Prof. Dr. A.K. Hartmann
WiSe 2023/24, 5.04.4521
Blatt 9, 29.02.2024

Exercises Computational Physics

9 Parallel Tempering

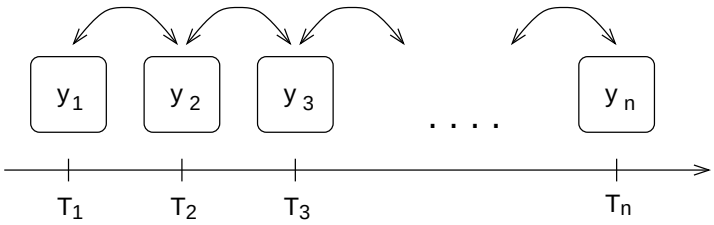
1. Preparation: Download the following files vom `StudIP`:

   `percol.c`, `percol.h`, `stacks.c`, `stacks.h`, `diluted.c`, `diluted.h`.

   (`diluted.c` is available unless you have written some functions of it yourself during the previous exercises; in this case you use your own function.)

2. *Parallel Tempering* method

   A system with configurations $\underline{y}$ and energy function $H(\underline{y})$ is given. To accelerate the convergence of Monte Carlo simulations, one simulates n_T independent configurations $\underline{y}_0, \ldots, \underline{y}_{n_T-1}$ "in parallel" at n_T different temperatures $T_0 < T_1 < \ldots < T_{n_T-1}$.



   In addition to the standard moves, in regular simulation-time intervals *swap* attempts are performed. Versuche In each swap attempt a temperatur index $k \in \{0, 1 \ldots, n_T - 2\}$ is randomly selected. Let $\underline{y}, \underline{z}$ the configurations at temperatures T_k, T_{k+1}. The two configurations are now exchanged with the Metropolis probability

$$W_{k,k+1}([\underline{y}, \underline{z}] \rightarrow [\underline{z}, \underline{y}]) = \min\left(1, e^{\Delta_{k,k+1}(\underline{y},\underline{z})}\right) , \tag{1}$$

   where

$$\Delta_{k,k+1}(\underline{y}, \underline{z}) \equiv (1/T_k - 1/T_{k+1})\Big(H(\underline{y}) - H(\underline{z})\Big) \tag{2}$$

   is a mixed temperature and energy difference. In case of an accepted swap configuration $\underline{y}$ will move to the higher temperature and $\underline{z}$ to the lower one.

   (Note: the above choice of exchange probability $W_{k,k+1}$ guarantees detailled balance for the combined system held at all temperatures.)

Basic spirit of the algorithm: at low temperatures the system "freezes", while at high temperatures the configurations evolve quickly, because energy barriers are easily overcome. With Parallel Temperaing the systems "walks" through the temperature space, automatically controlled. In this way permanent freezing of a configuration is prohibited and also at lower temperatures the system equilibrates faster.

3. Implementation

   Write a function `diluted_PT()` (e.g. in a new file `diluted_PT.c`) which takes `num_T`= n_T configurations and `num_T` temperature values as parameters. The functions performs for each configuration some sweeps (here, e.g., 1, maybe other values like `num_T` work better, never mind) of the local Monte Carlo (MC), by calling `diluted_mc_T()`. Afterwards `num_T`-1 times two neighbouring configurations at T_k, T_{k+1} are tried to be exchanged (each time $k \in \{0, 1, 2, \ldots,$`num_T`$-2\}$ is randomly selected). A full such cycle is considered as one step of the MC time.

   The Functions receives also arrays `exch[]` and `tries[]`, by which the number of accepted and attempted exchanges are counted, each time at the k-th entry. This is used to calculate the acceptance rates `(double) exch[k]/tries[k]`.

   The head of the functions is:

```
/************************* diluted_PT() ***********************/
/** Does parallel tempering for diluted ferromagnet           **/
/** at num_T temperatures T_0<T_1<...T_{num_T-1}.             **/
/** NOTE: This implmentation is for maximum simplicity,       **/
/** hence, it will perform poorly                             **/
/** PARAMETERS: (*)= return-paramter                          **/
/**          num_T: number of temperatures                    **/
/**   (*) spin_cfg: num_T spin-configurations                 **/
/**          num_n: number of neighbours in lattice           **/
/**              N: number of spins                           **/
/**           next: gives neighbours next[0..N][0..2*num_n+1] **/
/**              e: sites arre ocupied (e[i]=1),empty (=0  )  **/
/**              T: num_T temperatures in units of J          **/
/**      (*) exch: count accepted swaps T_k <-> T_{k+1}       **/
/**     (*) tries: count tried swaps T_k <-> T_{k+1}          **/
/** RETURNS:                                                  **/
/**     nothing                                               **/
/*************************************************************/
void diluted_PT(int num_T, short int **spin_cfg, int num_n, int N,
                int *next,  short int *e, double *T,
                int *exch, int *tries)
```

   Hint 1: Inside `diluted_PT()`you also have to call `diluted_energy()`.

   Hint 2: Exchange two configurations, by exchanging the corresponding two pointers stored in `spin_cfg`, do NOT exchange all spin values, which would be much slower.

(6 P)

4. Choice of temperatures

   Download the new (complete !) main program in `diluted_sim_PT.c` from `StudIP`. Read the code such that you understand what it does. Compile with:

   ```
   cc -o pt diluted_sim_PT.c diluted.c diluted_PT.c percol.c stacks.c -lm
   ```

   Perform simulations with not too high sweep number (e.g. 1000), for two-dimensional (not diluted) ferro magnets ($L = 20$) in temperature interval $[T_1, T_{n_T}] = [1.5, 2.5]$. Choose (with the program option -T) the number n_T of temperatures and their actual values by iterative trial and error such that for each pair of temperatures T_k, T_{k+1} the exchange $[\underline{y}, \underline{z}] \to [\underline{z}, \underline{y}]$ happens with about 50 percent probability (the range 30%-70% is fine).

   (2 P)

5. Question (by pondering or trying): Which adjustements to the temperatures you have to do if you increase the size L considerably? (1 P)

6. Comparison

   Plott for the $L = 20$ ferromagneten $m(t)$ curves, e.g.. with `gnuplot` at three different temperatures ($T = 1.5, 2.2, 2.5$). perform simulationen with *random* initial orientations of the spins.

   Hint: with `plot "file" using 1:3` you plot $m(t)$ in `gnuplot` for the first temperature, with `plot "file" using 1:6` for the second, etc. (1 P)