PY421/621 – Advanced Computing in Physics

Project #4:

-stochastic simulation techniques;

-the Metropolis Monte Carlo algorithm;

-simulation of the Ising model and Potts model;

-computational details;

-parallel implementation of the Metropolis algorithm.

Stochastic simulation techniques

In the study of thermodynamical phenomena the average over thermal fluctuations of a definite observable can be expressed as its weighted average over phase space with a weight given by the Boltzmann factor $\exp(-\frac{E}{kT})$. In this formula E is the total energy of the classical system, T is its temperature and k is Boltzmann constant. (We will not be considering quantum systems here. Their thermal averages can also be expressed in terms of suitable integrals, but one must combine the formalism of thermodynamics with the path integral formulation of quantum mechanics.)

Thus, for instance, for a gas of N non-interacting atoms, the energy is given by

$$E = \sum_{i=1,3\ k=1,N} \frac{p_{i,k}^2}{2m} \tag{1}$$

$p_{i,k}$ being the i^{th} component of the momentum of the k^{th} atom and m the common mass of all atoms.

The thermal average of the energy itself is then given by

$$< E >= Z^{-1} \int \prod_{i,k} dp_{i,k} \Big(\sum_{i=1,3\ k=1,N} \frac{p_{i,k}^2}{2m} \Big) \exp\Big(- \sum_{i=1,3\ k=1,N} \frac{p_{i,k}^2}{2mkT} \Big) \tag{2}$$

The normalizing factor Z in this equation is called the partition function and is a very important thermodynamical quantity. It is of course given by

$$Z = \int \prod_{i,k} dp_{i,k} \exp\big(- \sum_{i=1,3\ k=1,N} \frac{p_{i,k}^2}{2mkT}\big) \tag{3}$$

The integral in Eq. (2) factorizes and so can be computed very easily in terms of the two Gaussian integrals

$$\int dp \exp(-\frac{p^2}{2mkT}) = \sqrt{\pi(2mkT)} \tag{4a}$$

$$\int dp p^2 \exp(-\frac{p^2}{2mkT}) = mkT\sqrt{\pi(2mkT)} \tag{4b}$$

and thus one obtains

$$< E >= \sum_{i=1,3\ k=1,N} (\frac{mkT}{2m}) = \frac{3NkT}{2} \tag{5}$$

recovering the well known fact that every degree of freedom contributes a term $\frac{1}{2}kT$ to the energy of a gas of non-interacting atoms in thermodynamical equilibrium.

However the example given above constitutes one of the very few physically interesting cases where the thermal averages can be calculated analytically. If we imagine for instance that the atoms interact through two body potentials $V(x_{k1}, x_{k2})$ (we omit here and in the following equations explicit reference to the coordinate index i) the average over the spatial coordinates (omitted in Eqs. (2,3) because of its triviality) becomes

$$< O(x) >= Z_x^{-1} \int \prod_{k=1,N} d^3x_k\, O(x) \exp\big[- \frac{1}{kT} \sum_{k1=1,N\ k2=1,k1-1} V(x_{k1}, x_{k2})\big] \tag{6}$$

with

$$Z_x = \int \prod_{k=1,N} d^3x_k \exp\big[- \frac{1}{kT} \sum_{k1=1,N\ k2=1,k1-1} V(x_{k1}, x_{k2})\big] \tag{7}$$

Again, apart for very special cases (e.g., the case where the potentials are harmonic oscillator potentials so that the integrals become Gaussian integrals) the integrals in Eqs. (6,7) cannot be evaluated analytically and one must resort to numerical techniques. However, the methods that can be used to calculate integrals of low dimensionality cannot be applied either. In such algorithms one typically divides every axis of integration into some number of subintervals and approximates the integral wit a sum over the values the integrand takes at definite sampling points within the subintervals. Now it is patent that but for

the smallest number of atoms such methods cannot work: even with 20 atoms only and with 4 sampling points for each of the $x_{i,k}$ coordinates (most likely too small a number) we would need to evaluate the integrand at $4^{60} = 2^{120} \approx 10^{36}$ points, which is out of the question! What we will see in these lectures is that the integrals *can* in fact *be approximated numerically*, but that, rather than use a regular array of sampling points, one must use a stochastic distribution of sampling points, with a cleverly chosen distribution. The algorithms that use stochastic sampling to approximate integrals, or also sums, over many variables are generally called Monte Carlo algorithms.

The Ising model

We will illustrate the way Monte Carlo algorithms work by calculating the thermal averages for a system which is even simpler than the gas of interacting atoms of Eqs. (6,7), a system where the variables are discrete rather than continuous as the coordinates of the atoms $x_{i,k}$. The system we will consider is known as the Ising model (later we will consider also a generalization known as Potts model) and is defined as follows. One assumes that there is some geometric lattice of sites, so that the notion of nearest neighbor of a given site has a precise meaning. Most often these sites will form a square lattice (in two dimensions) or a cubical lattice (in 3D). Let us label for the moment a generic site with an index i. The dynamical variables are "spins" s_i defined over the sites which can take one of two alternative values. Of course any labeling of such values would do. We will label them by 0 and 1 so that for each site i either $s_i = 0$ or $s_i = 1$. (However, frequently the possible values of the spins are denoted by 1 and -1, and in a computer program the use of logical variables s(i)=.TRUE. or .FALSE. would also make sense.)

An "observable" O of the system is any function of its spins: $O = O(s_i)$. In order to define the thermodynamics of the model we must specify its energy. This is in some sense the most important observable. The energy E is defined as follows. E is the sum of terms associated with the links, or bonds, between all nearest neighbor sites in the lattice. Let us denote a pair of nearest neighbor sites by i, j. Then the energy $E_{i,j}$ associated with the bond is 0 if $s_i = s_j$ and is 1 otherwise.

$$E_{i,j} = 1 - \delta(s_i, s_j) \tag{8}$$

where the Kronecker symbol $\delta(x, y)$, defined for discrete variables x, y, equals 1 if $x = y$ and 0 otherwise.

For the total energy we have then

$$E = \sum_{<i,j>} E_{i,j} = 1 - \delta(s_i, s_j) \tag{9}$$

where by the symbol $< i, j >$ we denote a pair of nearest neighbor sites, i.e. the sum is extended to all pairs of neighboring sites (each pair of nearest neighbors being counted only once).

Starting from this expression for the energy we can now define the thermodynamical averages. They are given by

$$< O >= Z^{-1} \sum_{s_i} O(s_i) \exp\big(- \frac{E}{kT} \big) = Z^{-1} \sum_{s_i} O(s_i) \exp\big(- \sum_{<i,j>} \frac{1 - \delta(s_i, s_j)}{kT} \big) \qquad (10)$$

with

$$Z = \sum_{s_i} \exp\big(- \frac{E}{kT} \big) = \sum_{s_i} \exp\big(- \sum_{<i,j>} \frac{1 - \delta(s_i, s_j)}{kT} \big) \qquad (11)$$

The sum in these equations is extended to all possible values of the spins.

Our goal is to calculate thermodynamical averages given by the equations above, i.e. to obtain an accurate numerical approximation to the sums appearing in Eqs. (10) and (11).

Typical observables in whose average we will be interested in include the energy of the individual bonds, in which case

$$O = E_{<i,j>} = 1 - \delta(s_i, s_j) \qquad (12)$$

and the "magnetization" of the spins themselves, which we obtain taking

$$O = m_i = 1 - 2s_i \qquad (13)$$

(We define the magnetization in this fashion so that it takes the two more symmetric values 1 and -1 in correspondence to the two possible values, 0 and 1, of the spins.)

Before we move on to the actual computational techniques, let us briefly consider qualitatively what we can expect for the thermodynamical averages of E and m above.

If we take a very low temperature, the Boltzmann factor

$$\exp\big(- \frac{E_{total}}{kT} \big) = \exp\big(- \sum_{<i,j>} \frac{1 - \delta(s_i, s_j)}{kT} \big)$$

will become extremely small as soon as the the bonds are excited, i.e. as soon as the spins at the end vertices of the bonds take opposite values, because the denominator kT in the argument of the exponential makes the argument itself very negative as soon as there is an appreciable number of 1s in the sum.

Thus, only configurations where almost all of the spins are aligned will effectively contribute to the averages. This implies that the average energy of the bonds will be very close to 0. For the magnetization the argument is slightly more subtle. Our intuition would tell us that, since the spin will be mostly aligned, the average magnetization should be very close

to 1 or -1. However, in principle, there is no differentiation between these two values in the sums of Eqs. (10,11). The symmetry of the sums under a reversal of all of the spins ($s_i = 0 \leftrightarrow s_i = 1$) implies that the average value of the magnetization must be always zero. The way to recover a result in agreement with our intuition is to introduce a small symmetry breaking term in the energy through an additional "magnetic" field h coupled to the spins, i.e., to replace

$$E \to E - h \sum_i (1 - 2s_i) \tag{14}$$

Then, for very small temperature and, say, positive h, the extra term in the exponent of the Boltzmann factor will overwhelmingly favor the configurations where the spins are aligned with positive magnetization (i.e. $s_i = 0$) and the system will exhibit a strong average magnetization. If we let $h \to 0$ with a finite system, of course continuity demands that $< m > \to 0$. However, if we take the *limit of an infinite volume* (i.e. of an infinite number of spins) *first* and then the limit $h \to 0$, the average non-vanishing magnetization will persist even for $h = 0$. In this way one can recover the fact that the system exhibits spontaneous magnetization at low temperature.

If we consider a very large value of kT instead, the exponent in the Boltzmann factor will be small for all spin configurations, and therefore all values of the spins will be likely to occur with almost identical weight. This means that the averaging over configurations leading to the thermodynamical averages will become averages over almost random configurations of spins (extreme thermodynamical fluctuations). The energy of the individual bonds will then have an average very close to $\frac{1}{2}$ and the average magnetization will be 0.

The crucial question is what happens in between the two extreme regimes. Will there be any of those singularities in the behavior of the observables or of their derivatives which characterize the presence of a phase transition? These are the questions that can be solved by a numerical calculation of the averages of Eq. (10).

A straightforward computer calculation of the sums leading to the averages involves, however, way too many terms to be feasible for any systems but those of the smallest size (with N spins the sums contain 2^N terms, and even with N as small as 40 we enter into the multiTeraflop domain). There are more sophisticated techniques by which one can calculate the sums exactly using some form of recursion over the lattice size and which can be carried out for larger numbers of spins. But even these methods fail to reach the sizes that are required to answer to a variety of interesting dynamical questions. Moreover, they become harder or impossible to implement for a larger number of degrees of freedom per variable (in particular, with continuous variables, such as the coordinates of an atom).

A very large number of interesting thermodynamical averages can however be calculated, in an approximate but frequently very precise manner, by stochastic sampling techniques. The idea is to go back, in a sense, to the true dynamics that underlies the thermodynamical averages. In reality these are not averages over the "phase space" of all possible configurations, but rather averages of the configurations that the system goes through, in very rapid succession, over its time evolution. It is a remarkable theoretical achievement

that such averages can be reexpressed as averages over phase space.

Thus, for a numerical calculation, we will try to emulate the actual dynamics in the following way:

- the whole configuration C (e.g. the collection of the values of the N spins, for the Ising model) will be stored in the memory of the computer;

-by a suitable stochastic algorithm the configuration C will be replaced by a new configuration C'

$$C \to C' \tag{15}$$

giving origin to a stochastic sequence (or Markov chain):

$$C_0, C_1, \ldots C_n, C_{n+1}, \ldots \tag{16}$$

-the algorithm must be such that, when the sequence of configurations C_n reaches statistical equilibrium, the configurations are distributed according to the Boltzmann distribution, i.e., if $P(C)$ denotes the probability of encountering a configuration C in the sequence, then in statistical equilibrium

$$P(C) = Z^{-1} \exp\big(-\frac{E(C)}{kT}\big) \tag{17}$$

$E(C)$ being, obviously, the energy of the system in its configuration C.

It will be convenient to use the notation β for the factor $\frac{1}{kT}$ that always appears in the Boltzmann distribution. With this notation

$$P(C) = Z^{-1} \exp[-\beta E(C)] \tag{18}$$

- finally the averages over the observables will be approximated by averages over the values $O(C)$ that the observable takes oner the configurations encountered in the stochastic sequence (16) after the sequence has reached statistical equilibrium:

$$< O > \approx \frac{1}{N} \sum_{n=N_0+1}^{N_0+N} O(C_n) \tag{19}$$

The Metropolis Monte Carlo Algorithm

The crucial question becomes now, of course, how such an algorithm should be designed. There are several stochastic algorithms (as well as deterministic algorithms, but we will consider these when we will talk about molecular dynamics simulation) that accomplish

our goal. We will describe one of the conceptually simplest, yet very powerful and very widely used algorithm that is known as the algorithm of Metropolis et al. (first published in N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, J. Chem. Phys. 21, p. 1087, 1953) or the Metropolis Monte Carlo Algorithm.

The basic iterative step of the Metropolis algorithm consists in the following:

- by some preliminary transition probability $p_0(C \to C')$ that must satisfy the reversibility condition

$$p_0(C \to C') = p_0(C' \to C) \tag{20}$$

one selects a candidate new configuration C';

- one calculates the variation of the total energy of the system induced by the proposed change:

$$\triangle E = E(C') - E(C) \tag{21}$$

- if $\triangle E < 0$, so that the energy of C' is lower that the energy of C the proposed change is accepted and C' replaces in memory the current value C. Otherwise one extracts a random number r with uniform probability distribution in the interval $0 \leftrightarrow 1$ and the change is accepted if

$$r < \exp(-\beta \triangle E) \tag{22}$$

The algorithm we just described satisfies a very important condition known as the property of detailed balance. It is expressed by the equation

$$\frac{p(C \to C')}{p(C' \to C)} = \frac{P(C')}{P(C)} = \frac{\exp[-\beta E(C')]}{\exp[-\beta E(C)]} \tag{23}$$

where $p(C \to C')$ denotes the overall probability of a transition from configuration C to configuration C' in the stochastic sequence (note, p is not the same as the preliminary transition probability p_0).

It is easy to prove that the transition probability p for the algorithm described above satisfies to Eq. (23). Let us assume, for definiteness, that C' has a higher energy than C. Then the overall probability of a transition from C to C' is given by the probability $p_0(C \to C')$ of selecting C' as a candidate new configuration to begin with, multiplied by the probability $\exp(-\beta \triangle E)$ of accepting the actual change. On the other hand, the probability of the reverse transition will just be given by $p_0(C' \to C)$ since, once C is selected, the change will be always accepted, the energy of C being lower than that of C'. Thus

$$\frac{p(C \to C')}{p(C' \to C)} = \frac{p_0(C \to C') \exp(-\beta \triangle E)}{p_0(C' \to C)} =$$

(on account of the reversibility condition, Eq. (20))

$$= \exp(-\beta \triangle E) = \exp\{-\beta[E(C') - E(C)]\} = \frac{\exp[-\beta E(C')]}{\exp[-\beta E(C)]} \tag{24}$$

We will now prove that when a stochastic algorithm satisfies the detailed balance condition of Eq. (23), then the target probability distribution

$$P(C) = Z^{-1} \exp[-\beta E(C)] \tag{25}$$

is an eigenstate of the stochastic process. This means that if at some point the configurations are distributed according to that probability, they will remain so distributed at all subsequent steps in the stochastic chain.

The proof is again very simple and goes as follows. Let us assume that at some point the probability of having a definite configuration C is indeed given by $P(C)$ and let us then calculate the probability $P'(C')$ of encountering a configuration C' at the next step. This will be given by

$$P'(C') = \sum_C P(C) p(C \to C') \tag{26}$$

On account of the detailed balance condition (Eq. (23)) the r.h.s. of the above equation is equal to $\sum_C P(C') p(C' \to C)$. Thus

$$P'(C') = \sum_C P(C') p(C' \to C) = P(C') \sum_C p(C' \to C) = P(C') \tag{27}$$

In the last step we have used the completeness condition $\sum p(C' \to C) = 1$, obeyed by stochastic transition probabilities, which simply states that the sum of the probabilities to end up into some final state starting from a given initial state must add up to one.

The property of $P(C)$ of being an eigenvector of the transition probability matrix does not guarantee per se that the stochastic process will converge to this distribution. However one can also prove that, if one defines a metric in the space of probability distributions by defining a distance:

$$\mathrm{dist}(P, P') = \sum_C |P(C) - P'(C)| \tag{28}$$

then the distance between the current probability distribution and the target probability distribution can never increase in the course of the stochastic iterations.

This is as well as one can do on totally general grounds. More detailed information on the actual convergence of the process to $P(C)$ and on the rate of convergence must follow a more precise specification of the algorithm (and is in any case not easy to obtain on purely analytical grounds). However, in practice, the algorithm we have described above works very well in a large variety of applications.

Finally, let us conclude with an important observation. The detailed balance condition was used as a key ingredient in the proof that the target distribution is an eigenstate of the stochastic process. This is a sufficient, but not necessary condition. There are many algorithms for the approximate calculation of statistical averages of different nature than the Metropolis Monte Carlo algorithm we have described above. Some are stochastic and do obey the detailed balance condition, but there are other algorithms that do not obey it (and still produce, of course, the required distribution), and there are deterministic algorithms as well.

The Metropolis Monte Carlo algorithm: computational details
The Potts model

We will discuss the details of the actual computational implementation of the Metropolis algorithm. It will be convenient to do so in the context of a simple, but very important generalization of the Ising model, given by the q-state Potts model.

The Potts model is quite similar to the Ising model. The spin variables, however, can take q different values rather than two. We will take these values to be one of the integers between 0 and $q-1$ included. Everything else remains unchanged. In particular, the energy associated with a bond is still given by

$$E_{i,j} = 1 - \delta(s_i, s_j) \qquad 0 \leq s_1,\, s_j \leq q-1 \tag{29}$$

and the thermodynamical averages are given by

$$< O >= Z^{-1} \sum_{s_i} O(s_i) \exp(-\beta E) = Z^{-1} \sum_{s_i} O(s_i) \exp\big(-\beta \sum_{<i,j>} 1 - \delta(s_i, s_j)\big) \tag{30}$$

with $\beta = \frac{1}{kT}$ and

$$Z = \sum_{s_i} \exp(-\beta E) = \sum_{s_i} \exp\big(-\beta \sum_{<i,j>} 1 - \delta(s_i, s_j)\big) \tag{31}$$

The Potts model has properties similar to the Ising model (to which it reduces for $q = 2$). In systems of dimensionality larger than one it exhibits a phase transition at a definite, critical

value $\beta = \beta_{cr}$. For a two dimensional square lattice analytical arguments based on a change of variables in the sums giving the thermodynamical averages (duality transformations) can be used to prove that $\beta_{cr} = \log(1 + \sqrt(q))$. As q increases the phase transition strengthens and, in particular, in two dimension it changes from a second order phase transition for $q = 2, 3, 4$ to a first order transition for $q \geq 5$. Moreover the discontinuity in internal energy increases for increasing q (≥ 5).

In order to make the Metropolis algorithm well defined, we must specify the procedure by which the new, candidate configuration C' is selected (cfr. Eq. (20)). While more elaborated schemes are possible, in a typical implementation of the algorithm C' would differ from the current configuration C by the value of a single spin. Thus, the algorithm works as follows:

1) all of the spins s_i are stored in the memory of the computer. Their collection constitutes the current configuration C. Note that i is a rather generic index. In a typical application, e.g. for a two dimensional model, the spins could be indexed by two integer values coordinates $x,\ y$.

2) A lattice site is chosen (see later for the selection criterion). We denote its index by j.

3) A new candidate value $s'_j \neq s_j$ is chosen for the spin at the selected site. In the Potts model, because of the symmetry of the energy with respects to permutations of the values of the spins, s'_j will be chosen at random, with uniform probability distribution, among all of the values $0 \leq s'_j \leq q-1$ different from s_j. In practice, the selection can be implemented by extracting a real pseudorandom number $0 < r < 1$ with uniform distribution and by defining $s'_j = (s_j + (q-1)r + 1)$ mod q. (With $q = 2$, i.e. in the Ising model, it is of course not necessary to extract a pseudorandom number to select s'_j, since this variable can only take the value $1 - s_j$.) Notice that in the implementation of the algorithm for other models, still defined in terms of discrete variables s_i but with a different rule for the energy, it might be more convenient to use a different procedure to select s'_j. For instance s'_j could be chosen, with equal probability, to be equal to either $s_j + 1$ or $s_j - 1$ (always mod q). The only crucial criterion is that the probability of selecting s'_j starting from a given s_j must be equal to the probability of selecting s_j starting from s'_j.

4) This defines the new candidate configuration C', i.e. C' is the configuration where all of the spin variables s_i take the same values as in C apart from the spin at site j which takes value s'_j. At this point one evaluates the change in energy $\triangle E = E(C') - E(C)$. Note the very important fact that, because of the locality of the energy (i.e. the energy is the sum of many terms in which only nearest neighbor spins are "coupled"), one does not need to reevaluate the whole energy to calculate $\triangle E$, but only the contributions to the old energy $E(C)$ and to the new energy $E(C')$ coming from the terms which involve the spin at site j. In our case there are 4 or 6 such terms in 2 dimensions and 3 dimensions, respectively.

5) Finally one extracts another real pseudorandom number $0 < r < 1$ with uniform distribution and accepts the change if $r \leq \exp(-\beta \triangle E)$. That is, if the conditional is satisfied, s'_j is stored back in memory, replacing the current value s_j and C' becomes the new configuration C_{n+1} in the stochastic sequence. Otherwise, the configuration is left

unchanged and $C_{n+1} = C$.

We return now to the choice of lattice site i. This can be chosen

(i) at random among all of the sites in the lattice, or

(ii) by scanning the lattice sites one at a time according to some predefined ordering.

Both methods are valid, i.e. produce a sequence of configurations C_n that has the probability distribution $P(C) = \exp(-\beta E)/Z$ as eigenstate and which will in general converge to this limiting distribution. We must remark, however, that the proof of the properties of the algorithm given above strictly applies only to method (i). The reason is that, if one selects the sites according to a predefined order, one is not dealing with a single preselection probability $p_0(C \to C')$, but with a set of preselection probabilities $p_0^j(C \to C')$, one for each of the visited sites. The arguments used to prove that the Boltzmann distribution $P(C) = \exp(-\beta E)/Z$ can be nevertheless generalized also to this case.

In most applications one visits the lattice sites according to a definite order. In particular, this is necessary for a simple implementation of the algorithm on a parallel machine, and we will restrict our attention to this case. In order to establish some terminology, we will say that the steps (1-5) defined above constitute the upgrade of a single spin (even if the value of the spin is left unchanged), or, more simply, one upgrading step, or, even more simply, one step in the algorithm. When all of the spins of the lattice have been upgraded once, we will say that one has upgraded the whole configuration (even is, properly speaking, the configuration is upgraded at every step) and that one has performed one iteration of the Metropolis procedure.

Parallel implementation of the Metropolis algorithm

In the Metropolis algorithm the variation of the energy produced by the proposed change of a definite spin must be calculated while keeping all of the other spins fixed. This implies, in particular, that one cannot preselect several spins simultaneously and perform the accept-reject steps in parallel *if the chosen set of spins contains nearest neighbors*. Indeed, let us imagine that we want to upgrade simultaneously the spins $\{s_{i_1}, s_{i_2}, \ldots s_{in}\}$. Let the corresponding new candidate values be $\{s'_{i_1}, s'_{i_2}, \ldots s'_{in}\}$. The Metropolis algorithms would then require that we calculate the overall change of energy induced by the variation of all the spins in the set, and accept or reject the change of all of the spins as a whole. Accepting or rejecting the upgrades of the spins in the set individually would not implement the rules of the algorithm and and would violate the detailed balance condition. On the other hand, the simultaneous variation of a large number of spins would typically produce a large increase in the energy of the system and thus lead almost invariably to a rejection of the proposed upgrade. The algorithm, while still in principle correct, would become extremely inefficient.

From all of this it might seem that the Metropolis Monte Carlo algorithm is intrinsically serial, unable to take advantage of the speed up generated by parallel processing. However, the keywords in the above discussion are "if the chosen set of spins contains nearest neighbors". On the contrary, let us imagine that the set $\{s_{i_1}, s_{i_2}, \ldots s_{in}\}$ does not contain

nearest neighbor spins. Then, even if we proceed through the upgrades of the spins serially, as demanded by the algorithm, the probabilities for the upgrade of the individual spins will nevertheless be independent of each other. Specifically, once s_{i1} has been upgraded, the steps for the upgrade of s_{i2} make no reference to the outcome of the upgrade of s_{i1}, since s_{i1} is not one of the nearest neighbors of s_{i2}. Similarly, the upgrade of s_{i3} makes no reference to the former upgrades of s_{i1} and s_{i2} since neither of these spins is one of the nearest neighbors of s_{i3}, etc. Thus, although conceptually we may think that the spins are upgraded serially, in practice the steps leading to the upgrade of $\{s_{i1}, s_{i2}, \ldots s_{in}\}$ can all be taken in parallel. The outcome of these discussion is that the Metropolis Monte Carlo algorithm can be implemented in parallel over subsets of all the spins which have the property of being "statistically independent", i.e. not coupled in the expression for the energy, so that the Boltzmann factor $\exp(-\beta E)$ can be decomposed into the product of factors each containing only one of the spins in the set. With the Ising and Potts models we have been considering, this means that the sets must not contain any pair of nearest neighbor spins.

We will describe now two ways of decomposing the spins in the square lattice into sets of statistically independent spins which can be upgraded in parallel. One subdivision, the so called "checkerboard subdivision", is particularly suitable for a data parallel implementation of the algorithm. The other subdivision lends itself to an efficient implementation of the "message passing" programming technique.

The checkerboard subdivision is represented in the figure labelled (A) below. We illustrate there a lattice of size $(N_x = 6) \times (N_y = 4)$. We assume that N_x and N_y are even. The symbols $\otimes$ represent the lattice sites, the numbers by the lattice sites indicate their progressive numbering, with and index i ranging from 0 to $N_x N_y - 1$ and the letters e and o, for even and odd, label the lattice sites which can be upgraded in parallel.

$$\begin{array}{cccccc} \otimes 21, o & \otimes 9, e & \otimes 22, o & \otimes 10, e & \otimes 23, o & \otimes 11, e \\ \otimes 6, e & \otimes 18, o & \otimes 7, e & \otimes 19, o & \otimes 8, e & \otimes 20, o \\ \otimes 15, o & \otimes 3, e & \otimes 16, o & \otimes 4, e & \otimes 17, o & \otimes 5, e \\ \otimes 0, e & \otimes 12, o & \otimes 1, e & \otimes 13, o & \otimes 2, e & \otimes 14, o \end{array} \qquad \text{(A)}$$

The explicit mapping between the index i and the lattice coordinates x, y (with $0 \leq x \leq N_x - 1$, $0 \leq y \leq N_y - 1$) is better expressed in terms of a pair of subindices $j = 0 \ldots (N_x N_y/2) - 1$ and p the parity index $= 0, 1$:

$$i = j + p N_x N_y / 2 \qquad (32a)$$

$$j = \mathrm{int}(x/2) + y(N_x/2); \quad p = (x + y) \bmod 2 \qquad (32b)$$

We see from these equations and from Fig. (A) that all the sites with even parity, i.e. with coordinates whose sum is even, are numbered first, then the sites with odd parity follow.

Since even and odd sites are statistically independent, upgrading all the even sites simultaneously first, and then all the odd sites simultaneously, has the same effect as going serially through the lattice according to the index i. Thus, on a multiprocessor, it is possible to upgrade the even and the odd sites in parallel, farming out the job of upgrading the spins of the two sublattices to different processors.

For message passing programming it is useful to subdivide the lattice into blocks of contiguous sites. For example, let us imagine that the size in the x-direction is a multiple of some integer $M \geq 2$: $N_x = MK$, where K is of course also an integer. We can then subdivide the lattice into K blocks of M columns each. We number the sites then by proceeding sequentially through homologous sites in the various blocks. The formulae relating the numbering index i to the coordinates are as follows:

$$i = Kj + k \tag{33a}$$

$$j = y + N_y(x \bmod M); \quad k = \mathrm{int}(x/M) \tag{33b}$$

where $k = 0 \ldots K-1$ labels the blocks. This is illustrated in Fig. (B), where $M = 2$, $K = 3$. The spins at the sites which have the same position in the blocks, namely the sites with identical j (the sites with the same letter in Fig. (B)), can be upgraded in parallel (provided that the number of columns in the blocks is at least 2). Thus, with a multiprocessor, the upgrades of these spins can be assigned to different processors.

$$\begin{array}{cccccc} \otimes 9, d & \otimes 21, h & \otimes 10, d & \otimes 22, h & \otimes 11, d & \otimes 23, h \\ \otimes 6, c & \otimes 18, g & \otimes 7, c & \otimes 19, g & \otimes 8, c & \otimes 20, g \\ \otimes 3, b & \otimes 15, f & \otimes 4, b & \otimes 16, f & \otimes 5, b & \otimes 17, f \\ \otimes 0, a & \otimes 12, e & \otimes 1, a & \otimes 13, e & \otimes 2, a & \otimes 14, e \end{array} \tag{B}$$

Which lattice subdivision should be used in a parallel implementation of the Potts model simulation depends on the characteristics of the parallel environment. If the memory is shared among the processors and the time for memory access is not the dominant consideration, the checkerboard subdivision may be the most convenient, especially because all the spins at the even, respectively odd sites of the lattice can be upgraded in parallel without concerns about synchronization of the various parallel processes. On the other hand, if the memory is distributed among the different processors and there is no mechanism implementing a shared memory, as would be the case, for example, if the parallel environment consisted of networked workstations, then one would be forced to use the message passing model. In this case the spins associated with the various blocks would reside in the memory of the different processors, which would perform the task of upgrading the spins in the respective blocks. Of course, this requires that information about the spins at the boundary of the blocks be passed to the processors that upgrade the spins in the neighboring blocks. For example, in the situation illustrated in Fig. (B), after the processors have upgraded

the first column in the respective blocks (i.e. the spins at sites a, b, c, d) then the values of the upgraded spins must be passed to the processors at the left (with periodic boundary conditions we think of the blocks as arranged in a circular fashion, so that each block has one neighboring block at its left and one at its right), before proceeding to the upgrade of the second column. Also, this requires a synchronization barrier: all processors must complete the upgrade of the first column before upgrading the last column in the block.

Codes that simulate of the Potts model implementing the two models of parallel programming discussed above are included in the software distributed for this course.