx 5.10^{-4}$$

taking gravity semi-classically into account

# Time of flight (TOF) images

Do we really measure n(k) in experiment?

F. Gerbier *et al*, PRL (2008)

$$n(\mathbf{r},t) \sim e^{-i\frac{\mathbf{K}(\mathbf{r}_\mu-\mathbf{r}_\nu)}{1+\delta^2} - i\frac{m(r_\nu^2-r_\mu^2)}{2\hbar t(1+\delta^2)} - \frac{a_0 K^2}{1+\delta^2}} \times e^{\frac{\delta\mathbf{K}(\mathbf{r}_\mu+\mathbf{r}_\nu)}{1+\delta^2} - \frac{m\delta(\mathbf{r}_\mu^2+\mathbf{r}_\nu^2)}{2\hbar t(1+\delta^2)}} \langle \hat{a}_\mu^\dagger \hat{a}_\nu \rangle.$$

$a_0$ : width of the initial Gaussian Wannier function

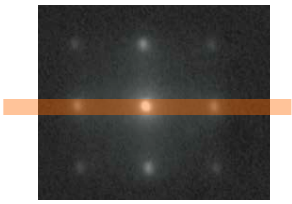
$$\mathbf{K} = \frac{m\mathbf{r}}{\hbar t},$$ "quasi-momentum"

$$K_y = \frac{r_y m}{\hbar t} + gmt$$

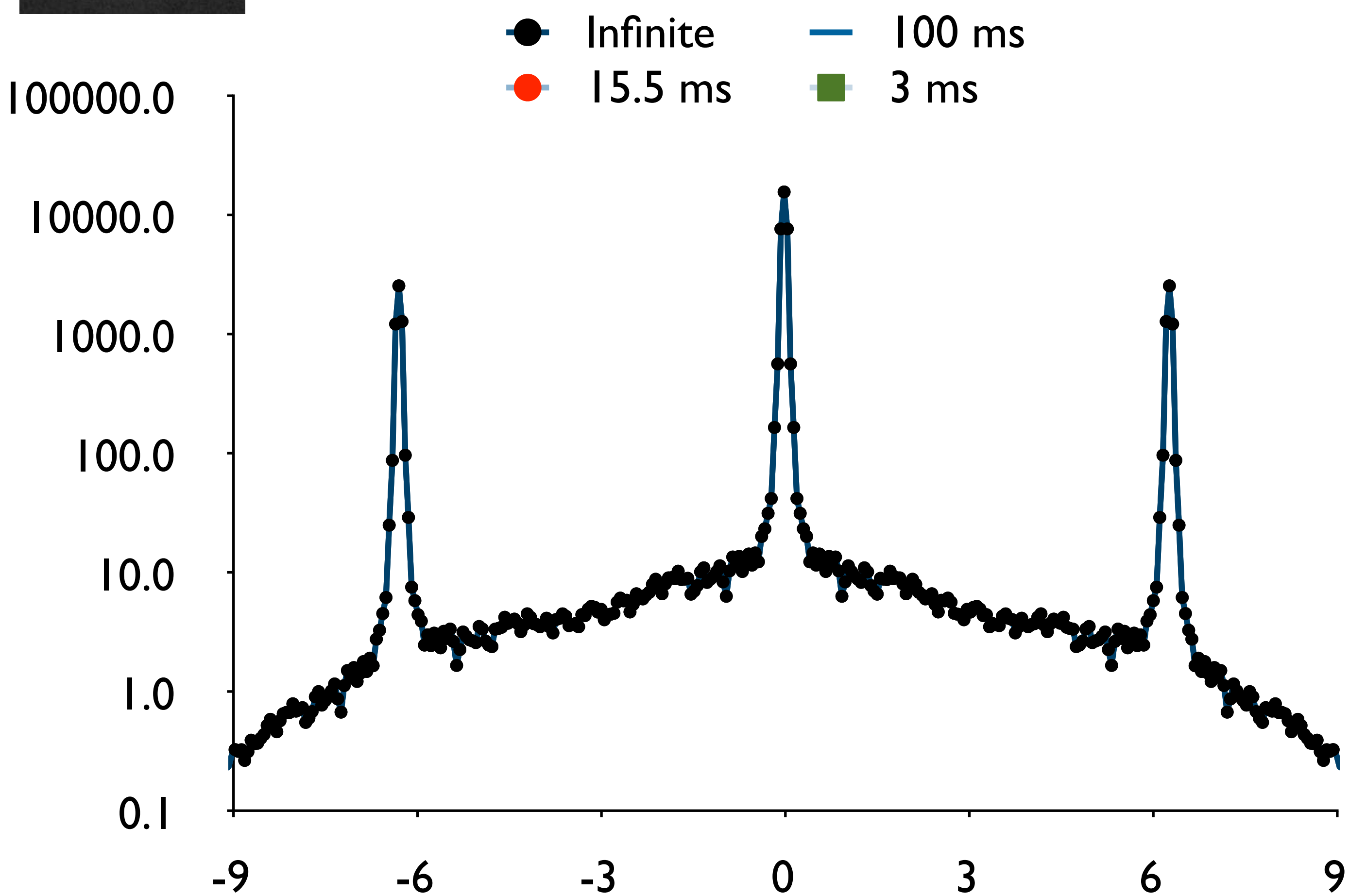
$$\delta = \frac{m a_0^2}{\hbar t} = 2\frac{m\lambda^2}{8\hbar t}\left(\frac{a_0}{\lambda/2}\right)^2 \approx 5.10^{-4}$$

taking gravity semi-classically into account

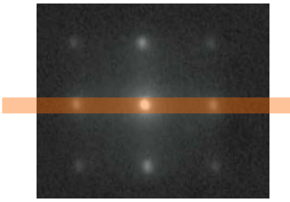
# Finite time of flight broadens the peaks



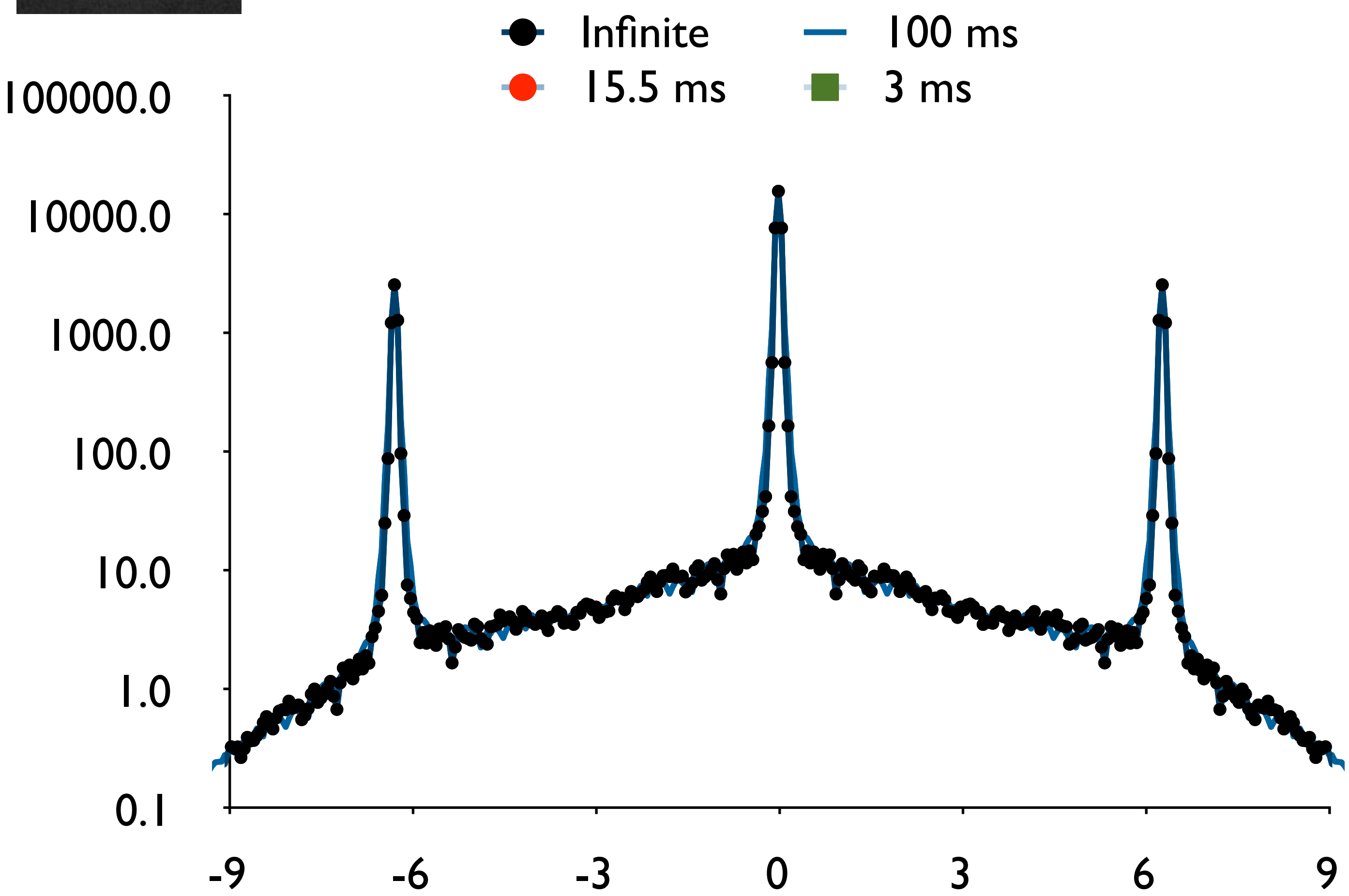
F. Gerbier *et al*, PRL (2008)



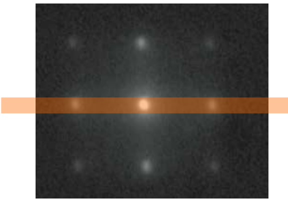
# Finite time of flight broadens the peaks



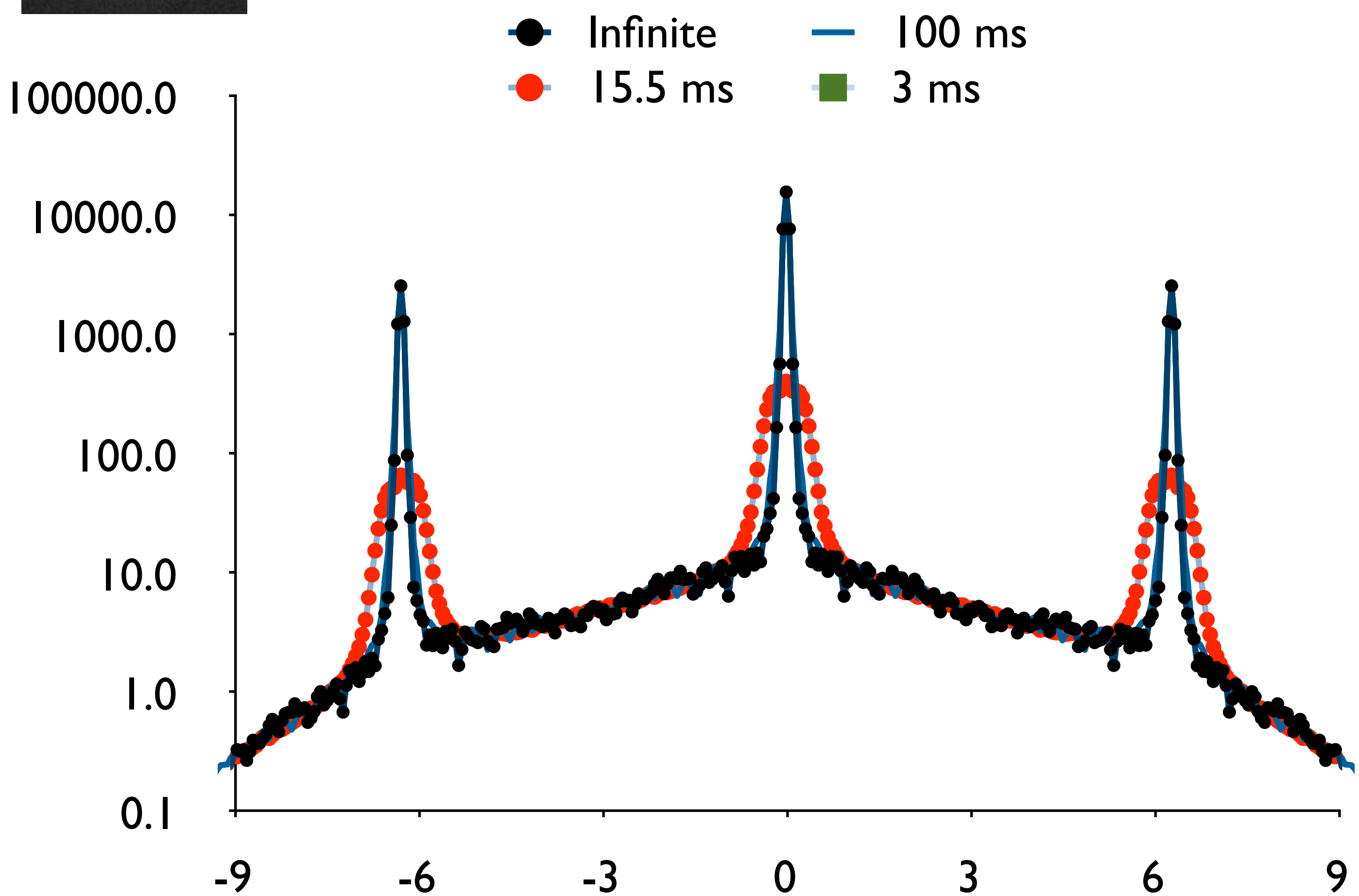
F. Gerbier *et al*, PRL (2008)



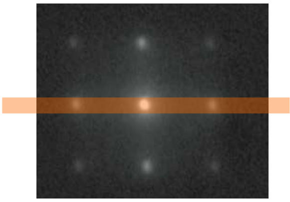
# Finite time of flight broadens the peaks



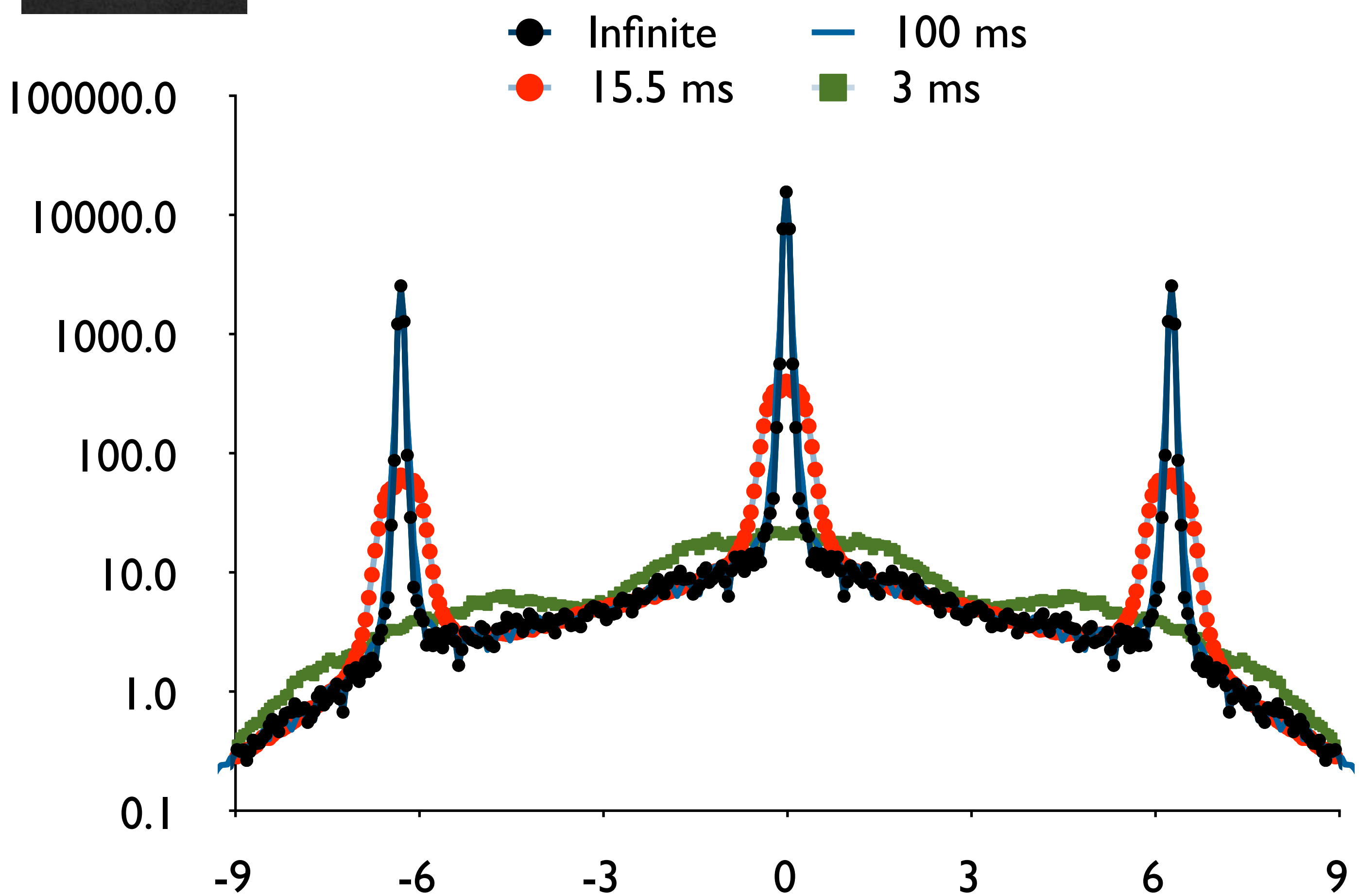
F. Gerbier *et al*, PRL (2008)



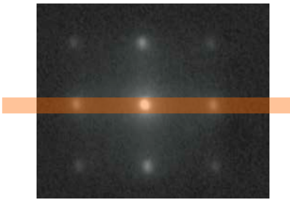
# Finite time of flight broadens the peaks



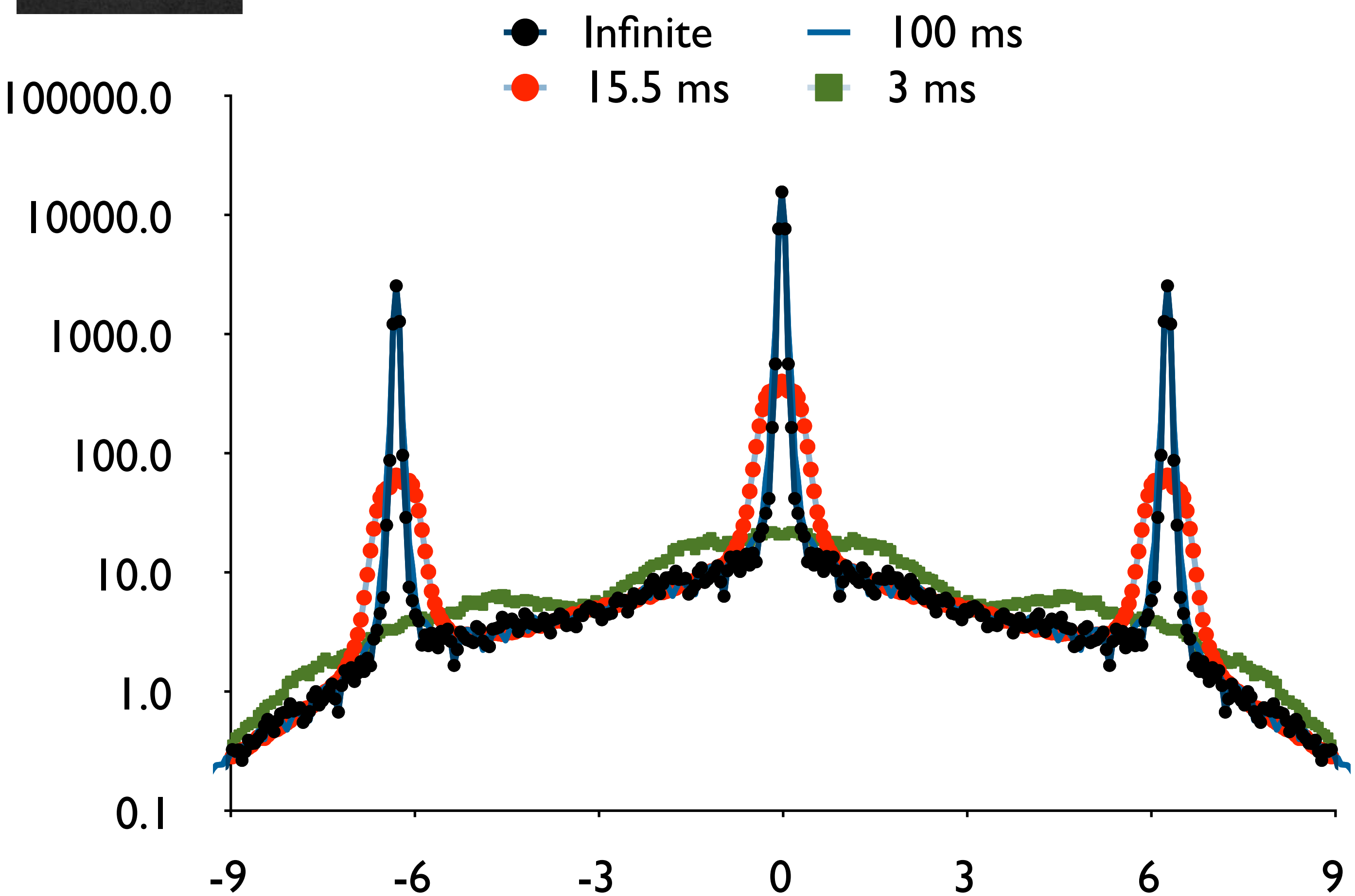
F. Gerbier *et al*, PRL (2008)



# Finite time of flight broadens the peaks



F. Gerbier *et al*, PRL (2008)




finite time of flight cuts off spatial correlations

broadens peaks

# Validating the quantum simulator

# Validating the quantum simulator

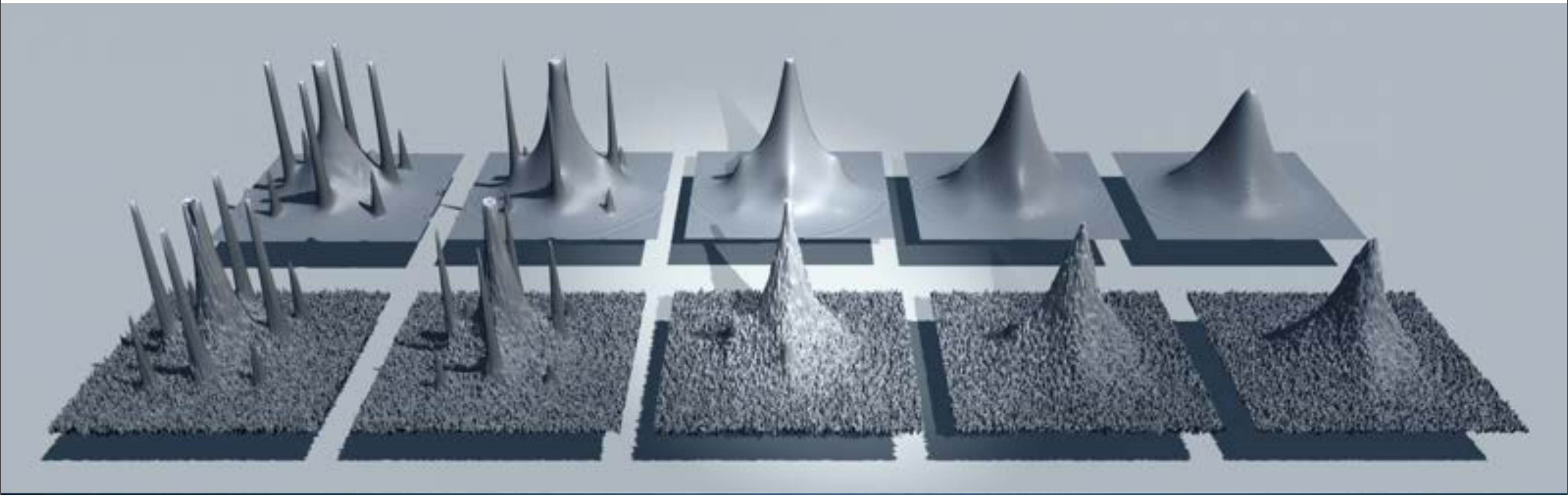
- We model all important details of the experiment
  - Same trap, interactions, particle number, total entropy
  - Calculate what the experiment should see

# Validating the quantum simulator

- We model all important details of the experiment
  - Same trap, interactions, particle number, total entropy
  - Calculate what the experiment should see
- The experiment should better reproduce what we get!

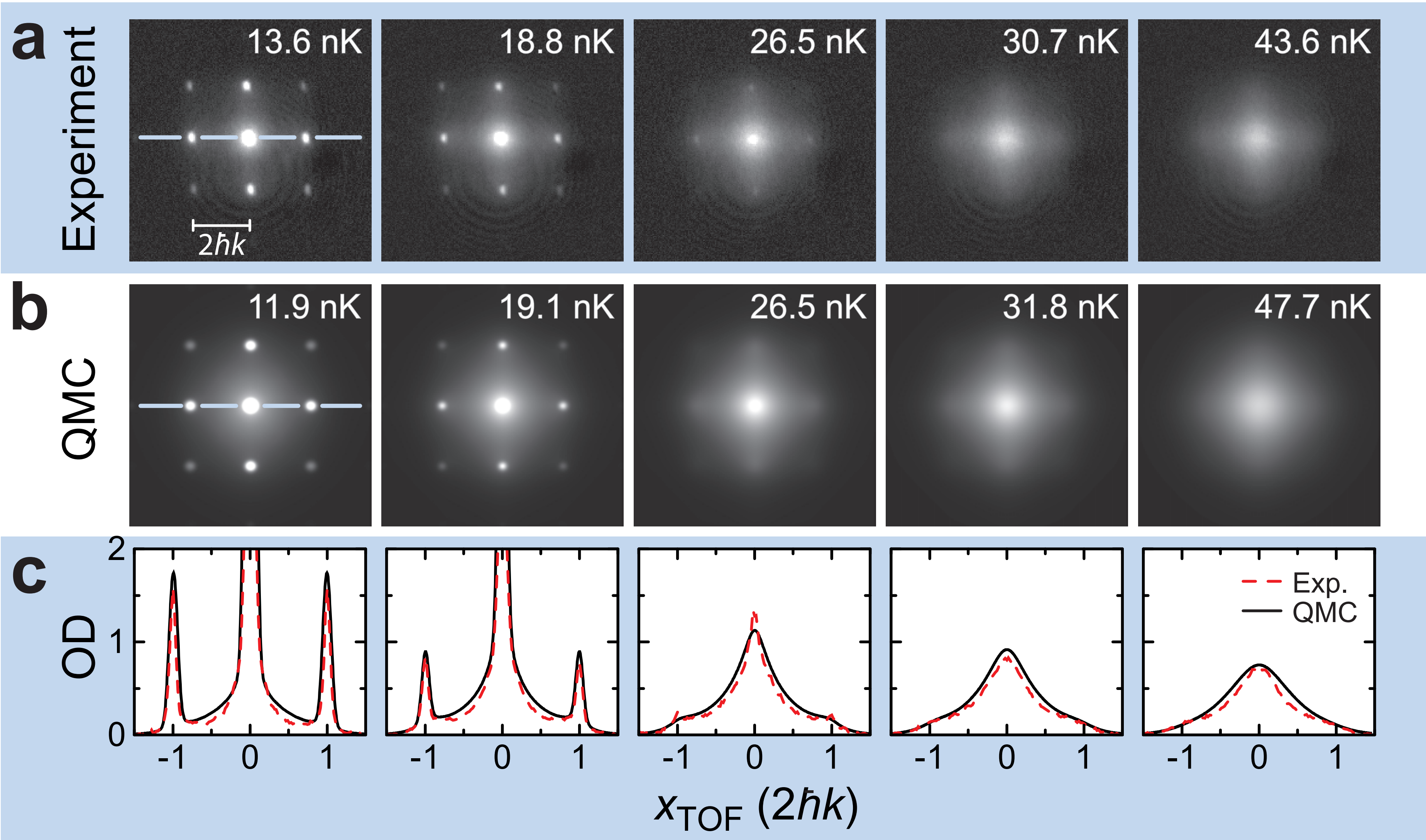
# Validating the quantum simulator

- We model all important details of the experiment
  - Same trap, interactions, particle number, total entropy
  - Calculate what the experiment should see
- The experiment should better reproduce what we get!
- and it does: Trotzky, Pollet *et al*, Nature Phys. **6**, 998 (2010).



# Validation of experiment by QMC: small *U/t*

$V_0 = 8E_r\,,\quad U/J = 8.11\,,\quad T_c = 26.5\text{nK}$

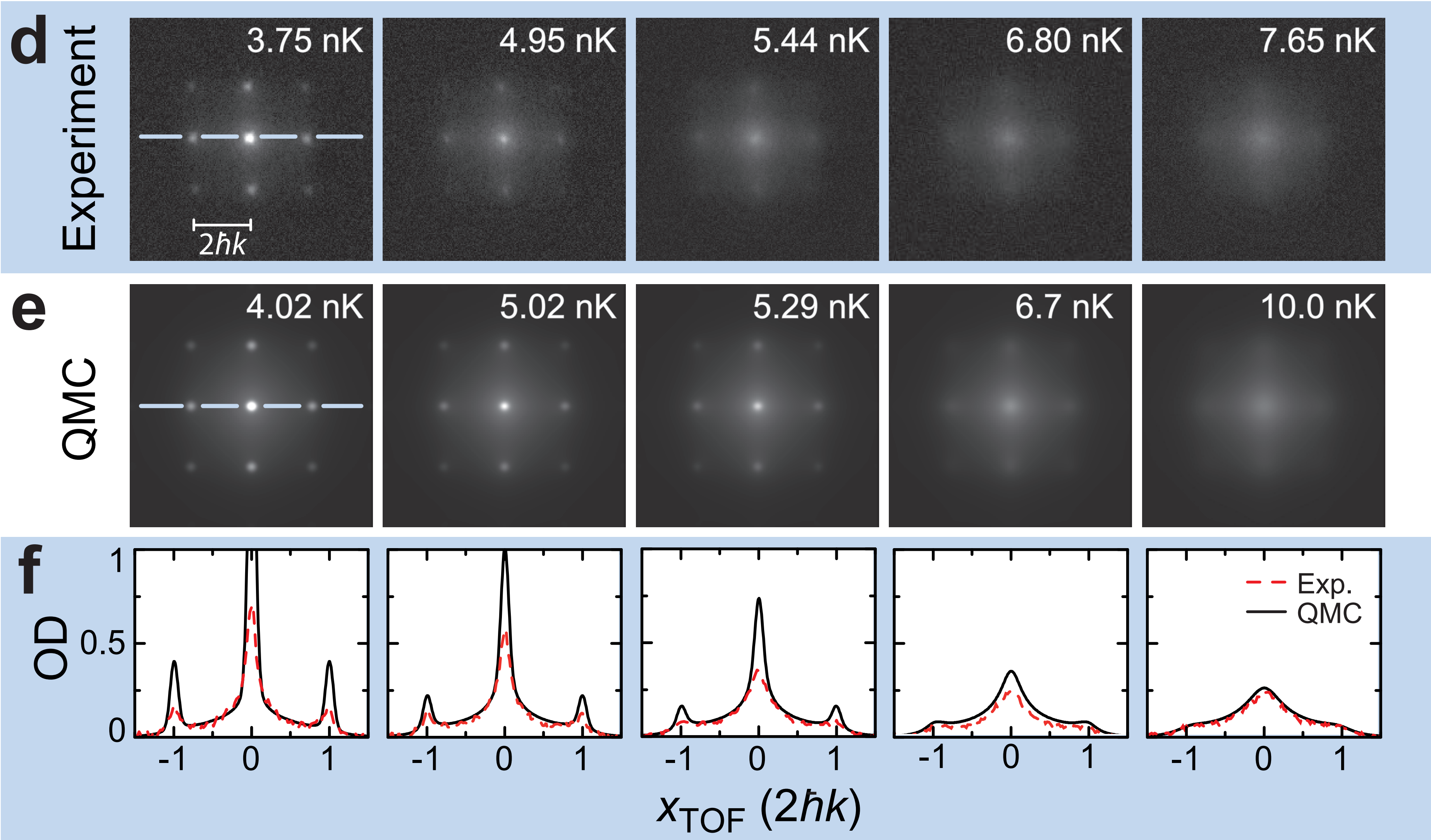




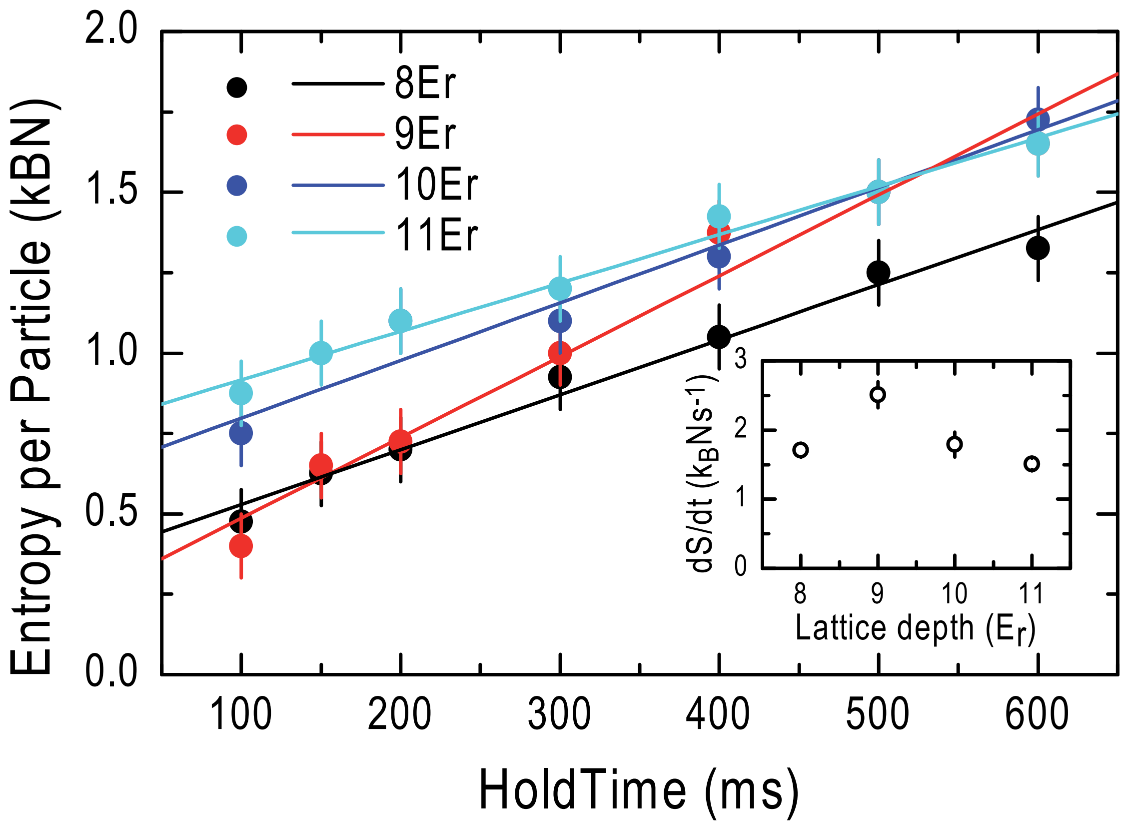

Saturday, January 21, 12

# Validation of experiment by QMC: large *U/t*

$$V_0 = 11.75 E_r\,,\quad U/J = 27.5\,,\quad T_c = 5.31\,\text{nK}$$



# Non-adiabaticity: heating from lattice laser

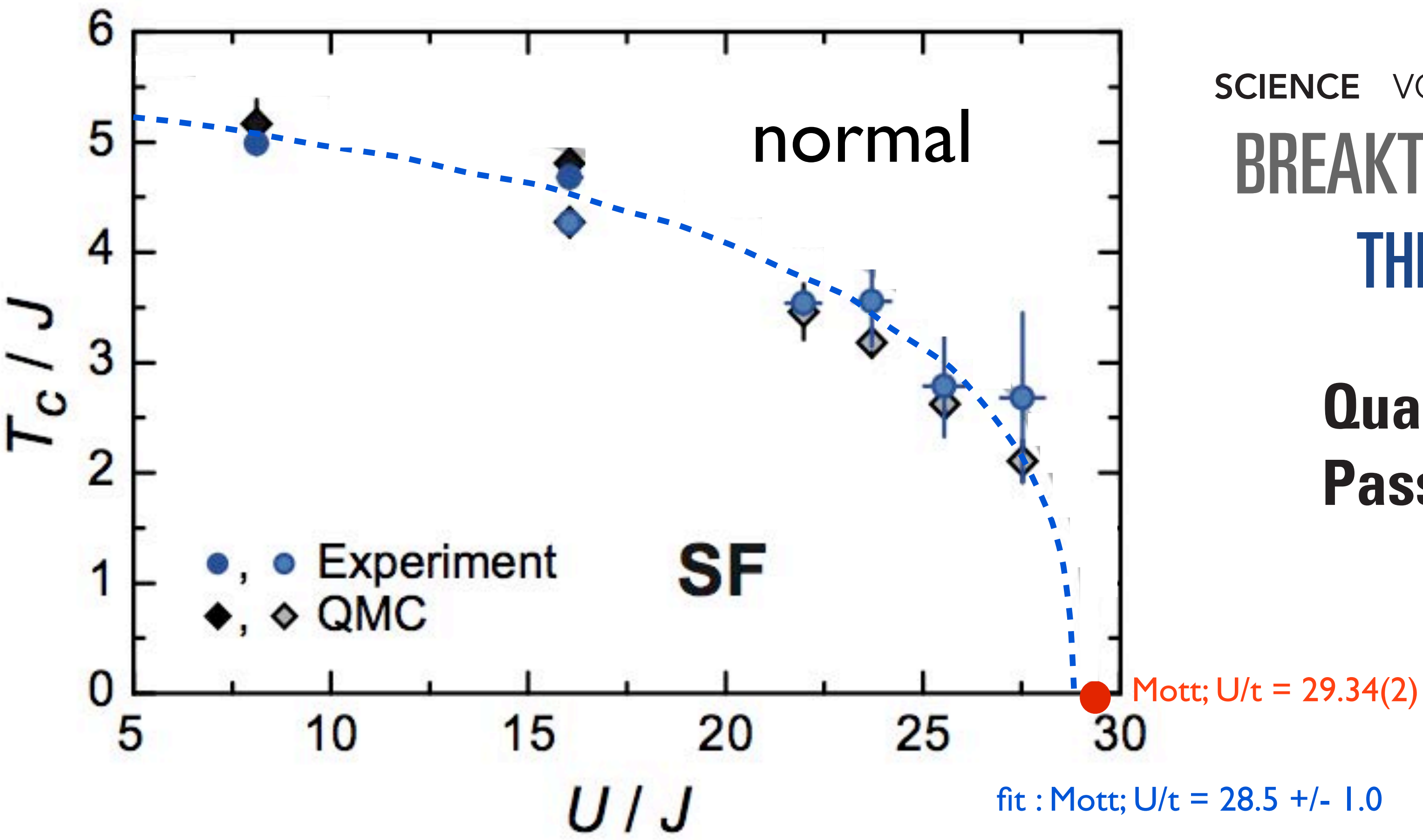



Entropy determined by comparing TOF images to QMC simulations

Severe limitation on accessible temperatures in experiments

Theoretically discussed by H. Pichler, A. J. Daley, P. Zoller, PRA (2011)

# Phase diagram obtained by the quantum simulation




SCIENCE VOL 330 17 DECEMBER 2010

BREAKTHROUGH OF THE YEAR

THE RUNNERS-UP

**Quantum Simulators Pass First Key Test**