Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 153 (2003) 397–406
Computer Physics Communications
www.elsevier.com/locate/cpc

Multicanonical simulations step by step

Bernd A. Berg

Department of Physics, Florida State University, Tallahassee, FL 32306, USA

Received 20 August 2002; received in revised form 25 March 2003

Abstract

The purpose of this article is to provide a starter kit for multicanonical simulations in statistical physics. Fortran code for the q-state Potts model in $d = 2, 3, \ldots$ dimensions can be downloaded from the Web and this paper describes simulation results, which are in all details reproducible by running prepared programs. To allow for comparison with exact results, the internal energy, the specific heat, the free energy and the entropy are calculated for the $d = 2$ Ising ($q = 2$) and the $q = 10$ Potts model. Analysis programs, relying on an all-log jackknife technique, which is suitable for handling sums of very large numbers, are introduced to calculate our final estimators.

PACS: 05.10.Ln; 05.50.+q

Keywords: Multicanonical algorithm; Fortran code; All-log jackknife technique; Internal energy; Free energy; Entropy

1. Introduction

The conventional Metropolis [1] method simulates the Gibbs canonical ensemble at a fixed temperature T and allows for easy calculations of the (internal) energy and functions thereof. However, some of the most important quantities of statistical physics, free energy and entropy, can only be obtained by tedious integrations. One way to overcome this problem is by multicanonical (MUCA) simulations, which calculate canonical expectation values over a temperature range in a single simulation by using the weight factor [2]

$$w_{1/n} = \frac{1}{n(E)} = \mathrm{e}^{-S(E)} = \mathrm{e}^{-b(E)E+a(E)}, \qquad (1)$$

where $n(E)$ is the number of states with energy E and $S(E)$ the microcanonical entropy. In an extension of the microcanonical terminology one may call $b(E)$ microcanonical inverse temperature ($b(E) = 1/T(E)$ in natural units with Boltzmann constant $k_B = 1$) and $a(E)$ microcanonical, dimensionless free energy, see Appendix A.

MUCA simulations became popular with the interface tension calculation [2] of the $2d$ 10-state Potts model, when the method emerged as the winner of largely disagreeing estimates, which after their publication became resolved by exact values [3]. Similar simulation concepts can actually be traced back to the work by Torrie and Valleau [4] in the 1970s. In recent years the MUCA method has found many applications, besides for first order phase transitions [5] mainly for complex systems including spin glasses and

E-mail address: berg@hep.fsu.edu (B.A. Berg).

0010-4655/03/$ – see front matter
doi:10.1016/S0010-4655(03)00245-5

peptides, see [6] for a brief review and a summary of related methods.

The scope of this article is limited to the Ising model and its generalization in form of q-state Potts models, for a review see [7]. Fortran routines which work in arbitrary integer dimensions $d = 2, 3, \ldots$ are provided, but we confine our demonstrations to $d = 2$, to allow for comparison with rigorous analytical calculations. The Ising model simulation is seen to match the exact finite lattice results of Ferdinand and Fisher [8], while for the $q = 10$ Potts model one finds agreement with the rigorously known transition temperature and latent heat of Baxter [9]. Details of the model are summarized in the first part of Section 2. In the second part of this section the downloading of the Fortran code and its use are explained.

In Section 3 MUCA simulations are treated. The temperature dependence of the standard thermodynamic quantities—energy, specific heat, free energy and entropy—is calculated for the $2d$ Ising model as well as for the $2d$ 10-state Potts model and the canonically re-weighted histograms are shown. Special attention is given to the analysis procedure, which has to be able to handle sums of very large numbers. This is done by using only the logarithms until finally the quotient of two such numbers is obtained. Jackknife [10] binning is used to minimize bias problems which occur in the re-weighting of the simulation data to canonical ensembles.

Using the provided Fortran code and following the instructions allows for a step by step reproduction of the figures and all other numerical results presented in this article. This could be a desirable standard for more involved simulations too. Some conclusions are given in the final Section 4.

2. Getting started

2.1. The Potts model

We introduce the Potts models on d-dimensional hypercubic lattices with periodic boundary conditions. For this paper we stay close to the notation used in the accompanying computer programs and define the energy E via the action variable

$$\texttt{iact} = \sum_{\langle ij\rangle} \delta\big(q_i^{(k)}, q_j^{(k)}\big), \quad E = \frac{2dN}{q} - 2\texttt{iact}, \tag{2}$$

where $\delta(q_i, q_j)$ is the Kronecker delta and N the number of sites of the lattice. The sum $\langle ij\rangle$ is over the nearest neighbor lattice sites and $q_i^{(k)}$ is the *Potts state* of configuration k at site i. For the q-state Potts model $q_i^{(k)}$ takes on the values $1, \ldots, q$. As the variable `iact` takes on integer values, it allows for convenient histograming of its values during the updating process. Occasionally, we use the related mean values

$$\texttt{actm} = \texttt{iact}/(dN) \quad \text{and} \quad e_s = E/N. \tag{3}$$

Each configuration (microstate of the system) k defines a particular arrangements of all states at the sites and, vice versa, each arrangement of the states at the sites determines uniquely a configuration:

$$k = \big\{q_1^{(k)}, \ldots, q_N^{(k)}\big\}. \tag{4}$$

The expectation value of an observable O is defined by

$$\langle O\rangle = Z^{-1} \sum_{k=1}^{K} O^{(k)} \mathrm{e}^{-\beta E^{(k)}}, \tag{5}$$

where the sum is over all microstates and the partition function $Z = Z(\beta)$ normalizes the expectation value of the unit operator to $\langle 1\rangle = 1$. As there are q possible Potts states at each site, the total number of microstates is

$$Z(\beta = 0) = q^N. \tag{6}$$

Including $\beta = 0$ in a MUCA simulation allows for the normalization of the partition function necessary to calculate the canonical free energy and the entropy as a function of the temperature.

Our definition (5) of β agrees with the one commonly used for the Ising model [11], but disagrees by a factor of two with the one used for the Potts model in [3,9]:

$$\beta = \beta^{\text{Ising}} = \beta^{\text{Potts}}/2. \tag{7}$$

For the $2d$ Potts models a number of exact results are known in the infinite volume limit. The critical temperature [9] is for $q = 2, 3, \ldots$

$$\frac{1}{2}\beta_c^{\text{Potts}} = \beta_c = \frac{1}{T_c} = \frac{1}{2}\ln(1 + \sqrt{q}\,). \tag{8}$$

The phase transition is second order for $q \leqslant 4$ and first order for $q \geqslant 5$. At β_c the average energy per Potts state is [9]

$$-e_s^c = 2 + 2/\sqrt{q}, \tag{9}$$

where, by reasons of consistency with the Ising model notation, also our definition (3) of e_s differs by a factor of two from the one used in most of the Potts model literature. For the first order transitions at $q \geqslant 5$, Eq. (9) gives the average of the limiting energies from the ordered and the disordered phase. The exact infinite volume latent heats $\triangle e_s$ and the entropy jumps $\triangle s$ were also calculated by Baxter [9], whereas the interfacial tensions f_s were derived more recently [3].

2.2. The Fortran code

Fig. 1 shows the directory tree in which the Fortran routines are stored. `MUCA` is the parent directory and on the first level we have the directories `Exercises`, `ForLib`, `ForProg` and `Work`. The master code is provided in the directories `ForLib` and `ForProg`. `ForLib` contains the source code of a library of functions and subroutines. The library is closed in the sense that no reference to nonstandard functions or subroutines outside the library is ever made. The master versions of the main programs and certain routines (which need input from the parameter files discussed below) are contained in the subdirectory `ForProg`. The demonstrations of this article are contained in the subdirectories of `Exercises`.

To download the code, start with the URL `www.hep.fsu.edu/~berg` and click the `Research` link, then the link `Multicanonical Simulations`. On this page follow the link `Fortran Code` and get either the file `muca.tar` (399 KB) or the file `muca.tgz` (40 KB). On most Unix platforms you obtain the directory structure of Fig. 1 from `muca.tar` by typing

```
tar -xvf muca.tar                                    (10)
```

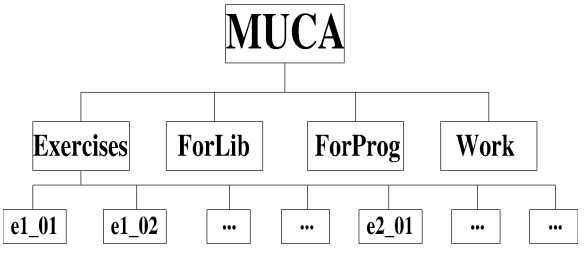

Fig. 1. Fortran code directory structure.

alternatively from `muca.tgz` by typing either `tar -zxvf muca.tgz` or `gunzip muca.tgz` followed by (10).

You obtain the results of this paper by compiling and running the code prepared in subdirectories of `Exercises`, e.g., `f77 -0 program.f` followed by `./a.out`. Due to the `include` and `parameter` file structure used, the programs and associated routines of `ForProg` compile only in subdirectories which are two levels down from the `MUCA` parent directory. This organization should be kept, unless you have strong reasons to change the dependencies. The present structure allows to create `Work` directories for various projects, with the actual runs done in the `Work` subdirectories. Note that under MS Windows with the (no longer marketed) MS Fortran compiler a modification of the `include` structure of the code turned out to be necessary. If such problems are encountered, one solution is to copy all needed files to the subdirectory in question and to modify all include statements accordingly.

Each `Exercises` subdirectory contains a

```
readme.txt                                           (11)
```

file with instructions, which should be followed. The subdirectories `e1`... prepare some code to check for the correctness of the Metropolis code for conventional, canonical simulations. The subdirectories `e2`... prepare examples of multicanonical simulations, which are discussed in the next section. The simulation parameters are set in the files

```
lat.par, lat.dat, potts.par,
mc.par and muca.par,                                 (12)
```

or a subset thereof, which are kept in `ForProg` and in each of the subdirectories of `Exercises`. The dimension `nd` of the system and the maximum lattice length `ml` are defined in `lat.par`. In `lat.dat` the lattice lengths for all directions are assigned to the array `nla`, which is of dimension `nd`. This allows for asymmetric lattices. The number of Potts states, `nq`, is defined in `potts.par`. The parameters of the conventional Metropolis simulation are defined in `mc.par`: These are the β value `beta`, which defines the initial weights in case of a MUCA simulation, the number of equilibrium sweeps `nequi`, the number of measurement repetitions `nrpt` and the number of

measurement sweeps `nmeas`. Additional parameters of the MUCA recursion are defined in `muca.par`: The maximum number of recursions `nrec_max`, the number of sweeps between recursions `nmucasw` and the maximum number of tunnelings (16) `maxtun`, which terminates the recursion unless `nrec_max` is reached first.

Whenever data for a graphical presentation are generated by our code, it is in a form suitable for *gnuplot*, which is freely available. Gnuplot driver files `fln.plt` are provides in the solution directories, such that one obtains the plot by typing `gnuplot fln.plt` on Unix and Linux platforms (under MS Windows follow the gnuplot menu).

3. Multicanonical simulations

A conventional, canonical simulation calculates expectation values at a fixed temperature T and can, by re-weighting techniques, only be extrapolated to a vicinity of this temperature [12]. In contrast, a single MUCA simulation allows to obtain equilibrium properties of the Gibbs ensemble over a range of temperatures, which would require many canonical simulations. This coined the name multicanonical. The MUCA method requires two steps:

(i) Obtain a *working estimate* $w_{\mathrm{mu}}(k)$ of the weights $w_{1/n}(k)$. Working estimate means that the approximation to (1) has to be good enough to ensure movement in the desired energy range, but deviations of $w_{\mathrm{mu}}(E)$ from (1) by a factor of, say, ten are tolerable.
(ii) Perform a Markov chain MC simulation with the final, fixed weights $w_{\mathrm{mu}}(k)$. Canonical expectation values are found by re-weighting to the Gibbs ensemble.

To obtain working estimates $w_{\mathrm{mu}}(k)$ of the weight factors (1), a slightly modified version of the recursion of Ref. [13] is used. As the analytical derivation of the modified recursion has so far only been published in conference proceedings, it is for the sake of completeness included in Appendix A of this paper. The Fortran implementation is given by the subroutine

`p_mu_rec.f` (13)

of `ForProg`. One subtlety is that two histogram arrays `hup` and `hdn` are introduced to keep separately track of the use of upper and lower entries of nearest neighbor pairs. Detailed explanations of the code will be part of a book [14].

The question whether more efficient recursions exists is far from being settled. For instance, F. Wang and Landau [15] made recently an interesting proposal. Exploratory comparisons with the recursion used in the present paper reveal similar efficiencies [16] for getting a working estimate of the multicanonical weights.

In between the recursion steps the Metropolis updating routine

`potts_met_f` (14)

is called, which implements the standard Metropolis algorithm [1] for general weights. The random number generator of Marsaglia et al. [17] is implemented to ensure identical results on distinct platforms.

3.0.1. *Example runs*

First, we illustrate the MUCA recursion for the 20^2 Ising model. We run the recursion in the range

$$\texttt{namin} = 400 \leqslant \texttt{iact} \leqslant 800 = \texttt{namax}. \tag{15}$$

These values of `namin` and `namax` are chosen to cover the entire range of temperatures, from the completely disordered ($\beta = 0$) region to the groundstate ($\beta \to \infty$). In many applications the actual range of physical interest is smaller, `namin` and `namax` should correspondingly be adjusted, because the recursion time increases quickly with the range. The recursion is completed after `maxtun` tunneling events have been performed. A *tunneling event* is defined as an updating process which finds its way from

$$\texttt{iact} = \texttt{namin} \quad \text{to} \quad \texttt{iact} = \texttt{namax} \quad \text{and back.} \tag{16}$$

This notation comes from the applications of the method to first order phase transitions [2], for which `namin` and `namax` are separated by a free energy barrier in the canonical ensemble. Although the MUCA method removes this barrier, the terminus tunneling was kept. The requirement that the process tunnels also back is included in the definition, because a one way tunneling is not indicative for the convergence

of the recursion. Most important, the process has still to tunnel when the weights are frozen for the second stage of the simulations.

Note that things work differently for the Wang–Landau recursion [15]. Obviously, it can also be considered as a method for obtaining a working estimate of the multicanonical parameters (equivalently the spectral density). Once the recursion is stopped, one may perform a Markov chain MC simulation with the estimated parameters, i.e. the second part of the multicanonical approach. As before, a litmus test for the success of the recursion is whether tunneling is achieved using the estimated weights. The Wang–Landau recursion tunnels already in its initial stage, but this is not indicative for being close to a working estimate of the spectral density, which in most cases is obtained when the recursion is run long enough [16]. Although the Wang–Landau recursion was originally proposed to estimate the spectral density directly, I would recommend to switch to refining the spectral density by ordinary MC simulations, once a working estimate of the multicanonical parameters becomes available. Namely, for Metropolis sampling convergence is proven, whereas this is not the case for either of the recursions (and definitely not true for the recursion of this paper).

For most applications ten tunnelings during our recursion part lead to acceptable weights. If the requested number of tunnelings is not reached after a certain maximum number of recursion steps, the problem will disappear in most cases by rerunning (eventually several times) with different random numbers. Otherwise, the number of sweeps between recursions should be enlarged, because our recursion is strictly only valid when the system is in equilibrium. One may even consider to discard some sweeps after each recursion step to reach equilibrium, but empirical evidence indicates that the improvement (if any) does not warrant the additional CPU time. The disturbance of the equilibrium is weak when the weight function approaches its fixed point. In the default setting of our programs we take the number of sweeps between recursion steps to be $10q$ divided by the acceptance rate, because an equal number of accepted moves is a better relaxation criterion than an equal number of sweeps.

In the subdirectory `e2_01` of `Exercises` a 20×20 lattice Ising model simulation is prepared for which we request ten tunneling events. We find them after 787 recursions and 64,138 sweeps, corresponding to an average acceptance rate of `20*787/64138=0.245` (the acceptance rate can be calculated this way, because the number of accepted sweeps triggers the recursion). Almost half of the sweeps are spent to achieve the first tunneling event. Subsequently, an MUCA production run of 10,000 equilibrium and $32 \times 10{,}000$ sweeps with measurements is carried out. On a GHz Linux PC the entire runtime (recursion plus production) is about thirty seconds. In the subdirectory `e2_02` a similar simulation is prepared for the $2d$ 10-state Potts model on a 20^2 lattice.

3.1. Re-weighting to the canonical ensemble

Let us assume that we have performed a MUCA simulation which covers the action histogram needed for a temperature range

$$\beta_{\min} \leqslant \beta \leqslant \beta_{\max}. \tag{17}$$

In practice this means that the parameters `namax` and `namin` in `muca.par` have to be chosen such that

$$\texttt{namin} \ll \overline{\text{act}}(\beta_{\min}) \quad \text{and} \quad \overline{\text{act}}(\beta_{\max}) \ll \texttt{namax}$$

holds, where $\overline{\text{act}}(\beta)$ is the canonical expectation value of the action variable (2). The $\ll$ conditions may be relaxed to equal signs, if $b(\texttt{iact}) = \beta_{\min}$ is used for all action values iact $\leqslant \overline{\text{act}}(\beta_{\min})$ and $b(\texttt{iact}) = \beta_{\max}$ for all action values iact $\geqslant \overline{\text{act}}(\beta_{\max})$.

Given the MUCA time series, where $i = 1, \ldots, n$ labels the generated configurations, the definition (5) of the canonical expectation values leads to the MUCA estimator

$$\overline{O} = \frac{\sum_{i=1}^{n} O^{(i)} \exp[-\beta E^{(i)} + b(E^{(i)})E^{(i)} - a(E^{(i)})]}{\sum_{i=1}^{n} \exp[-\beta E^{(i)} + b(E^{(i)})E^{(i)} - a(E^{(i)})]}. \tag{18}$$

This formula replaces the MUCA weighting of the simulation by the Boltzmann factor of Eq. (5). The denominator differs from Z by a constant factor, which drops out because the numerator differs by the same constant factor from the numerator of (5). If only functions of the energy (in our computer programs the action variable) are calculated, it is sufficient to keep histograms instead of the entire time series. For an operator $O^{(i)} = f(E^{(i)})$ Eq. (18) simplifies then to

$$\overline{f} = \frac{\sum_E f(E) h_{\rm mu}(E) \exp[-\beta E + b(E)E - a(E)]}{\sum_E h_{\rm mu}(E) \exp[-\beta E + b(E)E - a(E)]}, \tag{19}$$

where $h_{\rm mu}(E)$ is the histogram sampled during the MUCA production run and the sums are over all energy values for which $h_{\rm mu}(E)$ has entries. When calculating error bars for estimates from Eqs. (18) or (19), we employ jackknife [10] estimators to reduce bias problems.

A computer implementation of Eqs. (18) and (19) requires care. The differences between the largest and the smallest numbers encountered in the exponents can be really large. To give one example, for the Ising model on a 100×100 lattice and $\beta = 0.5$ the groundstate configuration contributes $-\beta E = 10^4$, whereas for a disordered configuration $E = 0$ is possible. Clearly, overflow disasters will result, if we ask Fortran to calculate numbers like $\exp(10^4)$. When the large terms in the numerator and denominator take on similar orders of magnitude, one can avoid them by subtracting a sufficiently large number in all exponents of the numerator as well as the denominator, resulting in a common factor which divides out. Instead of overflows one encounters harmless underflows of the type $\exp(-10^4)$. We implement the idea in a more general fashion, which remains valid when the magnitudes of the numerator and the denominator disagree. We avoid altogether to calculate large numbers and deal only with the logarithms of sums and partial sums.

We first consider sums of positive numbers and discuss the straightforward generalization to arbitrary signs afterwards. For $C = A + B$ with $A > 0$ and $B > 0$ we calculate $\ln C = \ln(A + B)$ from the values $\ln A$ and $\ln B$, without ever storing either A or B or C. The basic observation is that

$$\begin{aligned} \ln C &= \ln\left[\max(A, B)\left(1 + \frac{\min(A, B)}{\max(A, B)}\right)\right] \\ &= \max(\ln A, \ln B) + \ln\bigl\{1 + \exp\bigl[\min(\ln A, \ln B) \\ &\qquad - \max(\ln A, \ln B)\bigr]\bigr\} \end{aligned} \tag{20}$$

holds. By construction the argument of the exponential function is negative, so that an underflow occurs when the difference between $\min(\ln A, \ln B)$ and $\max(\ln A, \ln B)$ becomes too big, whereas it becomes calculable when this difference is small enough.

To handle alternating signs one needs in addition to Eq. (20) an equation for $\ln|C| = \ln|A - B|$ where $A > 0$ and $B > 0$ still holds. Assuming $\ln A \neq \ln B$, Eq. (20) converts for $\ln|C| = \ln|A - B|$ into

$$\begin{aligned} \ln|C| &= \max(\ln A, \ln B) \\ &\quad + \ln\bigl\{1 - \exp\bigl[\min(\ln A, \ln B) \\ &\qquad - \max(\ln A, \ln B)\bigr]\bigr\} \end{aligned} \tag{21}$$

and, because the logarithm is a strictly monotone function, the sign of $C = A - B$ is positive for $\ln A > \ln B$ and negative for $\ln A < \ln B$.

The computer implementation of Eqs. (20) and (21) is provided by the Fortran function `addln.f` and the Fortran subroutine `addln2.f` of `ForLib`, respectively. The subroutines `potts_zln.f` and `potts_zln0.f` of `ForLib` rely on this to perform the jackknife re-weighting analysis for various physical observables.

3.2. Energy and specific heat calculations

We are now ready to analyze the MUCA data for the energy per spin of the $2d$ Ising model on a 20×20 lattice, which we compare in Fig. 2 with the exact results of Ferdinand and Fisher [8]. The code is prepared in the subdirectory `e2_03`.

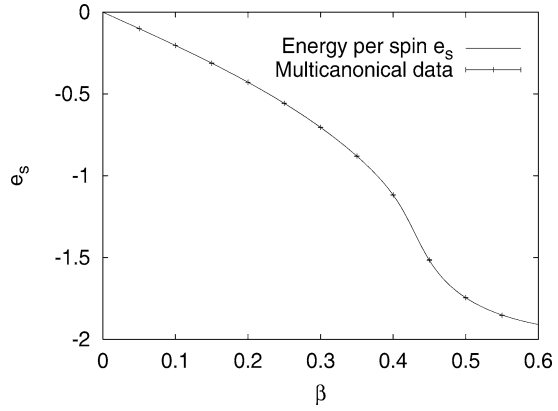

Fig. 2. Energy per spin e_s versus β for the $2d$ Ising model on an 20×20 lattice. Multicanonical data are compared with the exact result of Ferdinand and Fisher (full line), see the subdirectory `e2_02`.

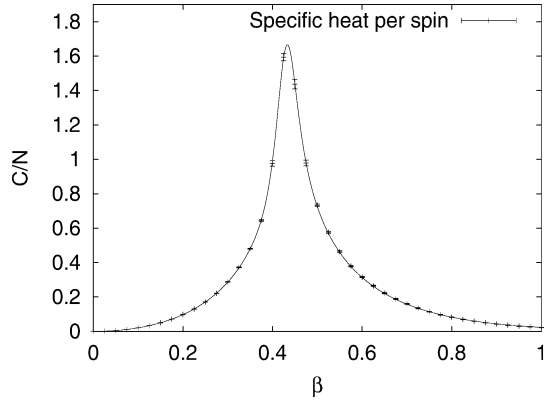

Fig. 3. Specific heat per spin versus β for the $2d$ Ising model on a 20×20 lattice. Multicanonical data are compared with the exact result of Ferdinand and Fisher (full line), see the subdirectory `e2_03`.

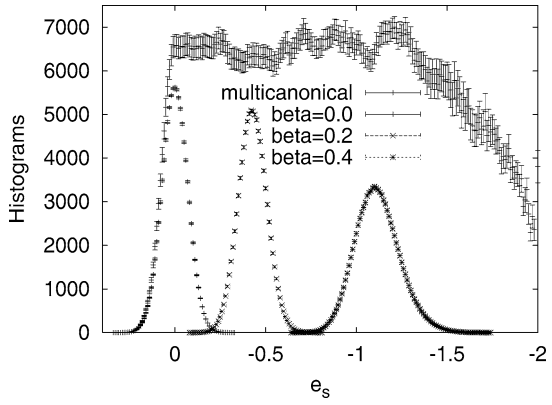

Fig. 4. Energy histogram from a multicanonical simulation of the $2d$ Ising model on a 20×20 lattice together with the canonically re-weighted histograms at $\beta = 0$, $\beta = 0.2$ and $\beta = 0.4$, see the subdirectory `e2_03`.

The same numerical technique allows us to calculate the *specific heat*, which is defined by

$$C = \frac{d\widehat{E}}{dT} = \beta^2\big(\langle E^2\rangle - \langle E\rangle^2\big). \tag{22}$$

Fig. 3 compares the thus obtained MUCA data with the exact results of Ferdinand and Fischer [8]. The code is also prepared in subdirectory `e2_03`.

Fig. 4 shows the energy histogram of the MUCA simulation together with its canonically re-weighted descendants at $\beta = 0$, $\beta = 0.2$ and $\beta = 0.4$. The Fortran code is prepared in the subdirectory `e2_04`. The normalization of the MUCA histogram is adjusted

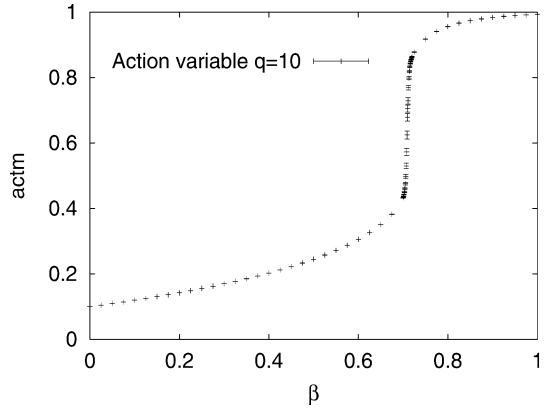

Fig. 5. Multicanonical mean action variable (3) data for the $2d$ 10-state Potts model on a 20×20 lattice, see the subdirectory `e2_02`.

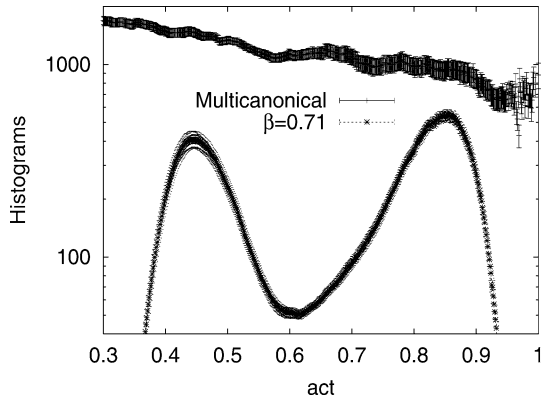

Fig. 6. Action histogram from a multicanonical simulation of the $2d$ 10-state Potts model on a 20×20 lattice together with the canonically re-weighted histograms at $\beta = 0.71$, see the subdirectory `e2_06`.

so that it fits reasonably well into one figure with the three re-weighted histograms. In Fig. 4 it is remarkable that the error bars of the canonically re-weighted histograms are not just the scale factors times the error bars of the MUCA histogram, but in fact much smaller. This can be traced to be an effect of the normalization. The sum of each canonical jackknife histogram is normalized to the same number and this reduces the spread.

Relying on the $2d$ 10-state Potts model data of the run prepared in subdirectory `e2_02`, we reproduce in subdirectory `e2_05` the action variable `actm` results plotted in Fig. 5. Around $\beta = 0.71$ we observe a sharp increase of `actm` from 0.433 at $\beta = 0.70$ to 0.864 at

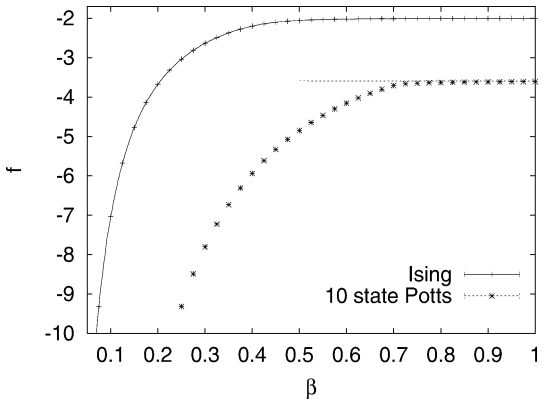

Fig. 7. Free energies from multicanonical simulations of the $2d$ Ising and $2d$ 10-state Potts models on a 20×20 lattice, see subdirectories `e2_03` and `e2_05`. The lines are the exact results of Ferdinand and Fischer for the Ising model and the asymptotic equation (26) for the 10-state Potts model.

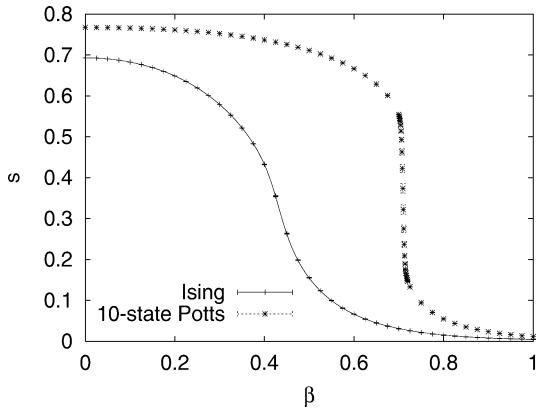

Fig. 8. Entropies from multicanonical simulations of the $2d$ Ising and $2d$ 10-state Potts models on a 20×20 lattice, see the subdirectories `e2_03` and `e2_05`. The full line is the exact result of Ferdinand and Fischer for the Ising model.

$\beta = 0.72$, which signals the first order phase transition of the model.

In Fig. 6 we plot the canonically re-weighted histogram at $\beta = 0.71$ together with the MUCA histogram using suitable normalizations, as prepared in subdirectory `e2_06`. The ordinate of Fig. 6 is on a logarithmic scale and the canonically re-weighted histogram exhibits the double peak structure which is characteristic for first order phase transitions. The MUCA method allows then to estimate the interface tension of the transition by calculating the minimum to maximum ratio on larger lattices, see [2,6].

3.3. Free energy and entropy calculations

At $\beta = 0$ the Potts partition function Z is given by Eq. (6). MUCA simulations allow for proper normalization of the partition function by including $\beta = 0$ in the temperature range (17). The normalized partition function yields important quantities of the canonical ensemble, the *Helmholtz free energy*

$$F = -\beta^{-1} \ln(Z) \tag{23}$$

and the *entropy*

$$S = \beta(F - E), \tag{24}$$

where E is the internal energy (2).

Fig. 7 shows the free energy density per site

$$f = F/N \tag{25}$$

for the $2d$ Ising model as well as for the $2d$ 10-state Potts model. The Ising model analysis is prepared in the subdirectory `e2_03` and the $q = 10$ analysis in `e2_05`. As in previous figures, the Ising model data are presented together with the exact results, whereas we compare the 10-state Potts model data with the $\beta \to \infty$ asymptotic behavior. For large β the partition function of our q-state Potts models approaches $q \exp(+2\beta dN - 2\beta dN/q)$ and, therefore,

$$f_{\text{as}} = \frac{2d}{q} - 2d - \beta^{-1} \frac{\ln(q)}{N}. \tag{26}$$

Finally, in Fig. 8 we plot for the entropy density

$$s = S/N \tag{27}$$

the $2d$ Ising model the MUCA results together with the exact curve. In addition, entropy data for the $2d$ 10-state Potts model are included in this figure. In that case we use $s/3$ instead of s, such that both graphs fit into the same figure. The analysis code for the entropy is contained in the same subdirectories as used for the free energy.

In all the figures of this section excellent agreement between the numerical and the analytical results is found.

4. Conclusions

There are many ways to extend the multicanonical simulations of this paper. The interested reader is

simply referred to the literature [6]. The purpose of this article is to provide a start-up kit for the computer implementation of some of the relevant steps. There appears to be need for this, because to get the first program up and running appears to be a major stumbling block in the way of using the method.

In the opinion of the author, multicanonical simulations have the potential to replace canonical simulations as the method of first choice for studies of small to medium-sized systems. As seen here, once the recursion necessary for the first part of a MUCA simulation is programmed, the entire thermodynamics of the system follows from the second part of the simulation. However, the slowing down for larger system sizes is rather severe.

Quite a number of similar methods exists, see [6] for a summary. A sound comparison would require that the goals of the simulations and their benchmarks are defined first. So far the community has not set such standards.

Acknowledgements

I would like to thank Alexander Velytsky for useful discussions and for contributing Fig. 1. This work was in part supported by the U.S. Department of Energy under the contract DE-FG02-97ER41022.

Appendix A. Weight recursion

We first discuss the weights (1). By definition, the microcanonical temperature is

$$b(E) = \frac{1}{T(E)} = \frac{\partial S(E)}{\partial E} \tag{A.1}$$

and we define the dimensionless, microcanonical free energy by

$$\begin{aligned} a(E) &= \frac{F(E)}{T(E)} = \frac{E}{T(E)} - S(E) \\ &= b(E)E - S(E). \end{aligned} \tag{A.2}$$

It is determined by relation (A.1) up to an (irrelevant) additive constant. We consider the case of a discrete minimal energy ϵ and choose

$$b(E) = \big[S(E+\epsilon) - S(E)\big]/\epsilon \tag{A.3}$$

as the definition of $b(E)$. The identity

$$S(E) = b(E)E - a(E)$$

implies

$$\begin{aligned} &S(E) - S(E-\epsilon) \\ &\quad = b(E)E - b(E-\epsilon)(E-\epsilon) - a(E) + a(E-\epsilon). \end{aligned}$$

Inserting $\epsilon b(E-\epsilon) = S(E) - S(E-\epsilon)$ yields

$$a(E-\epsilon) = a(E) + \big[b(E-\epsilon) - b(E)\big]E \tag{A.4}$$

and $a(E)$ is fixed by defining $a(E_{\max}) = 0$. Once $b(E)$ is given, $a(E)$ follows.

A convenient starting condition for the initial ($n = 0$) simulation is

$$b^0(E) = 0 \quad \text{and} \quad a^0(E) = 0, \tag{A.5}$$

because the system moves freely in the disordered phase. Other $b^0(E)$ choices are of course possible. Our Fortran implementation allows for $b^0(E) = \beta$ with β defined in the parameter file `mc.par`.

The energy histogram of the nth simulation is given by $H^n(E)$. To avoid $H^n(E) = 0$ we replace for the moment

$$H^n(E) \to \widehat{H}^n(E) = \max\big[h_0, H^n(E)\big], \tag{A.6}$$

where h_0 is a number $0 < h_0 < 1$. Our final equations allow for the limit $h_0 \to 0$. In the following subscripts $_0$ are used to indicate quantities which are not yet our final estimators from the nth simulation. We define

$$w_0^{n+1}(E) = \mathrm{e}^{-S_0^{n+1}(E)} = c\frac{w^n(E)}{\widehat{H}^n(E)},$$

where the (otherwise irrelevant) constant c is introduced to ensure that $S_0^{n+1}(E)$ is an estimator of the microcanonical entropy

$$S_0^{n+1}(E) = -\ln c + S^n(E) + \ln \widehat{H}^n(E). \tag{A.7}$$

Inserting this relation into (A.3) gives

$$\begin{aligned} &b_0^{n+1}(E) \\ &\quad = b^n(E) + \big[\ln \widehat{H}^n(E+\epsilon) - \ln \widehat{H}^n(E)\big]/\epsilon. \end{aligned} \tag{A.8}$$

The estimator of the variance of $b_0^{n+1}(E)$ is obtained from

$$\begin{aligned} \sigma^2\big[b_0^{n+1}(E)\big] = {} & \sigma^2\big[b^n(E)\big] + \sigma^2\big[\ln \widehat{H}^n(E+\epsilon)\big]/\epsilon \\ & + \sigma^2\big[\ln \widehat{H}^n(E)\big]/\epsilon. \end{aligned}$$

Now $\sigma^2[b^n(E)] = 0$ as $b^n(E)$ is the fixed function used in the nth simulation and the fluctuations are governed by the sampled histogram $H^n = H^n(E)$

$$\sigma^2\big[\ln(\widehat{H}^n)\big] = \sigma^2\big[\ln(H^n)\big] = \big[\ln(H^n + \triangle H^n) - \ln(H^n)\big]^2,$$

where $\triangle H^n$ is the fluctuation of the histogram, which is known to grow with the square root of the number of entries $\triangle H^n \sim \sqrt{H^n}$. Hence, under the assumption that autocorrelation times of neighboring histogram entries are identical, the equation

$$\sigma^2\big[b_0^{n+1}(E)\big] = \frac{c'}{H^n(E+\epsilon)} + \frac{c'}{H^n(E)} \tag{A.9}$$

holds, where c' is an unknown constant. The assumption would be less strong if it were made for the energy-dependent acceptance rate histogram instead of the energy histogram. In the present models the energy dependence of the acceptance rate is rather smooth between nearest neighbors and there is less programming effort when using only energy histograms. Eq. (A.9) shows that the variance is infinite when there is zero statistics for either histogram, $H^n(E) = 0$ or $H^n(E+\epsilon) = 0$. The statistical weight for $b_0^{n+1}(E)$ is inversely proportional to its variance and the over-all constant is irrelevant. We define

$$g_0^n(E) = \frac{c'}{\sigma^2[b_0^{n+1}(E)]} = \frac{H^n(E+\epsilon)\, H^n(E)}{H^n(E+\epsilon) + H^n(E)} \tag{A.10}$$

which is zero for $H^n(E+\epsilon) = 0$ or $H^n(E) = 0$. The nth simulation is carried out using $b^n(E)$. It is now straightforward to combine $b_0^{n+1}(E)$ and $b^n(E)$ according to their respective statistical weights into the desired estimator:

$$b^{n+1}(E) = \hat{g}^n(E) b^n(E) + \hat{g}_0^n(E) b_0^{n+1}(E), \tag{A.11}$$

where the normalized weights

$$\hat{g}_0^n(E) = \frac{g_0^n(E)}{g^n(E) + g_0^n(E)} \tag{A.12}$$

and

$$\hat{g}^n(E) = 1 - \hat{g}_0^n(E) \tag{A.13}$$

are determined by the recursion

$$g^{n+1}(E) = g^n(E) + g_0^n(E), \quad g^0(E) = 0. \tag{A.14}$$

We can eliminate $b_0^{n+1}(E)$ from Eq. (A.11) by inserting its definition (A.8) and get

$$b^{n+1}(E) = b^n(E) + \hat{g}_0^n(E) \times \big[\ln \widehat{H}^n(E+\epsilon) - \ln \widehat{H}^n(E)\big]/\epsilon. \tag{A.15}$$

Notice that it is now save to perform the limit $h_0 \to 0$. Finally, Eq. (A.15) can be converted into a direct recursion for ratios of the weight factor neighbors. We define

$$R^n(E) = \mathrm{e}^{\epsilon b^n(E)} = \frac{w^n(E)}{w^n(E+\epsilon)} \tag{A.16}$$

and get

$$R^{n+1}(E) = R^n(E) \left[\frac{\widehat{H}^n(E+\epsilon)}{\widehat{H}^n(E)}\right]^{\hat{g}_0^n(E)}. \tag{A.17}$$

References

[1] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087.
[2] B.A. Berg, T. Neuhaus, Phys. Rev. Lett. 68 (1992) 9; B.A. Berg, Int. J. Mod. Phys. C 3 (1992) 1083.
[3] C. Borgs, W. Janke, J. Phys. I (France) 2 (1992) 2011, and references therein.
[4] G.M. Torrie, J.P. Valleau, J. Comp. Phys. 23 (1977) 187.
[5] T. Neuhaus, J.S. Hager, cond-mat/0201324, submitted to J. Stat. Phys.
[6] B.A. Berg, Comp. Phys. Commun. 104 (2002) 52.
[7] F.Y. Wu, Rev. Mod. Phys. 54 (1982) 235.
[8] A.E. Ferdinand, M.E. Fisher, Phys. Rev. 185 (1969) 832.
[9] R.J. Baxter, J. Phys. C 8 (1973) L445.
[10] R. Miller, Biometrika 61 (1974) 1.
[11] K. Huang, Statistical Mechanics, 2nd edn., John Wiley & Sons, New York, 1987.
[12] A. Ferrenberg, R. Swendsen, Phys. Rev. Lett. 61 (1988) 2635.
[13] B.A. Berg, J. Stat. Phys. 82 (1996) 323–342 (The modifications of this recursion rely on work with Wolfhard Janke).
[14] B.A. Berg, Introduction to Markov Chain Monte Carlo Simulations and Their Statistical Analysis, in preparation.
[15] F. Wang, D. Landau, Phys. Rev. Lett. 86 (2001) 2050–2053.
[16] T. Nagasima, Y. Sugita, A. Mitsutake, Y. Okamoto, in preparation; U. Hansmann, unpublished.
[17] G. Marsaglia, A. Zaman, W.W. Tsang, Stat. Prob. 8 (1990) 35.