Monte Carlo Simulation of Spins

§1 MC Simulation of Lattices

ISING MODEL[1]

- **Ising model**: A model in statistical mechanics, which was originally used to study the behavior of magnetic particles in a magnetic field. The model consists of a collection of "spins" on lattice sites (see the Figure below). Each spin takes only the value of +1 (up) or −1 (down). The model has also been used to study alloys and even the load-balancing problem in parallel computing.

  The potential energy of a given spin configuration is

$$V(s^N) = -J\sum_{(k,l)} s_k s_l - H\sum_k s_k, \quad s_k = \pm 1 \quad ,$$

where (k, l) are nearest neighbor pairs of lattice sites, H is the external magnetic field, and J is the exchange coupling.

For a positive J, the first term takes the minimum value if all the spins are aligned. The latter term, for a positive H, takes minimum, when the spins are all up. In statistical mechanics, the all-down configuration is most probable, but other configurations also exist with a probability proportional to the Boltzmann factor, $\exp(-V/k_BT)$.

For an $L \times L$ two-dimensional lattice, each spin index consists of a pair of integers, $k = (i,j)$ $(0 \le i,j \le L-1)$. The four neighbors (north, south, east, west) of grid point (i,j) are, $(i\pm1,j)$ and $(i,j\pm1)$.

- **Periodic boundary condition**: Used to simulate bulk systems without surface effects. It allows a wrap-around condition. For an $L \times L$ two-dimensional lattice, grid points $(L-1, j)$ and $(0, j)$ are east-west neighbors and $(i, L-1)$ and $(i, 0)$ are north-south neighbors.

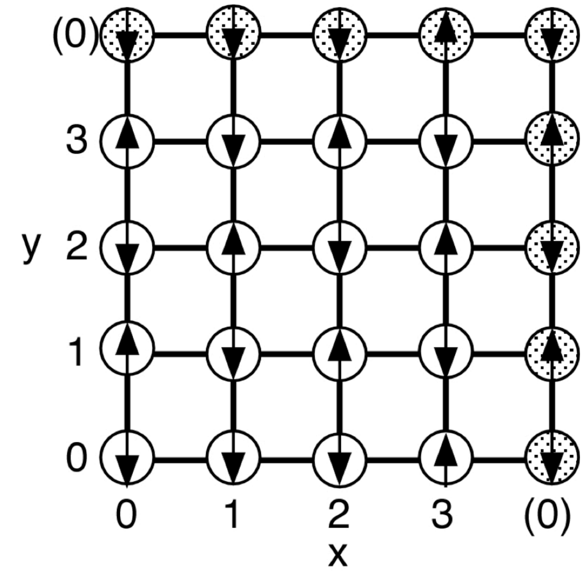


Figure: A spin configuration of the two-dimensional Ising model on a 4 × 4 lattice.

MC ALGORITHM FOR THE ISING MODEL

The following pseudocode performs a MC simulation for a 2-dimensional Ising model. The simulation consists of a series of single-spin updates. One grid point, (i, j), is flipped and the corresponding change in energy is calculated. The spin flip is accepted or rejected according to the Metropolis algorithm. In each MC step, one grid point on the lattice is randomly selected. Totally `maximum_step` steps are performed.

[1] D. W. Heermann and A. N. Burkitt, *Parallel Algorithms in Computational Science* (Springer-Verlag, Berlin, 1991).

Algorithm: MC simulation of a 2-dimensional Ising model

```
initialize the spins, s[i][j] (0 ≤ i,j ≤ L-1)
Sum_A = 0
for step =1 to maximum_step
  randomly pick a grid point, (i,j)
  compute the change in potential energy, δV, with a single spin flip, $s_{i,j} \to -s_{i,j}$
  if dV ≤ 0 accept the flip, $s_{i,j} \leftarrow -s_{i,j}$
  else if random() ≤ exp(-δV/$k_B$T) then // Here random() is in the range [0,1]
    accept the flip, $s_{i,j} \leftarrow -s_{i,j}$
  endif
  Sum_A = Sum_A + A($s^N$)
endfor
Average_A = Sum_A/maximum_step
```

Physical quantities we calculate include

Magnetization: $M = \sum_k s_k$;

Fluctuation of the magnetization: $Std\{M\} = \sqrt{\langle M^2 \rangle - \langle M \rangle^2}$.

The change in energy, $\delta V(s^N) = V(\ldots,s_k',\ldots) - V(\ldots,s_k,\ldots)$, associated with a single spin flip, $s_k \to s_k' = -s_k$, is calculated as follows:

$$\delta V = -J \sum_{l \in n.n.(k)} (s_k' - s_k) s_l - H(s_k' - s_k) \ ,$$

where $n.n.(k)$ denotes the set of nearest-neighbor grid points of the k-th grid point. Note that $s_k' - s_k = 2s_k'$ (2 for $-1 \to 1$; -2 for $1 \to -1$), so that

$$\delta V = -2s_k' \left(J \sum_{l \in n.n.(k)} s_l + H \right) .$$

For the 2D Ising model, there are only two possible values for s_k': ±1 and five values for $\sum_{l \in n.n.(k)} s_l$: 0, ±2, ±4. It is therefore convenient to set up a table for $\exp(-\delta V/k_B T)$ as an array, `exp_dV[2][5]` in the program. The look-up table eliminates many exponential evaluations, which require a significant CPU time, and consequently saves enormous computing.

The calculation of the magnetization is also simplified by keeping a running value M_{tot} that, each time a spin s_k is flipped to s_k', is replaced by $M_{tot} + 2s_k'$.

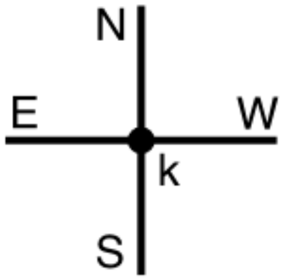


Table: possible central and neighbor spins.

Central spin, s_k'
–1
1

Neighbor spins				Sum
–1	–1	–1	–1	–4
–1	–1	–1	1	–2
–1	–1	1	1	0
–1	1	1	1	2
1	1	1	1	4

- **Initialization**: We choose a "cold start" in which all the spins are set to +1.

ISING MC CODE

Follow the explanation and data structures below to implement your Ising MC code.

-
```
int s[L][L];
#define L 20
```
  `s[i][j]` = 1|–1 represents up|down spin at lattice site, (i, j), where $0 \le i, j \le L{-}1$. $L = 20$ is the size of the lattice in each of the x and y directions.

- **Periodic boundary condition**: The four (west, east, south, north) neighbors of site (i, j) are (im, j), (ip, j), (i, jm), (i, jp), where

  $im = (i + L - 1)\ \%\ L$
  $ip = (i + 1)\ \%\ L$
  $jm = (j + L - 1)\ \%\ L$
  $jp = (j + 1)\ \%\ L$

  and % represents the modulo operator (in C notation).

- `double exp_dV[2][5];`

$$\text{exp\_}dV[k][l] = \exp\left(-\frac{\delta V}{k_B T}\right) = \exp\left(2s'\left[\frac{J}{k_B T}\sum_{s \in n.n.(s')} s + \frac{H}{k_B T}\right]\right),$$

  where the array indices and spin variables satisfy the following relations,

  $k = (1 + s')/2$ ($k = 0,1$; $s' = -1,1$)
  $l = (4 + S)/2$ ($l = 0,1,2,3,4$; $S = \Sigma_{\text{neighbor}(s')}\, s = -4, -2, 0, 2, 4$)

- **Ising model parameters**: There are two independent model parameters.

  `double JdivT` = J/k_BT: Exchange coupling in unit of temperature
  `double HdivT` = H/k_BT: Magnetic field in unit of temperature

- `double runM, sumM = 0, sumM2 = 0;`

  `runM` is the running value of the total magnetization, $runM = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} s[i][j]$.

  When the simulation is cold started, `s[i][j]` = 1 for all $(i\ ,j)$, `runM` is initialized to $1.0{*}L^2$. Every time a spin is flipped from $s \rightarrow s_{\mathrm{new}} = -s$, update `runM` += $2{\times}s_{\mathrm{new}}$.

  After each MC step (i.e., a spin-flip attempt that may or may not be accepted), update the accumulated values for the magnetization and its square:

  ```
  sumM += runM;
  sumM2 += runM*runM;
  ```

  (Even though these are integer values, they can become too big to be represented by a 32-bit integer. Therefore we use 64-bit double type. Note M^2 can be as large as $(20^2)^2$ = 160,000, and accumulated sum of 5,000 M^2 values is 0.8×10^9, close to the largest integer $\sim 2^{31} \sim 2 \times 10^9$.)

- `int Sta_step;`

  Number of MS steps to be performed. There will be `Sta_step` accumulated values of M and M^2.

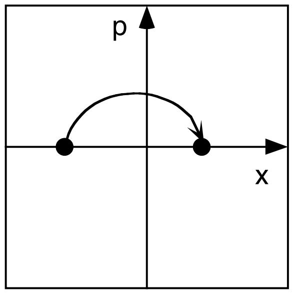
- `double avgM, sigM;`

  The mean magnetization and its standard deviation are given by

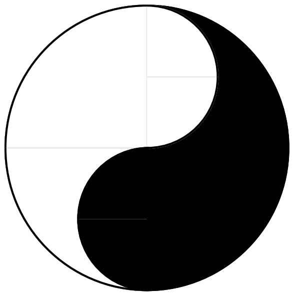
  $$avgM = sumM/Sta\_step\ ,$$
  $$sigM = \sqrt{sumM2/Sta\_step - avgM^2}\ .$$

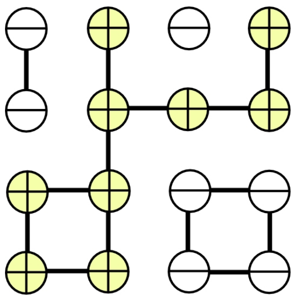
§2 Cluster MC Algorithm—Swendsen-Wang[2]

To achieve the desired Boltzmann distribution, $\exp(-V(x)/k_BT)$, the hybrid MC algorithm[3] uses the **extended phase space**, (x, p), as a vehicle to generate a Markov chain, $(x_1, x_2, ...)$, which efficiently samples a wide coordinate space, x, by allowing collective motions of the entire system. In cluster MC methods, a **dual representation** is introduced for the system: A state of a lattice spin system can be represented either as a collection of lattice spins or as a graph with edges connecting nearest-neighbor parallel spins. Clusters of connected spins are flipped at a time to realize collective motions.



Warping through an extended space.


Dual representation.


Graph representation of lattice spins.

A drawback of the hybrid MC method, however, arises from the limitation of the molecular dynamics (MD) method it utilizes. The MD discretization time unit, δt, must be on the order of the time scale of the fastest motion which exists in the system to achieve a reasonable acceptance ratio. However, global motions of the entire system are often characterized by much longer time than δt, and it requires many MD iterations to achieve such motions, resulting in enormous computing time.

In cluster MC methods, cluster flips help exploring a wide configuration space rapidly. The number of MC steps required before two states become uncorrelated is known as the **correlation time**, τ. (Note that before and after one spin flip, the two states are very well correlated.) The correlation time thus describes how long it takes to lose a memory of an old state. Near the critical temperature, T_c, at which the spin system transforms from a magnetic (at the low-temperature side) to a nonmagnetic (at the high-temperature side) states, the correlation time scales with the system size, L, as $\tau \sim L^z$. The exponent, z, known as the **dynamical critical exponent**, is 2.125 for the two-dimensional Ising model, if a standard single spin flips are used. For a large system, therefore, the correlation time becomes very long. This **critical slowing down** of the system dynamics causes computational inefficiency. An MC simulation must be performed much longer than τ to achieve a reasonable sampling of the configuration space. Therefore the computation time for the standard MC simulation scales as roughly quadratically as the problem size, $O(L^{2.125})$. The first cluster MC developed by Swendsen and Wang in 1987 achieves z = 0.35 for the same model, achieving a significant reduction in computing time. Later we will learn that by introducing a multilevel concept, the critical slowing down can be completely eliminated, $z \sim 0$. Namely, the multilevel algorithm is scalable!

POTTS MODEL

The Swendsen-Wang algorithm is most simply described for a Potts model with the energy function,

[2]R. H. Swendsen and J.-S. Wang, "Nonuniversal critical dynamics in Monte Carlo simulations," *Physical Review Letters* **58**, 86 (1987).
[3] B. Mehlig, D. W. Heermann, and B. M. Forrest, "Hybrid Monte Carlo method for condensed-matter systems," *Physical Review B* **45**, 679 (1992).

$$H = -J \sum_{(i,j)} \left(\delta_{\sigma(i),\sigma(j)} - 1 \right),$$

where the spins, σ(*i*), take on the values, 1, 2, ..., *q*. For a given spin configuration, σ = (σ(1), σ(2), ..., σ *L*)) (*L* is the number of lattice sites), the probability density function takes a value,

$$P(\sigma) = \exp\left(-H(\sigma)/k_B T\right)/Z ,$$

where the partition function,

$$Z = \sum_{\sigma} \exp\left(-H(\sigma)/k_B T\right),$$

consists of a summation over all possible spin configurations, σ. Below we consider a two-state Potts model for which *q* = 2. This model is equivalent to the Ising model except for a constant in the energy function.

THE SWENDSEN-WANG ALGORITHM

In the Swendsen-Wang algorithm, a transition, σ → σ', in a Markov chain is achieved in two phases like the double-buffer mode in computer graphics. In the first phase, the starting **spin configuration**, σ, is converted to a **bond configuration**, Γ, which is a set of bonds (or edges) connecting nearest-neighbor spin sites. This transition, σ → Γ, is achieved by the following algorithm:

Phase 1: Spin-to-bond transition

1. There are no bonds between anti-parallel nearest-neighbor spins.
2. A bond between parallel nearest-neighbor spins is created randomly with probability,

$$p = 1 - \exp\left(-J/k_B T\right).$$

In the second phase of the Swendsen-Wang algorithm, the resulting bond configuration at the first phase is converted to a new spin configuration, Γ → σ', as follows.

Phase 2: Bond-to-spin transition

1. Build clusters of connected spins by bonds.
2. Each cluster is randomly given either spin up or down with equal probability, 1/2.

Below, we prove that the joint transition probability,

$$P(\sigma' \mid \sigma) = \sum_{\Gamma} P(\sigma' \mid \Gamma) P(\Gamma \mid \sigma),$$

has the Boltzmann distribution as its unit eigenstate, i.e., the Boltzmann distribution is achieved as the asymptotic limit by successive applications of the transitions:

$$\frac{1}{Z}\exp\left(-H(\sigma')/k_B T\right) = \sum_{\sigma} P(\sigma' \mid \sigma) \frac{1}{Z}\exp\left(-H(\sigma)/k_B T\right).$$

CLUSTER IDENTIFICATION

Before proving the eigenstate property mentioned above, let us consider the cluster-identification step in the second phase of the algorithm. In the physics community, the most popular cluster-identification algorithm is the Hoshen-Kopelman algorithm.[4] Here we describe an alternative algorithm, i.e., the breadth-first search,[5] which is more familiar to computer science students.

We consider a two-dimensional lattice with size, NX×NY. A Vertex (spin site) has an integer ID, iv, in the range, [1,nv]. In addition, a vertex's position is encoded sequentially in the column-major fashion: For row ix and column iy, id = ix*NY+iy. **Periodic boundary conditions** are imposed. Each vertex is assigned an image index, m = (mx+2)*5+(my+2)+1, where mx and my denote which image the vertex belongs in the x and y directions, respectively.

We construct a cluster starting from a seed vertex, which is assigned the central image index m = 2*5 + 2 + 1 = 13. The cluster is grown by traversing all the connected vertices by the breadth-first fashion. Bonded neighbors are assigned image indices relative to the seed vertex.

It is often of interest to determine whether a cluster is percolating or not, i.e., whether the infinitely repeated periodic images are connected in one or more directions or they are all disconnected. In the breadth-first traverse, **percolation** is easily detected by an attempt to assign different indices to a single vertex.

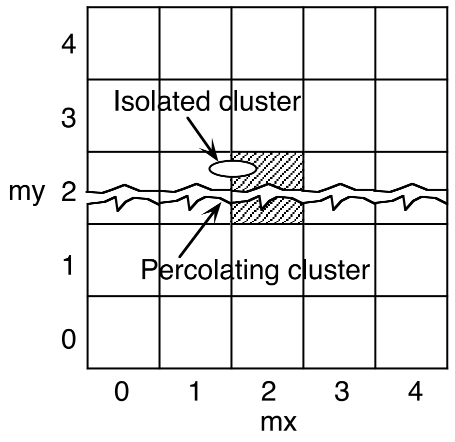
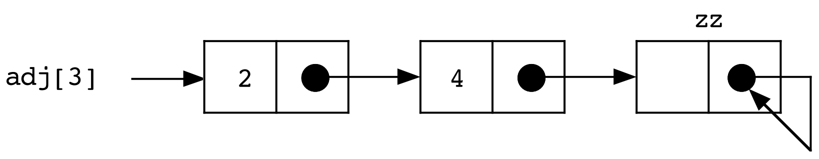


Figure: Periodic image indices with percolating and isolated clusters.

Data structures

```
#define NVMAX  10000 /* Maximally accommodated vertices */
int nv; /* Number of vertices */
struct vertex { /* Vertex data structure */
  int id;    /* Scalar site ID */
  int image; /* Which periodic image to belong to */
};
struct vertex vcl[NVMAX];
```

In the bond picture, the state of the spin system is represented as a graph: Nodes are spin sites and edges are bonds connecting neighboring spin sites. Adjacency lists are used to store the graph. For vertex id, adj[id] points to the first neighbor nodes in the list; a self-pointing node is used to signify the tail of the list.



[4]D. Stauffer, *Introduction to Percolation Theory* (Taylor & Francis, London, 1985).
[5]B. B. Lowekamp and J. C. Schug, "An efficient algorithm for nucleation studies of particles using the octree data structure," *Computer Physics Communications* **76**, 281 (1993).

```
struct node { /* Data structure for linked-list elements */
  int v;             /* Node value */
  struct node *next; /* Pointer to the next node */
};
struct node *adj[NVMAX];     /* Adjacency-list headers */
struct node *zz;             /* Tail node for linked lists */
zz = (struct node *) malloc(sizeof *zz);
zz->next = zz; /* Artificial tail node *zz points to itself */
```

A cluster has a sequential ID, `ip`, and is characterized by its size (= the number of member vertices) and whether it percolates or not. Furthermore, its member vertices are stored as a linked list, the header of which is `lsp[ip]`.

```
#define NPMAX  10000 /* Maximally accommodated clusters */
int np; /* Number of clusters */
struct cluster { /* Cluster data structure */
  int size;             /* Number of member vertices */
  int percolates;       /* Is the cluster percolates? */
};
struct cluster por[NPMAX];   /* Cluster array */
struct node *lsp[NPMAX];     /* Cluster-member-list headers */
```

Algorithm

The breadth-first search algorithm uses a **queue**, or a FIFO (first-in first-out) list, to store vertices while traversing a graph. Initially all the vertices are marked untouched, denoted by image index, 0. Then all the vertices are sequentially tested if it has already touched; if not, the vertex is "visited" to traverse all the connected vertices starting from it.

```
/*--------------------------------------------------------------------*/
void bfs() {
/*--------------------------------------------------------------------
Breadth-fist search to visit all the vertices to construct clusters as
each represented by a linked list whose headers are stored in lsp[].
--------------------------------------------------------------------*/
  int iv;
  /* Initialize a queue */
  queueinit();
  /* All the vertices are initialized to be untouched */
  for (iv=1; iv<=nv; iv++) vcl[iv].image = 0;
  /* Loop over vertices, visit if disconnected from previous trees */
  for (iv=1; iv<=nv; iv++)
    if (vcl[iv].image == 0)
      /* Traverse all the vertices connected to it if untouched */
      visit(iv);
      /* When exited from visit(), queue is emptied */
}
```

Each visit function adds a new cluster using the starting vertex as a seed; therefore `np` is incremented by one. The seed vertex is first enqueued in a queue. A main while loop, which continues until the queue becomes empty, dequeues a vertex and enqueus all its untouched neighbors. Each dequeued vertex is inserted into the current cluster. The percolation (conflicting image assignments) is checked in the while loop.

```
/*----------------------------------------------------------------------*/
void visit(int iv0) {
/*----------------------------------------------------------------------
Starting from vertex iv0, traverses all the vertices connected to it
by the breadth-first search.  Each call to visit() constructs a linked
list of all the member vertices of one cluster.  During the search,
whether the cluster percolates or not is determined.
----------------------------------------------------------------------*/
  struct node *t;
  int ix,iy,id,iv,jx,jy,jd,jv;
  int kx,ky,k,mx,my,m;
  /* Cluster initialization */
  /* Artificial header node for a "new" cluster is grounded */
  if (++np > NPMAX) printf("Number of pores overflowed\n");
  lsp[np] = zz;
  por[np].size = 0; /* Initialize the pore(cluster) size */
  por[np].percolates = 0; /* Initially, cluster doesn't percolate */
  /* Enqueue the seed vertex */
  put(iv0);
  /* The seed is chosen to be in the central image */
  vcl[iv0].image = 2*5 + 2 + 1;
  /* Continue until all the connected vertices are traversed */
  while (!queueempty()) {
    /* Extract the first vertex from the queue */
    id = vcl[iv = get()].id; /* Scalar relative(periodic) site ID */
    ix = id/NY;              /* Vector cell ID */
    iy = id%NY;
    /* Image index of vertex iv */
    kx = ((k = vcl[iv].image)-1)/25-2;
    ky = ((k-1)/5)%5-2;
    kz = (k-1)%5-2;
    /* Insert vertex iv at the top of the member list of cluster np */
    t = (struct node *) malloc(sizeof *t);
    t->v = iv;
    t->next = lsp[np];
    lsp[np] = t;
    /* Increment the cluster size */
    ++(por[np].size);
    /* Enqueue all the untouched adjacents jv of vertex iv */
    for (t=adj[iv]; t!=zz; t=t->next) {
      jv = t->v;
      jd = vcl[jv].id; /* Relative(periodic) scalar cell address */
      jx = jd/NY;      /* Vector cell address */
      jy = jd%NY;
      /* Image index of the neighbor void-cell(vertex) j */
      mx = kx;
      if (ix==0 && jx==NX-1)
        --mx;
      else if (ix==NX-1 && jx==0)
        ++mx;
      my = ky;
      if (iy==0 && jy==NY-1)
        --my;
      else if (iy==NY-1 && jy==0)
        ++my;
      m = (mx+2)*5+(my+2)+1; /* Scalar image index */
      /* If untouched, enqueue this neighbor */
      if (vcl[jv].image == 0) {
        put(jv);
        vcl[jv].image = m; /* Image has been assinged by propagation */
      }
      /* If already touched, check the percolation conflict */
      else if (vcl[jv].image != m)
        por[np].percolates = 1;
    } /* Endfor neighbor vertices */
  } /* Endwhile !queueempty */
}
```

Basic queue operations

```
#define NQ 1000              /* Maximum queue size */
int queue[NQ];               /* Queue */
int head, tail;              /* Head & tail indices for the queue */
void put(int v) { /* Enqueue function */
  queue[tail++] = v;
  if (tail >= NQ) tail = 0;
}
int get() { /* Dequeue function */
  int temp = queue[head++];
  if (head >= NQ) head = 0;
  return temp;
}
void queueinit() {head = 0; tail = 0;} /* Queue initialization */
int queueempty() {return head == tail;} /* Queue inquiry */
```

DUAL REPRESENTATION

Let us prove that the Swendsen-Wang algorithm converges to the Boltzmann distribution, $\exp[-H(\sigma)/k_BT]/Z$, at the asymptotic limit. To do so, the partition function of the two-state Potts model is rewritten as a summation over all possible bond configurations. Let us start from the representation of the partition function as a summation over spin configurations,

$$Z = \sum_{\sigma} \exp\left(-H(\sigma)/k_BT\right).$$

Consider now the interaction between spin sites *l* and *m*, and remove it from the energy function:

$$H_{(l,m)} = -J \sum_{(i,j)\neq(l,m)} \left(\delta_{\sigma(i),\sigma(j)} - 1\right).$$

Noting that

$$H = H_{l,m} + \begin{cases} 0 & \left(\sigma(l)=\sigma(m)\right) \\ J & \left(\sigma(l)\neq\sigma(m)\right) \end{cases},$$

we can rewrite *Z* as

$$\begin{aligned}
Z &= \sum_{\sigma} \left\{ \exp\left(-H_{(l,m)}/k_BT\right)\delta_{\sigma(l),\sigma(m)} + \exp\left[-\left(H_{(l,m)}+J\right)/k_BT\right]\left(1-\delta_{\sigma(l),\sigma(m)}\right)\right\} \\
&= \sum_{\sigma} \exp\left(-H_{(l,m)}/k_BT\right)\left[\delta_{\sigma(l),\sigma(m)} + \exp\left(-J/k_BT\right)\left(1-\delta_{\sigma(l),\sigma(m)}\right)\right] \\
&= \sum_{\sigma} \exp\left(-H_{(l,m)}/k_BT\right)\left\{\delta_{\sigma(l),\sigma(m)}\left[1-\exp\left(-J/k_BT\right)\right] + \exp\left(-J/k_BT\right)\right\} \\
&= \left[1-\exp\left(-J/k_BT\right)\right]\sum_{\sigma} \exp\left(-H_{(l,m)}/k_BT\right)\delta_{\sigma(l),\sigma(m)} + \exp\left(-J/k_BT\right)\sum_{\sigma}\exp\left(-H_{(l,m)}/k_BT\right) \\
&= pZ_{(l,m)}^{same} + (1-p)Z_{(l,m)}^{ind},
\end{aligned}$$

where

$$\begin{aligned}
Z_{(l,m)}^{same} &= \sum_{\sigma} \exp\left(-H_{(l,m)}/k_BT\right)\delta_{\sigma(l),\sigma(m)}, \\
Z_{(l,m)}^{ind} &= \sum_{\sigma} \exp\left(-H_{(l,m)}/k_BT\right),
\end{aligned}$$

and

$$0 \le p = 1 - \exp\left(-J / k_B T\right) \le 1 .$$

Now consider another pair, (k,n). By applying a similar procedure, we obtain

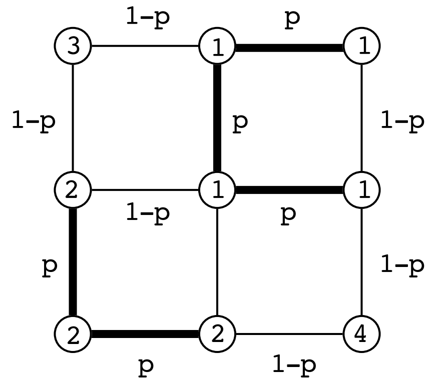
$$Z = p^2 Z^{same,same}_{(l,m),(k,n)} + p(1-p) Z^{same,ind}_{(l,m),(k,n)} + (1-p) p Z^{ind,same}_{(l,m),(k,n)} + (1-p)^2 Z^{ind,ind}_{(l,m),(k,n)} ,$$

where

$$Z^{same,same}_{(l,m),(k,n)} = \sum_{\sigma} \exp\left(-H_{(l,m),(k,n)} / k_B T\right) \delta_{\sigma(l),\sigma(m)} \delta_{\sigma(k),\sigma(n)} ,$$

etc. Recursively applying this procedure for the rest of the pairs, we end up with a binomial expression in which each term is a product of p's and (1-p)'s and a restricted sum of Z. The restricted Z sum now has no exponential factor (no pair left to retain in H) but only a product of delta functions.

To determine what the value of the restricted Z sum is, let us consider a specific term in a 3×3 lattice without periodic boundary conditions:



In this example, the total number of pairs, $N_P = 12$, and $N_B = 5$ of them are restricted to be the same spin (denoted by thick lines) and the rest 7 pairs are unrestricted. Let us define the restricted pairs to be **bonds**. Each bond contributes factor p and each nonbond contributes $1-p$. It is important to note that the delta functions in the restricted Z sum define clusters. This particular bond configuration defines $N_C = 4$ clusters (two of them are individual-spin cluster). For each cluster, there are two possible spin configurations in the 2-state Potts model. Therefore the value of the restricted Z sum is

$$p^{N_B} \left(1-p\right)^{N_P - N_B} 2^{N_C} = 4 p^5 \left(1-p\right)^7 .$$

Let Γ denote a bond configuration which specifies each of N_P pairs is either bonded or nonbonded. The partition function is then a summation over all the bond configurations,

$$Z = \sum_{\Gamma} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)} 2^{N_C(\Gamma)} .$$

Accordingly the probability for a particular bond configuration to occur in the equilibrium is

$$P\left(\Gamma\right) = \frac{1}{Z} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)} 2^{N_C(\Gamma)} .$$

DETAILED BALANCE

Let us consider the transition probability defined by the Swendsen-Wang algorithm. In the first phase, parallel pairs (the number of which is $N_{PP}(\sigma)$) in a spin configuration, σ, becomes a bond with probability p and a nonbond with probability $1-p$. The transition probability from the spin configuration σ to a particular bond configuration Γ is thus

$$P(\Gamma \mid \sigma) = \Delta_{\Gamma,\sigma} p^{N_B(\Gamma)} \left(1-p\right)^{N_{PP}(\sigma)-N_B(\Gamma)},$$

where $\Delta_{\Gamma,\sigma}$ is 1 if the bond configuration is compatible with the spin configuration, and is 0 otherwise. Summing the transition probability over all compatible bond configurations is equivalent to enumerating all combinations of the states—bond or nonbond—of all the parallel pairs. The resulting sum is just

$$\sum_{\Gamma} P(\Gamma \mid \sigma) = \left[p + \left(1-p\right)\right]^{N_{PP}(\sigma)} = 1,$$

i.e., $P(\Gamma \mid \sigma)$ defined above is a stochastic matrix.

To apply this transition probability matrix to the Boltzmann probability density for a spin configuration, note that

$$\frac{1}{Z}\exp\left(-H(\sigma)/k_B T\right) = \frac{1}{Z}\left(1-p\right)^{N_P - N_{PP}(\sigma)}.$$

This is because an antiparallel spin pair contributes a factor, $\exp(-J/k_B T) = 1-p$, and a parallel spin contributes 1 to the Boltzmann probability. By using this relation, we can derive

$$\begin{aligned}\sum_{\sigma} P(\Gamma \mid \sigma) P(\sigma) &= \sum_{\sigma} \Delta_{\Gamma,\sigma} p^{N_B(\Gamma)} \left(1-p\right)^{N_{PP}(\sigma)-N_B(\Gamma)} \frac{1}{Z}\exp\left(-H(\sigma)/k_B T\right) \\ &= \sum_{\sigma} \Delta_{\Gamma,\sigma} p^{N_B(\Gamma)} \left(1-p\right)^{N_{PP}(\sigma)-N_B(\Gamma)} \frac{1}{Z}\left(1-p\right)^{N_P - N_{PP}(\sigma)} \\ &= \frac{1}{Z}\sum_{\sigma} \Delta_{\Gamma,\sigma} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)}.\end{aligned}$$

Counting the number of all compatible spin configurations to a given bond configuration gives a factor $2^{N_C(\Gamma)}$ (each cluster can be spin up or down). Consequently the spin-configuration sum above gives the relation,

$$\sum_{\sigma} P(\Gamma \mid \sigma) P(\sigma) = \frac{1}{Z} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)} 2^{N_C(\Gamma)} = P\left(\Gamma\right),$$

i.e., the phase 1 of the Swendsen-Wang algorithm indeed generates the equilibrium probability density for bond configuration.

In the second phase of the Swendsen-Wang algorithm, each cluster is independently assigned either spin up or down. Therefore, the resulting probability density is

$$P(\sigma') = \sum_{\Gamma} P(\sigma' \mid \Gamma) P(\Gamma),$$

where the transition probability matrix chooses any of such spin assignments with equal probability,

$$P(\sigma' \mid \Gamma) = \Delta_{\sigma',\Gamma} \left(\frac{1}{2}\right)^{N_C(\Gamma)}.$$

By substituting the analytic expressions, we obtain

$$\begin{aligned}P(\sigma') &= \sum_{\Gamma} P(\sigma' \mid \Gamma) P(\Gamma) \\ &= \sum_{\Gamma} \Delta_{\sigma',\Gamma} \left(\frac{1}{2}\right)^{N_C(\Gamma)} \frac{1}{Z} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)} 2^{N_C(\Gamma)} = \frac{1}{Z} \sum_{\Gamma} \Delta_{\sigma',\Gamma} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)}.\end{aligned}$$

For a given spin configuration, σ', the above sum over all compatible bond configurations gives

$$\begin{aligned}\frac{1}{Z} \sum_{\Gamma} \Delta_{\sigma',\Gamma} p^{N_B(\Gamma)} \left(1-p\right)^{N_P - N_B(\Gamma)} &= \sum_{\Gamma} \Delta_{\sigma',\Gamma} p^{N_B(\Gamma)} \left(1-p\right)^{N_{PP}(\sigma') - N_B(\Gamma)} \frac{1}{Z} \left(1-p\right)^{N_P - N_{PP}(\sigma')} \\ &= \left[\sum_{\Gamma} \Delta_{\sigma',\Gamma} p^{N_B(\Gamma)} \left(1-p\right)^{N_{PP}(\sigma') - N_B(\Gamma)}\right] \frac{1}{Z} \exp\left(-\frac{H(\sigma')}{k_B T}\right) \\ &= \frac{1}{Z} \exp\left(-\frac{H(\sigma')}{k_B T}\right) = P\left(\sigma'\right).\end{aligned}$$

In summary for the Boltzmann probability, $P(\sigma)$,we have shown that

$$P(\sigma') = \sum_{\Gamma} \sum_{\sigma} P(\sigma' \mid \Gamma) P(\Gamma \mid \sigma) P\left(\sigma\right).$$

Therefore, the Boltzmann distribution is the eigenvalue of the dual transition matrix,

$$P\left(\sigma \mid \sigma'\right) = \sum_{\Gamma} P(\sigma' \mid \Gamma) P(\Gamma \mid \sigma),$$

which describes the transition probability from σ to σ′.

§3 Cluster MC Algorithm—Wolff[6]

Another cluster algorithm for lattice models has been developed by Ulli Wolff. The basic idea is again to look for clusters of similarly oriented spins and then flip them in entirety. We can start from a seed spin and look at its neighbor spins to see if any of them are pointing the same direction. Then we can look at the neighbors of those neighbors, and so on. This procedure can be implemented by using a last in first out (LIFO) stack.

How many of the spins that point to the same direction as the seed spin depends on the temperature. At high temperatures, the spins tend to be uncorrelated with their neighbors and accordingly the neighbor spins are added to the cluster with a low probability. To satisfy the detailed balance, the sticking probability, P_{add}, is shown to be $1 - \exp(-2J/k_BT)$ for the Ising model without external magnetic field (see below).

Algorithm: one step of Wolff's cluster-flip[7]

1. Choose a seed spin at random from the lattice.
2. Look in turn at each of the neighbors of that spin. If they are pointing in the same direction as the seed spin, add them to the cluster with probability $P_{\mathrm{add}} = 1 - \exp(-2J/k_BT)$.
3. For each spin that was added in the last step, examine each of its neighbors to find the ones that are pointing in the same direction and add each of them to the same probability P_{add}. (If some of the neighbors are already cluster members, they do not need to be added again. For spins that have been considered for addition but rejected before, they get another chance to be added to the cluster at this step.) Repeat this step as many times as necessary until there are no spins left in the cluster whose neighbors have not been considered for inclusion in the cluster.
4. Flip the cluster.

SAMPLE PROGRAM: `wolff.c`

Data structures

```
#define N (L*L) /* Total number of spins */
double JdivT;   /* Ising model parameters */
int s[L][L];    /* Lattice of spins */
double padd;    /* 1 - exp(-2*J/T) */
int stack[N];   /* LIFO stack */
int sp;         /* Stack pointer */
```

One cluster-flip step

```
/* Put a random seed spin site onto a stack */
  i = rand()%L; j = rand()%L;
  stack[0] = i*L + j;
  sp =1;
```

[6]U. Wolff, "Collective Monte Carlo updating for spin systems," *Physical Review Letters* **62**, 361 (1989).
[7]M.E.J. Newman and G.T. Barkema, *Monte Carlo Methods in Statistical Physics* (Clarendon Press, Oxford, 1999).

```
  /* Flip the seed and remember the old & new spins */
  oldspin = s[i][j]; newspin = -s[i][j];
  s[i][j] = newspin;

  while (sp) {
    /* Pop a site off the stack */
    current = stack[--sp];
    i = current/L;
    j = current%L;

    /* Check the neighbors */
    if ((nn=i+1) >= L) nn -= L; /* South neighbor */
    if (s[nn][j] == oldspin)
      if (rand()/(double)RAND_MAX < padd) {
        stack[sp++] = nn*L + j;
        s[nn][j] = newspin;
      }

    if ((nn=i-1) < 0) nn += L; /* North neighbor */
    if (s[nn][j] == oldspin)
      if (rand()/(double)RAND_MAX < padd) {
        stack[sp++] = nn*L + j;
        s[nn][j] = newspin;
      }

    if ((nn=j+1) >= L) nn -= L; /* East neighbor */
    if (s[i][nn] == oldspin)
      if (rand()/(double)RAND_MAX < padd) {
        stack[sp++] = i*L + nn;
        s[i][nn] = newspin;
      }

    if ((nn=j-1) < 0) nn += L; /* West neighbor */
    if (s[i][nn] == oldspin)
      if (rand()/(double)RAND_MAX < padd) {
        stack[sp++] = i*L + nn;
        s[i][nn] = newspin;
      }
  } /* End while stack is not empty */
```

Sample run

The following graph compares the conventional lattice MC algorithm in §1 and the Wolff algorithm for a 20 × 20 Ising lattice under zero magnetic field. The inverse temperature is, $J/k_BT = 0.5$.

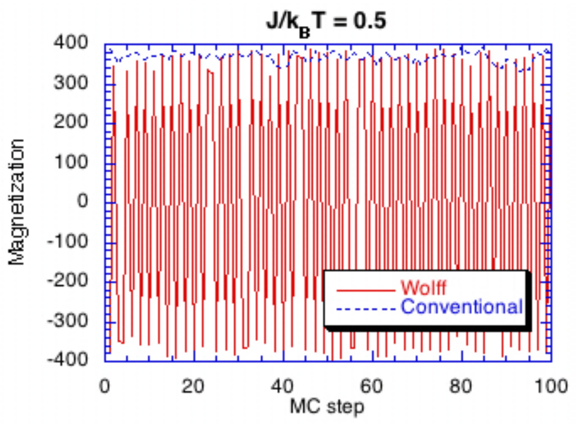


Figure: Magnetization as a function of MC steps for the conventional algorithm in §1 and Wolff's algorithm.

With the conventional algorithm, the magnetization is nearly unchanged from the initial value. (The system is stuck in a local minimum.) On the other hand, the Wolff algorithm explores both positive and negative magnetizations equally. If we wait long enough, however, even the conventional algorithm finds the negative magnetization; it just takes too many MC steps.

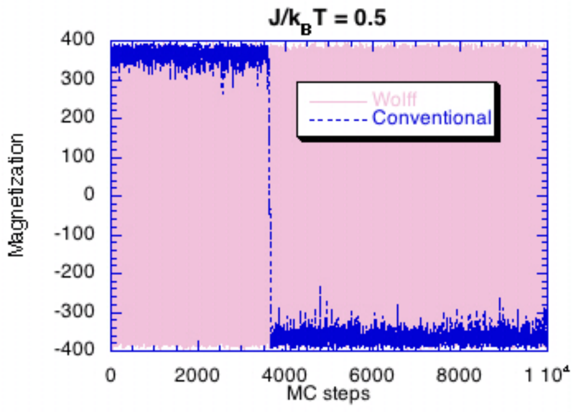


Figure: The same as above for 10,000 MC steps.

DETAILED BALANCE CONDITION

Let us prove that to satisfy the detailed balance condition P_{add} must be $1 - \exp(-2J/k_BT)$. Consider two states of the system, μ and ν, that differ from one another by the flipping of a single cluster of similarly oriented spins. In each of the two states, some of the spins just outside the cluster are pointing the same way as the spins in the cluster. The bonds between these spins and the ones in the cluster have to be broken when the cluster is flipped.

Consider now a move from μ to ν. Suppose that, for a forward move from μ to ν, there are m bonds that have to be broken. These broken bonds represent pairs of similarly oriented spins that were not added to the cluster by the algorithm. The probability of not adding such a spin is $1 - P_{\text{add}}$ and the probability of not adding all of them, which is proportional to the selection probability for the forward move, is $(1 - P_{\text{add}})^m$. If there are n bonds that need to be broken in the reverse move, ν to μ, then the probability of doing it will be $(1 - P_{\text{add}})^n$. The detailed balance condition is stated as

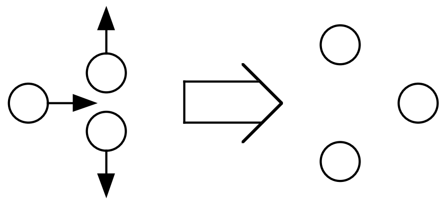
$$(1 - P_{\text{add}})^m\, P(\mu) = (1 - P_{\text{add}})^n\, P(\nu),$$

where $P(\mu)$ and $P(\nu)$ are the equilibrium probabilities of states μ and ν, respectively. Equivalently,

$$(1 - P_{\text{add}})^{m-n} = P(\nu)/P(\mu).$$

Now let us consider the energetics. For each of m bonds that are broken in going from μ to ν, the energy changes by $+2J$. For each of n bonds that are made in the same transition is $-2J$. The energy difference between the two states is thus $E_\nu - E_\mu = 2J(m - n)$. The equilibrium probability of the two states thus satisfy $P(\nu)/P(\mu) = \exp(-[E_\nu - E_\mu]/k_BT) = \exp(-2J[m-n]/k_BT)$, which is the right-hand side of the equality. With $P_{\text{add}} = 1 - \exp(-2J/k_BT)$, the left-hand side is $(1 - P_{\text{add}})^{m/n} = \exp(-2J[m-n]/k_BT)$ so that the detailed balance condition is satisfied.

§4. Hybrid MC Algorithm[8]



We will look at another advanced MC sampling technique to explore a wide search space with reasonable computing time.

When we perform an MC simulation on particles (e.g., Lennard-Jones atoms), "suggested" moves are usually those in which only a single particle is randomly displaced at a time. This excludes "collective" movements, an example of which is that two blocking particles give a way to a third particle to let it pass.

Molecular dynamics (MD) simulation, on the other hand, deals with the dynamics of an entire particle system by integrating the Newton's second law of motion. Here all the particles move simultaneously, so that collective motions are included. However, if significant events occur only infrequently, MD simulation cannot sample them. For example, a crack in your car's windshield may propagate at a rate of 1 meter/year (~ 10^8 atomic bonds are broken per 3×10^{22} femtoseconds, or using a typical discretization time unit of 1 femtosecond, in 3×10^{22} MD iterations.) That is one bond-breakage event occurs at every 300 trillion MD steps! In MC simulation, we at least have leverage on sampling methods to tackle such long-time phenomena.

Hybrid MC simulation combines the merits of both: MC's ability to generate infrequent events; and MD's ability to incorporate collective particle movements.

MOLECULAR DYNAMICS[9]

In MD simulation, the state of an N-particle system in two spatial dimension is specified by a point in the 4N-dimensional space: $\{(\vec{r}_i,\vec{p}_i)=(x_i,y_i,p_{xi},p_{yi})\,|\,i=1,...,N\}$, where $\vec{r}_i$ and $\vec{p}_i$ are the position and momentum of the i-th particle. The time evolution of the system is governed by

$$\begin{cases} \dfrac{d\vec{r}_i}{dt}=\dfrac{\vec{p}_i}{m_i} \\ \dfrac{d\vec{p}_i}{dt}=\vec{F}_i \end{cases}$$

where m_i is the mass of the i-th particle and

$$\vec{F}_i=-\frac{\partial V}{\partial \vec{r}_i}$$

is the interatomic force acting on the i-th particle. Here V is the interatomic potential energy as defined before. The velocity of the i-th particle is related to its momentum via a relation, $\vec{v}_i=\vec{p}_i/m_i$. The above equations are combined to produce Newton's second law of motion,

$$m_i\frac{d^2\vec{r}_i}{dt^2}=\vec{F}_i\,.$$

Let me emphasize that a MD program must have 4N variables to keep track of its state instead of 2N, as is the case in an MC program.

[8] B. Mehlig, D. W. Heermann, and B. M. Forrest, "Hybrid Monte Carlo method for condensed-matter systems," *Physical Review B* **45**, 679 (1992).
[9] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford Univ. Press, Oxford, 1987) p. 71.

In MD simulation, the time variable must be discretized so that the physical system is sampled at times $t_n = n\,\Delta t$ (n = 1, 2, 3, ...). The most common discretized form is

$$m_i \frac{\vec{r}_i(t_{n+1}) - 2\vec{r}_i(t_n) + \vec{r}_i(t_{n-1})}{\Delta t^2} = \vec{F}_i\left(\vec{r}^N(t_n)\right).$$

In order to write a simulation program, solutions to discretized algebraic models must be translated to a sequence of computer instructions based on some numerical algorithms. For example, the velocity-Verlet algorithm translates the above equation into iterated operations of the following time-stepping procedures:

```
Velocity-Verlet Algorithm
Compute the initial forces, $\vec{F}_i(t_0)$, as a function of the initial particle positions, $\vec{r}^N(t_0)$
for $n$ = 1 to $Max\_step$
    Update the velocities, $\vec{v}_i \leftarrow \vec{v}_i(t_{n-1}) + \vec{F}_i(t_{n-1})\Delta t / 2m_i$, of the particles
    Obtain the new particle positions, $\vec{r}_i(t_n) \leftarrow \vec{r}_i(t_{n-1}) + \vec{v}_i \Delta t$
    Compute the new forces, $\vec{F}_i(t_n)$, as a function of $\vec{r}^N(t_n)$
    Obtain the new velocities, $\vec{v}_i(t_n) \leftarrow \vec{v}_i + \vec{F}_i(t_n)\Delta t / 2m_i$
endfor
```

The velocity-Verlet integrator possesses several crucial properties, which are *required* to implement hybrid MD simulation.

1. Phase-Space Volume Conservation

Assume that we have a set of initial configuration points (say 100 slightly different initial configurations) around $(\vec{r}^N, \vec{p}^N)$ in the 4*N*-dimensional space in a small region of 4*N*-dimensional volume, $d\vec{r}^N d\vec{p}^N$. Let apply the above Velocity-Verlet algorithm for each initial configuration for, say, 1,000 MD steps. Each configuration point is mapped to a new configurational point according to

$$(\vec{r}^N, \vec{p}^N) \rightarrow g(\vec{r}^N, \vec{p}^N) = (\vec{r}'^N, \vec{p}'^N)$$

where g is a mapping from $\mathbb{R}^{4N}$ to $\mathbb{R}^{4N}$ defined by the above algorithm. The question is: What is the 4*N*-dimensional volume enclosing the mapped 100 points?

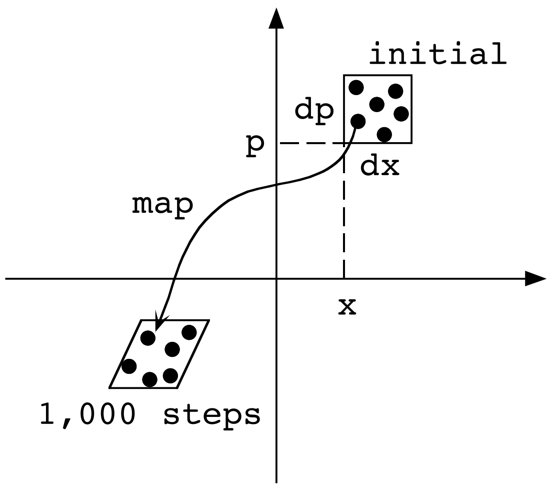


Figure: Volume conservation.

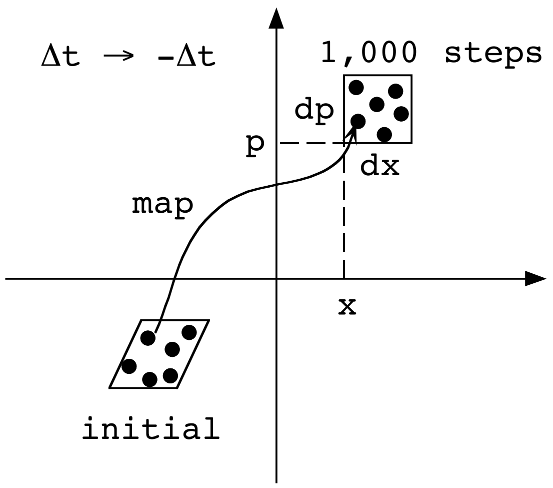


Figure: Time reversibility.

The answer is: It is the same as the initial volume, $d\vec{r}^N d\vec{p}^N$. The volume element may deform as time progresses, but the volume is invariant. This volume conservation is a property satisfied by the original differential equation, but fails for many discretization algorithms. The velocity-Verlet algorithm is one of many discretization algorithms that satisfy this property.

2. Time Reversibility

Let's consider the final configuration after 1,000 MD steps in the above example. Now modify the velocity-Verlet algorithm by replacing every occurrence of Δt by $-\Delta t$ (time reversal—we will play back the simulation backwards). After 1,000 steps of this time-reversed run, does the system come back to the original point? For the velocity-Verlet algorithm, the answer is yes, i.e., the algorithm is time reversible.

3. Energy "Non"-conservation

The original Newton's equation further conserves the total energy (K and V below are called the kinetic and potential energies, respectively),

$$H = K + V = \sum_{i=1}^{N} \frac{m_i}{2} \left|\vec{v}_i\right|^2 + V\left(\vec{r}^N\right).$$

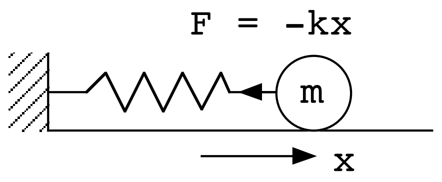
Unfortunately, this **energy conservation** is not satisfied by the velocity-Verlet algorithm. In fact, any commonly used discretization scheme does not conserve the total energy. This "violation of the energy conservation" actually does GOOD when we implement a hybrid MC. Generally, a larger discrete time unit, Δt, results in a larger variation of the total energy.

Project 1 (Easy but important)

Consider one particle in the 1-dimensional system,

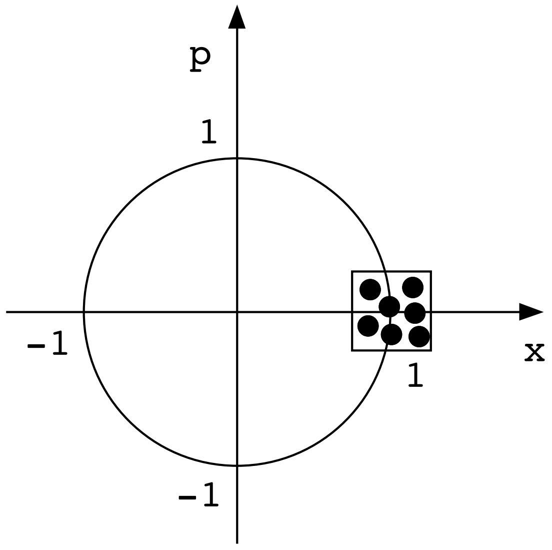
$$H = \frac{m}{2} v^2 + \frac{K}{2} x^2 ,$$

which represents the motion of a particle of mass m connected to a spring of the spring constant of K. The particle position x is a real number in the range, $[-\infty,\infty]$. The force, $F = -dV/dx = -Kx$.



Let's choose $m = K = 1$.

1. Generate 100 random points (x,p) in the 2-dimensional space around (1,0) (stretched spring) and within the square of edge length 0.1. Make a 2D plot showing all the initial points.



2. Run an MD simulation for 5,000 MD steps for each of the above set of 100 initial configurations, using the velocity-Verlet algorithm with $\Delta t = 0.01$. Plot the distribution of the final configurations. Would you say the area that these points occupy is conserved?
3. For the final configuration corresponding to the initial configuration, (1, 0), perform a time-reversal simulation for 5,000 steps. Does it come back to the original configuration?
4. For the initial configuration, (1, 0), plot the total energy as a function of time. Use Δt = 0.0001, 0.001, 0.01, 0.1, and 1. In each case, the maximum time, $Max\_step \times \Delta t$ must be 50.0.
5. Answer the above questions, 1-4, by replacing the velocity-Verlet algorithm by the explicit Euler algorithm shown below.

Explicit Euler Algorithm

```
Set up an initial configuration $(x(t_0), v(t_0))$
for $n = 1$ to $Max\_step$
    Compute the force, $F\left(t_{n-1}\right) = -Kx_{n-1}$
    Update the position, $x(t_n) \leftarrow x(t_{n-1}) + v(t_{n-1})\Delta t$
    Update the velocity, $v(t_n) \leftarrow v(t_{n-1}) + F(t_{n-1})\Delta t / m$
endfor
```

HYBRID MC ALGORITHM

Let us consider a mapping, $g^{t,\delta t} : (\vec{r}^N, \vec{v}^N) \rightarrow (\vec{r}'^N, \vec{v}'^N)$, which is defined in the velocity-Verlet MD integration for a fixed time t with discretization time unit of Δt. This mapping can be used to generate attempts (or suggested moves) in a Markov chain for particle positions. Due to discretization error, the total energy, H, changes during the MD integration, and therefore we can apply the Metropolis rejection/acceptance algorithm to achieve the desired equilibrium probability density as the asymptotic limit. At the beginning of the MD integration, velocities are initialized with the Maxwell-Boltzmann (or normal) distribution with temperature T (this is used later to prove the detailed balance condition). The major difference of the current scheme from the standard MC is that each attempt is global since it affects the entire system.

Hybrid MC Algorithm

```
for step_MC = 1 to Max_step_MC
  Generate normal velocity distributions with temperature, T
  Compute the total energy, $H_{init}$, and forces, for the initial state
  for step_MD = 1 ro Max_step_MD
    Velocity-Verlet update of coordinates and velocities for $\delta t$
  endfor
  Compute the final energy, $H_{final}$
  if $\Delta H = H_{final} - H_{init} < 0$ then
    Accept the state change, $x_{init} \leftarrow x_{final}$
  else if rand()/RAND_MAX < $\exp(-\Delta H/k_BT)$ then
    Accept the state change, $x_{init} \leftarrow x_{final}$
  endif
endfor
```

As an application of the hybrid MC algorithm, let us consider charged particles on a circular disk of radius *R*. In order to make the force calculation easier, we modify the problem slightly: The interatomic potential is extended to have a confinement term in the radial direction; particle positions are not confined within radius *R* any more.

$$V(\vec{r}^N) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{q^2}{|\vec{r}_i - \vec{r}_j|} + \sum_{i=1}^{N} \frac{K}{2}\left(|\vec{r}_i| - R\right)^2 \Theta(|\vec{r}_i| - R).$$

Here the step function $\Theta(x)$ is 1 if $x \geq 0$, and 1 otherwise. The force acting on the *k*-th particle is evaluated as

$$\begin{aligned}\vec{F}_k &= -\frac{\partial}{\partial \vec{r}_k} V \\ &= \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left\{ \frac{q^2}{|\vec{r}_i - \vec{r}_j|^3} \left(\vec{r}_i - \vec{r}_j\right) \right\} \left(\delta_{k,i} - \delta_{k,j}\right) - K\left(|\vec{r}_i| - R\right) \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|} \Theta(|\vec{r}_i| - R).\end{aligned}$$

In deriving the analytical expression for the force, we have used the following relation, where $\vec{r} = (x, y)$:

$$\frac{\partial}{\partial \vec{r}} f(r) = \frac{\partial r}{\partial \vec{r}} \frac{df}{dr} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right) \sqrt{x^2 + y^2} \frac{df}{dr} = \frac{(x, y)}{\sqrt{x^2 + y^2}} \frac{df}{dr} = \frac{\vec{r}}{|\vec{r}|} \frac{df}{dr}.$$

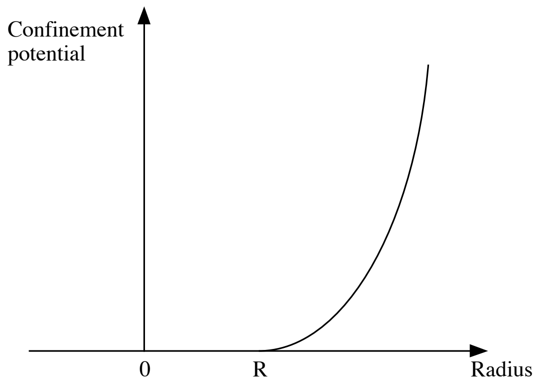


Figure: Radial confinement potential.

Project 2

Implement the hybrid MC algorithm for the charged particles on a disk.

DETAILED BALANCE CONDITION

In order to generate a correct probability density as its asymptotic limit, the Markov chain must satisfy the detailed balance condition. This condition is satisfied if the MD integrator is time reversible and volume conserving as in the case of the velocity-Verlet integrator. To show this, let us start from the transition probability in the coordinate space,

$$p_M(x \to x') = \int dp \int dp' p_G(p) \delta\left[(x',p') - g^{t,\delta t}(x,p)\right] p_A\left((x,p) \to (x',p')\right).$$

In this expression, the initial momenta are drawn from a Gaussian distribution,

$$p_G(p) \propto \exp\left(-\sum_{i=1}^{N} \frac{|\vec{p}_i|^2}{2m_i k_B T}\right),$$

and the acceptance probability is

$$p_A\left((x,p) \to (x',p')\right) = \min\left\{1, \exp(-\delta H / k_B T)\right\},$$

where

$$\delta H = H(x',p') - H(x,p)$$

is nonzero due to the **discretization error in energy**. The delta function, $\delta[(x',p') - g^{t,\delta t}(x,p)]$, tells us that the transition occurs only to (x',p') which is connected to the initial phase space point, (x, p), through the velocity-Verlet mapping, $g^{t,\delta t}$. Note that momenta are used only as a mean to define a transition probability in the coordinate space.

Now let us consider the probability,

$$\exp\left(-V(x)/k_B T\right) p_M(x \to x') dx dx',$$

that the system is initially within a volume element, dx, around the position x, according to the equilibrium probability density, and that a transition occurs to a volume element, dx', around x'. By substituting the definition of the transition probability, we get

$$\begin{aligned}&\exp\left(-V(x)/k_B T\right) p_M(x \to x') dx dx' \\ &= \exp\left(-V(x)/k_B T\right) dx \int dp \int dp' dx' p_G(p) \delta\left[(x',p') - g^{t,\delta t}(x,p)\right] p_A\left((x,p) \to (x',p')\right).\end{aligned}$$

Due to the delta function, the integral is nonzero only when $(x',p') = g^{t,\delta t}(x,p)$. In other words, due to the deterministic, one-to-one mapping, (x',p') and (x,p) are uniquely related to each other. By evaluating the x' and p' integration, we get

$$\begin{aligned}&\exp\left(-V(x)/k_B T\right) p_M(x \to x') dx dx' \\ &= \exp\left(-V(x)/k_B T\right) dx \int dp p_G(p) p_A\left((x,p) \to (x',p')\right),\end{aligned}$$

where (x',p') and (x,p) are not independent and are related via $(x',p') = g^{t,\delta t}(x,p)$.

Substituting the definition of the acceptance probability, $p_A((x,p) \to (x',p'))$,

$$\begin{aligned}&\exp\left(-V(x)/k_BT\right)p_M(x\to x')dxdx'\\&=\int dp p_G(p)\exp\left(-V(x)/k_BT\right)\min\left\{1,\exp(-\delta H/k_BT)\right\}dx\\&=\int dp\exp\left(-H(x,p)/k_BT\right)\min\left\{1,\exp(-\delta H/k_BT)\right\}dx,\end{aligned}$$

where we have used the relation, $H(x,p) = K(p) + V(x)$. Noting the identity,

$$\begin{aligned}&\exp\left(-H(x,p)/k_BT\right)\min\left\{1,\exp(-\delta H/k_BT)\right\}\\&=\min\left\{\exp\left(-H(x,p)/k_BT\right),\exp\left(-H(x',p')/k_BT\right)\right\}\\&=\exp\left(-H(x',p')/k_BT\right)\min\left\{1,\exp(\delta H/k_BT)\right\},\end{aligned}$$

we can derive that

$$\begin{aligned}&\exp\left(-V(x)/k_BT\right)p_M(x\to x')dxdx'\\&=\int dp\exp\left(-H(x',p')/k_BT\right)\min\left\{1,\exp(\delta H/k_BT)\right\}dx\\&=\int dp p_G(p')\exp\left(-V(x')/k_BT\right)\min\left\{1,\exp(\delta H/k_BT)\right\}dx.\end{aligned}$$

If the MD integrator is **phase-space volume conserving**, we can change the integration variables from (x,p) to (x',p') with the unit volume scaling factor, i.e., $dxdp = |\partial(x,p)/\partial(x',p')|dx'dp' = dx'dp'$. Therefore,

$$\begin{aligned}&\exp\left(-V(x)/k_BT\right)p_M(x\to x')dxdx'\\&=\int dp' p_G(p')\exp\left(-V(x')/k_BT\right)\min\left\{1,\exp(\delta H/k_BT)\right\}dx'.\end{aligned}$$

Note that now the roles of (x',p') and (x,p) are interchanged: The right-hand side is the probability that the system is initially at x' with the initial momenta p' are drawn from a normal distribution, $p_G(p')$, at temperature T. The acceptance probability, $\min\{1,\exp(\delta H/k_BT)\}$, is a standard Metropolis one for a transition, $(x',p') \to (x,p)$. (Note that $\delta H = H(x,p) - H(x',p')$.) Now if the MD integrator is **time reversible**, (x',p') comes back to $(x,p) = g^{-t,-\delta t}(x',p')$ by reversing time. Therefore the above expression is equivalent to

$$\exp\left(-V(x')/k_BT\right)dx'\int dp' p_G(p')p_A\left((x',p')\to(x,p)\right),$$

which in turn is

$$\exp\left(-V(x')/k_BT\right)p_M(x'\to x)dx'dx.$$

We have therefore proven the detailed balance condition,

$$\exp\left(-V(x)/k_BT\right)p_M(x\to x')dxdx' = \exp\left(-V(x')/k_BT\right)p_M(x'\to x)dx'dx.$$

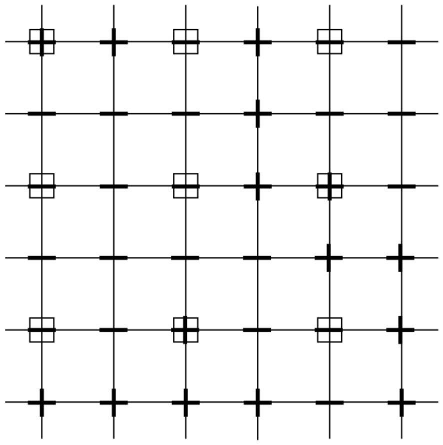
§5 Other Advanced MC Algorithms

In this section, we will survey several advanced MC algorithms which feature enhanced sampling of the solution space.

MULTIGRID MC[10]

As we have learned in the previous sections, the cluster algorithm does not completely remove the critical slowing down: the dynamical critical exponent is $z = 0.35$ for the two-dimensional Ising model, i.e., the computation still scales superlinearly with the system size.

The essence of the Swendsen-Wang and Wolff algorithms is "clustering" many parallel spins to allow for their simultaneous flips. The multigrid MC algorithm developed by Kandel, Domany, and Brandt in 1988 introduces a hierarchy of coarsening levels to generate successive problems with few degrees of freedom. Each coarsening procedure is based on stochastic "**freezing**" (keeping relative spin orientations during the next Markov transition) of interaction with probability p and "**deleting**" (no restriction on the transition) it with probability $1 - p$ (similar to a procedure used in the Swendsen-Wang algorithm).



Coarsening

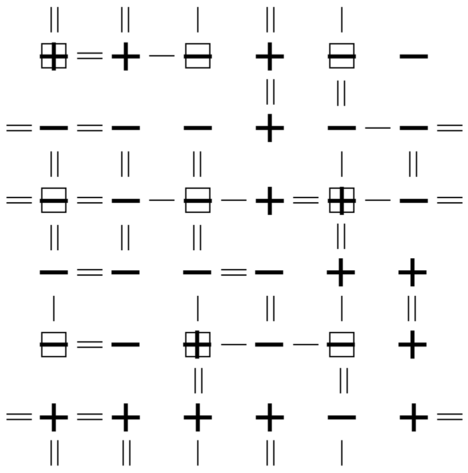
In the multigrid MC algorithm, a coarsening procedure is applied to a spin lattice recursively. Each coarsening procedure, applied to a lattice with L^2 spins, produces a coarser lattice with $(L/2)^2$ spins. This is achieved by decimating each other rows and columns. The figure below shows a typical spin configuration at a fine level. The boxed spins constitute the coarse lattice at the next level.

A coarsening procedure starts by applying a modified Swendsen-Wang stochastic blocking procedure:

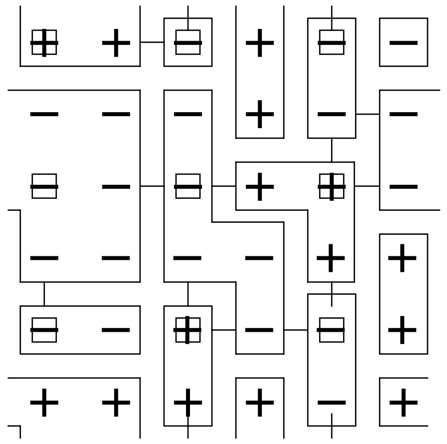
```
for all the bonds (i,j) (visit them "sequentially")
  if both i and j have been already frozen to different coarse spins then
    Leave the bond (i,j) "alive" (i.e., don't freeze or delete)
  else if at least one of the 2 sites has not been frozen to a coarse spin
    if $s_i \neq s_j$ then /* antiparallel spins */
      delete the bond (i.e., no interaction will be considered for this pair)
    else /* parallel spins */
      if rand()/RAND_MAX < $p=1-\exp(-J_{ij}(1+s_is_j)/k_BT)$ then /* freezing */
        freeze the bond (relative spin orientation will be fixed)
      else /* deleting */
        delete the bond
      endif
    endif
  else (both i and j have been frozen to the same coarse spin)
    the bond is frozen
  endif
endfor
```

The following figure shows a typical outcome from this procedure. In the figure, a double line denotes frozen spins and a single line denotes a live interaction. (Note that periodic boundary conditions are applied.)

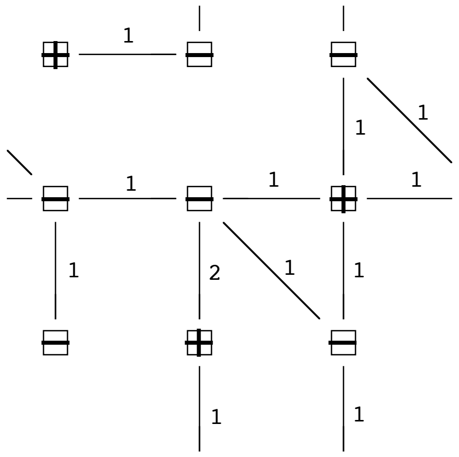
[10] D. Kandel, E. Domany, and A. Brandt, "Simulations without critical slowing down: Ising and three-state Potts models," *Physical Review B* **40**, 330 (1989); D. Kandel, E. Domany, D. Ron, A. Brandt, and E. Loh, "Simulations without critical slowing down," *Physical Review Letters* **60**, 1591 (1988).



As in the Swendsen-Wang algorithm, spins connected by frozen bonds are regarded as a cluster that flips together. The figure below shows the cluster configuration with live interactions for the above figure. (That is, the double bonds are eliminated and replaced by connected clusters that are denoted by boxes.)



This clustering procedure is similar to that in the Swendsen-Wang algorithm, but it avoids any cluster to contain more than one coarse spins. We now consider each cluster containing a squared spin site to be a coarse spin. (We will forget clusters that do not contain any squared spin site until we describe the "uncoarsening" procedure.) Interaction between any two coarse spins is the sum over all the bond strengths connecting the two clusters, and this defines a coarse energy function.



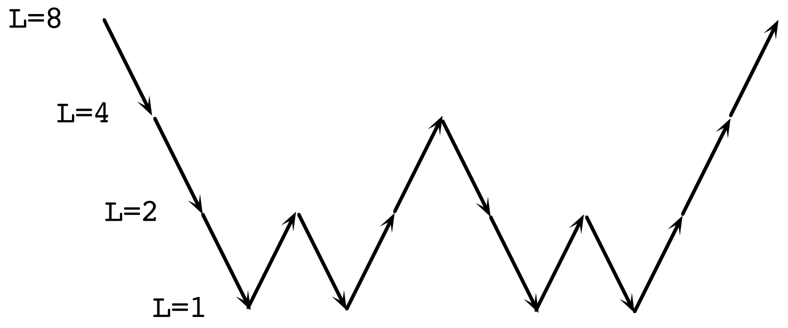
The coarse problem now has some non-nearest-neighbor bonds. However the above coarsening procedure can be applied recursively to this coarse lattice.

Uncoarsening

To "uncoarsen" the system, every decoupled block is set to some arbitrary value of spin, and all finer-lattice spins take the value of the block spin to which they were frozen. Finally the couplings that were present at this finer level are restored.

Multigrid cycling

At each coarsening level, the energy function at the level is used to perform the standard Metropolis procedures. The multigrid MC algorithm "cycles" all length scales, starting from the finest. Each level is coarsened twice before it is uncoarsened (see the figure below).



MULTICANONICAL MC[11]

The magnetic phase transition we have studied in our lattice MC simulation is called a **second-order phase transition**, meaning that the mean magnetization changes continuously as a function of temperature. Another type of phase transition, **first-order phase transition**, is even more problematic computationally. An example is the melting of a material, where the volume of the solid phase changes discontinuously to that of the liquid phase at the melting temperature. For first-order phase transitions, the correlation time scales exponentially with the system size, L,

$$\tau \sim \exp\left(F_S L^{d-1}\right),$$

for d-dimensional systems, where F_S is called the interfacial free energy. Thus *the computation for the MC simulation of a first-order phase transition scales exponentially with the system size!*

The multicanonical MC algorithm proposed by Berg and Neuhaus in 1992 reduces the correlation time to

$$\tau \sim L^{d\alpha},$$

where $\alpha \sim 1$, i.e., the algorithm reduces the computational complexity from exponential to nearly linear!

Multicanonical ensemble

The origin of the exponential correlation time is the Boltzmann weight,

$$P_B\left(E,T\right) \sim \exp\left(-E/k_B T\right),$$

which suppresses the occurrence of high-energy states. Suppose two states in the configuration space form double minima of the energy function (see the figure below). The corresponding Boltzmann

[11] B. A. Berg and T. Neuhaus, "Multicanonical ensemble: a new approach to simulate first-order phase transitions," *Physical Review Letters* **68**, 9 (1992).

weight exhibits double peaks. The energy barrier separating the two minima makes the transition from one minimum to the other so rare that we cannot observe it within our MC simulation.

For example, in a gallium arsenide crystal a defect called dislocation can move from one lattice site to a neighboring site with a barrier of a few electron volt ($E \sim 2$ eV or $E/k_BT \sim 20{,}000$ K). At room temperature (T ~ 300 K), the Boltzmann factor is $\exp(-E/k_BT) \sim 10^{-29}$. A defect in this system moves after 10^{29} attempts!

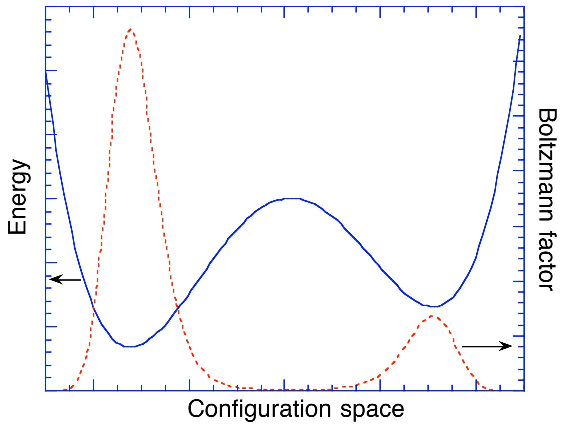


Figure. Double minimum energy well and the corresponding Boltzmann factor.

In order to explain the multicanonical ensemble, we need to introduce the density of states, $D(E)$: The number of states in the energy range, $[E, E+\Delta E]$, is given by $D(E)\Delta E$. The probability to find the state in the energy range, $[E, E+\Delta E]$, is then given by

$$\rho_B(E,T) = D(E)P_B(E,T),$$

for the Boltzmann distribution. In the multicanonical ensemble, instead, the probability density is defined to be constant:

$$\rho_M(E) = D(E)P_M(E) = \text{constant}.$$

That is, the multicanonical weight is given by

$$P_M(E) = 1/D(E).$$

With the uniform probability distribution in the multicanonical ensemble, MC simulation leads to a 1-dimensional random walk in the energy space, allowing itself to escape from any energy barrier to explore a wide range of energy space.

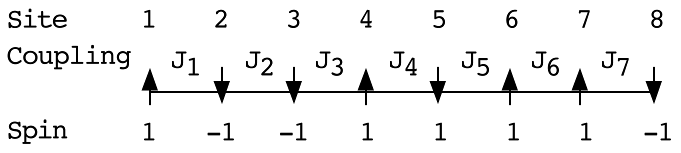
In principle, a long random sampling should give us the information on $P_M(E)$. Berg and Neuhaus have developed an efficient algorithm to estimate it. Using this weight factor, a standard Metropolis MC simulation produces a population proportional to $\rho_M(E)$ (which is roughly constant). Finally, the physical probability density at temperature T is obtained by re-weighting,

$$\rho_B(E,T) \sim \rho_M(E)P_M^{-1}(E)\exp(-E/k_BT).$$

GENETIC ALGORITHM[12]

Genetic algorithm is another type of stochastic algorithms. It also generates a Markov chain, but in a way different from Metropolis' algorithm.

To be specific, let us consider a one-dimensional Ising spin model. The system is represented by an array of spins, $S = (S_1, S_2, ..., S_{N+1})$, where each spin is either up (spin variable with 1) or down (-1).



The energy of the system is given by

$$E(S) = -\sum_{i=1}^{N} J_i S_i S_{i+1},$$

where the coupling strength, J_i, between spins varies from site to site.

Population in the solution space

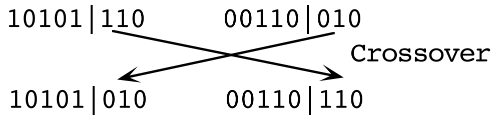
A system state can be encoded by a binary string of length N+1, by assigning 1 to an up spin and 0 to a down spin. A genetic algorithm starts by randomly generating M binary strings each of length N+1. These strings constitute an initial "population" in the solution space.

Selection

Associated with each string is an energy value, and we can rank all the strings in the population according to this value. A new "elite" population is generated by keeping low-energy strings with high probability. We can use the Boltzmann probability proportional to $\exp(-E/k_BT)$ for this purpose.

Crossover

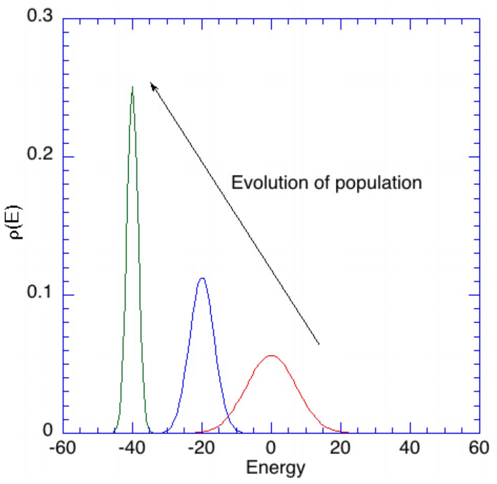
Next a crossover operation is applied to the new population: A pair of strings are chosen randomly from the population and their substrings after a randomly chosen lattice site are swapped (see the figure below).



A GA simulation repeatedly apply the selection and crossover operations to a population:

$$\rho_t(E) \xrightarrow{selection} \rho_t^S(E) \xrightarrow{crossover} \rho_t^{SC}(E) = \rho_{t+1}(E),$$

where $\rho_t(E)$ is the probability density function after t GA steps. The selection step narrows the distribution and lowers the mean energy (it is a greedy step). The crossover step adds diversity to the population so that the population in total can explorer a wider solution space efficiently. Crossover also allows global spin updates. Prügel-Bennett and Shapiro used a statistical mechanics

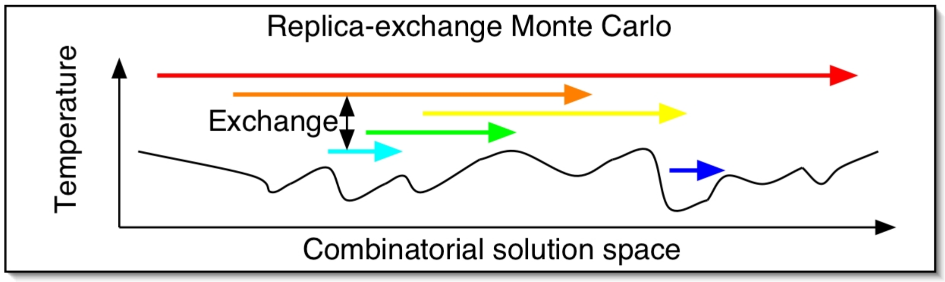


[12] A. Prügel-Bennett and J. L. Shapiro, "Analysis of genetic algorithms using statistical mechanics," *Physical Review Letters* **72**, 1305 (1994).

approach to analyze how the selection and crossover operations change the probability density function for the 1D Ising model.

REPLICA EXCHANGE MONTE CARLO[13]

Instead of the single thread of states in the Markov chain, the replica exchange Monte Carlo algorithm keeps a population of Markov chains at different temperatures. High-temperature chains explore wide volumes of the phase space, whereas low-temperature chains find local minima efficiently. By periodically swapping the temperatures of two threads, we can combine the broad/refined search strategies.

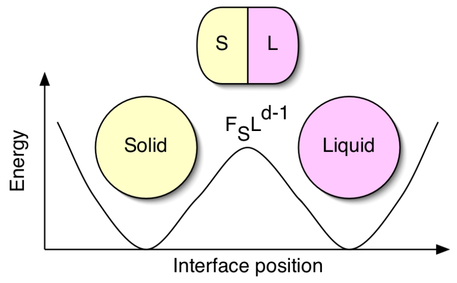


OTHER DEVELOPMENTS

Recently, low-complexity cluster MC algorithms for lattice spin models with long-range interactions[14] have attracted much attention in the light of experimentally realized vortex states in nanomagnets.[15] Also, an extension of the cluster MC algorithm (named geometric cluster algorithm) has been applied to non-lattice systems such as a particle model of complex fluids.[16]

APPENDIX: INTERFACIAL ENERGY

The first-order phase transition needs to go through an intermediate state, in which an interface separates the two phases. The interface has a higher (interfacial) energy, and is characterized as a "barrier" state. The barrier energy is obtained by multiplying the positive interfacial energy density, F_S, and the interfacial area, L^{d-1}, where L is the linear system size and d is the spatial dimension.



[13] A. Mitsutake, Y. Sugita, and Y. Okamoto, "Replica-exchange multicanonical and multicanonical replica-exchange Monte Carlo simulations of peptides. I. Formulation and benchmark test," *Journal of Chemical Physics* **118**, 6664 (2003).
[14] E. Luijten and H. W. J. Blöte, "Monte Carlo method for spin models with long-range interactions," *Int'l J. Mod. Phys.* **6**, 359 (1995).
[15] S.-B. Choe, et al., "Vortex core-driven magnetization dynamics," *Science* **304**, 420 (2004); T. Shinjo, et al., "Magnetic vortex core observation in circular dots of permalloy," *Science* **289**, 930 (2000).
[16] J. Liu and E. Luijten, "Rejection-free geometric cluster algorithm for complex fluids," *Phys. Rev. Lett.* **92**, 035504 (2004).