Dynamical behavior of the Niedermayer algorithm applied to Potts models

D. Girardi [a,b,*], T.J.P. Penna [a,b,c], N.S. Branco [d]

[a] *Instituto de Ciências Exatas, Universidade Federal Fluminense, Rua Des. Ellis Hermydio Figueira, 783, Volta Redonda, RJ 27.213-350, Brazil*

[b] *National Institute of Science and Technology for Complex System, Brazil*

[c] *Instituto de Física, Universidade Federal Fluminense, Niterói, RJ, Brazil*

[d] *Departamento de Física, Universidade Federal de Santa Catarina, 88040-900, Florianópolis, SC, Brazil*

ARTICLE INFO

ABSTRACT

In this work, we make a numerical study of the dynamic universality class of the Niedermayer algorithm applied to the two-dimensional Potts model with 2, 3, and 4 states. This algorithm updates clusters of spins and has a free parameter, E_0, which controls the size of these clusters, such that $E_0 = 1$ is the Metropolis algorithm and $E_0 = 0$ regains the Wolff algorithm, for the Potts model. For $-1 < E_0 < 0$, only clusters of equal spins can be formed: we show that the mean size of the clusters of (possibly) turned spins initially grows with the linear size of the lattice, L, but eventually saturates at a given lattice size $\tilde{L}$, which depends on E_0. For $L \geq \tilde{L}$, the Niedermayer algorithm is in the same dynamic universality class of the Metropolis one, i.e, they have the same dynamic exponent. For $E_0 > 0$, spins in different states may be added to the cluster but the dynamic behavior is less efficient than for the Wolff algorithm ($E_0 = 0$). Therefore, our results show that the Wolff algorithm is the best choice for Potts models, when compared to the Niedermayer's generalization.

1. Introduction

Numerical simulations of physical systems have been used for decades (for a review of numerical methods in statistical physics, see Refs. [1,2]). In recent years, however, this method has received a renewed interest, due to the increase in computational power and, more important, due to the introduction of new algorithms. The developments in the algorithms have the goal to allow for more efficient simulations, in many different directions: the calculation of the density of states using flat histograms, which allows for obtaining information at any temperature from one single simulation [3]; the use of bitwise operations and storage, which increases by a great amount the speed of the simulation and saves memory [4]; the introduction of cluster algorithms, which updates collections of spins, decreasing the autocorrelation time and reducing critical slowing down [1,2,5,6].

The Metropolis algorithm [7], for example, which had been the main choice of algorithm for a long time, suffers from a severe critical slowing-down effect near critical points. This is in part due to the fact that it updates one spin each time (therefore, being in the general category of single-spin algorithms). Critical slowing down is measured through the dynamic critical exponent z, defined from the dependence of the autocorrelation time τ on the linear size of the simulated lattice, L, at the critical temperature T_c. This dependence is assumed to be in the form $\tau \sim L^z$. Therefore, smaller values of z lead to smaller autocorrelation times and more efficient algorithms, since more configurations can be used to calculate the

---

* Corresponding author at: Instituto de Ciências Exatas, Universidade Federal Fluminense, Rua Des. Ellis Hermydio Figueira, 783, Volta Redonda, RJ 27.213-350, Brazil. Tel.: +55 24 30768944.

E-mail address: girardi@if.uff.br (D. Girardi).

necessary averages. The Metropolis algorithm, for example, when applied to the Ising model in two dimensions, presents $z \sim 2.16$ [8].

One possible way to overcome the difficulty of critical slowing down is to design algorithms which update clusters of spins (the so-called cluster algorithms), that may have a much lower value of z: this is the case for the Swendsen and Wang [5] and Wolff [6] algorithms, for which z may be zero for the two-dimensional Ising model [9,10]. See also Ref. [11] for another possible dependence of the autocorrelation time on the lattice size L. An alternative (and generalization) to these last two cluster algorithms, the Niedermayer algorithm, was introduced some time ago [12] but, to the best of our knowledge, has only had its dynamic behavior studied in detail for the Ising and XY models [11]. Our goal in this work is to study in a systematic way the dynamic behavior of the Niedermayer algorithm, applied to Potts models in two dimensions, for number of states $q = 2, 3$, and 4, for some values of the free parameter E_0 (see below), in order to determine the best choice of this parameter for these models. Note also that the critical temperature for two-dimensional Potts models are exactly known [13], which allows for a more precise determination of z.

The Potts model is defined by the Hamiltonian [13]

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{S_i, S_j}, \tag{1}$$

where $S_i = 1, 2, \ldots, q$, $\forall i$, the sum is over nearest neighbors on a lattice (in our case, a square lattice) and δ_{S_i, S_j} is the Kronecker delta ($\delta_{S_i, S_j} = 1$, if $S_i = S_j$, and $\delta_{S_i, S_j} = 0$, if $S_i \neq S_j$). We treat the ferromagnetic case in this work, i.e, $J > 0$. In two dimensions, the phase transition for this model is a continuous one for $q \leq 4$. In our study, we restrict ourselves to this interval.

This work is organized as follows: in the next section we present the Niedermayer algorithm for the Potts model and its relation to Metropolis' and Wolff's. In Section 3 we review some features connected to the autocorrelation time and the dynamic exponent z, in Section 4 we present and discuss our results, and in the last section we summarize our findings.

2. The Niedermayer algorithm

The Niedermayer algorithm was introduced some time ago as a cluster algorithm, motivated by the possibility to diminish the value of the dynamic critical exponent. It may be seen as a generalization of the Wolff or the Swendsen–Wang algorithms. The main idea is to build a cluster of spins and accept its updating as a single entity, with a parameter E_0 which controls the nature and size of the cluster and its acceptance ratio, as explained below. In this work, we have chosen to build the clusters according to the Wolff algorithm (they can be constructed according to the Swendsen–Wang rule but the results will not differ qualitatively in two dimensions and in higher dimensions Wolff algorithm is superior to Swendsen–Wang's).

In the Niedermayer algorithm, a spin in the lattice, which we will call the *seed* spin, is randomly chosen, being the first spin of the cluster. First-neighbors of this spin may be considered part of the cluster, with a probability

$$P_{add}(E_{ij}) = \begin{cases} 1 - e^{K(E_{ij} - E_0)}, & \text{if } E_{ij} < E_0 \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $K = J/kT$, T is the temperature, J is the exchange constant, and E_{ij} is the energy between nearest-neighbor spins in unities of J (i.e, $E_{ij} = -\delta_{s_i s_j}$). First-neighbors of added spins may be added to the cluster, according to the probability given above. Each spin has more than one chance to be part of the cluster, since it may have more than one first-neighbor in it. When no more spins can be added, all spins in the cluster have their state changed to a new state with an acceptance ratio A. Assuming that, at the frontier of the cluster there are m bonds linking spins in the same state *in the old configuration* and n bonds linking spins in the same state *in the new configuration* (there are also p bonds in the border of the cluster which are in different states from their neighbors in the cluster in the old and in the new configurations, but they cancel out in the expression below), A satisfies:

$$\frac{A(a \rightarrow b)}{A(b \rightarrow a)} = \left[e^K \left(\frac{1 - P_{add}(-J)}{1 - P_{add}(0)} \right) \right]^{n-m}, \tag{3}$$

where $a \rightarrow b$ represents the possible updating process, from the "old" (a) to the "new" (b) state, which differ from the flipping of all spins in the cluster, and $b \rightarrow a$ represents the opposite move. This expression ensures that detailed balance is satisfied [2].

One has to consider three different intervals for E_0:

(i) for $-1 \leq E_0 < 0$, only spins in the same state as the seed may be added to the cluster, with probability $P_{add} = 1 - e^{-K(1+E_0)}$. The new state of the cluster is randomly chosen between the remaining $q - 1$ states. The acceptance ratio (Eq. (3)) cannot be chosen to be one always and is given by $A = e^{-KE_0(m-n)}$, if $n < m$ (i.e, if the energy increases when the spins in the cluster are changed), or by $A = 1$, if $n \geq m$ (i.e, if the energy decreases when the spins in the cluster are changed). If $E_0 = -1$, we obtain the Metropolis algorithm, since only one-spin clusters are allowed (according to Eq. (2), $P_{add} = 0$ for $E_0 = -1$) and the acceptance ratio is $A = e^{-K\Delta E}$ for positive ΔE and 1 otherwise, where $\Delta E = (m - n)$ is the difference in energy when the spin is changed, in units of J;

(ii) for $E_0 = 0$, again only spins in the same state can take part of the cluster, with probability $P_{add} = 1 - e^{-K}$. Now, the acceptance ratio can be chosen to be 1, i.e, the cluster of like spins is always changed. Again, the new state of the cluster is randomly chosen between the remaining $q - 1$ states. This is the celebrated Wolff algorithm;

(iii) for $E_0 > 0$, spins in different states may be part of the cluster. Consider a spin already in the cluster: the probability of adding one of its first-neighbors to the cluster is $P_{add} = 1 - e^{-K(1+E_0)}$ if they are in the same state or $P_{add} = 1 - e^{-KE_0}$ otherwise. The acceptance ratio is again always 1. To change the state, for each cluster, we randomly choose a Δq between 1 and $q - 1$ and perform a cyclic sum. Note that for $E_0 \gg 0$ nearly all spins will be in the cluster and the algorithm will be clearly inefficient (in fact, it will not be ergodic for $E_0 \to \infty$). Therefore, we expect that, if the optimal choice of E_0 is greater than 0, it will not be much greater than this value.

After constructing a cluster, possible updating it and calculating the relevant thermodynamic functions for the new configuration, the whole process is repeated with a new seed spin. In this way, the Markov chain of configurations is generated.

3. Autocorrelation time and dynamic exponent

To calculate the relevant averages from a numerical simulation, one has to build a Markov chain of spin configurations and use data from uncorrelated configurations along this chain. Therefore, one important quantity is the autocorrelation time for a given quantity, say $\Phi(t)$, obtained from the autocorrelation function $\rho(t)$:

$$
\rho(t) = \int \left[\Phi(t') - \langle\Phi\rangle\right]\left[\Phi(t' + t) - \langle\Phi\rangle\right] dt'
$$

$$
= \int \left[\Phi(t')\Phi(t' + t) - \langle\Phi\rangle^2\right] dt', \tag{4}
$$

Since time is a discrete quantity on Monte Carlo simulations, one has to discretize the above equation [2]:

$$
\rho(t) = \frac{1}{t_{max} - t} \sum_{t'=0}^{t_{max}-t} \left[\Phi(t')\Phi(t' + t)\right] - \frac{1}{(t_{max} - t)^2} \sum_{t'=0}^{t_{max}-t} \Phi(t') \times \sum_{t'=0}^{t_{max}-t} \Phi(t' + t). \tag{5}
$$

It is usually assumed that the autocorrelation function behaves, in its simplest form, as [2]:

$$
\rho(t) = A e^{-t/\tau}, \tag{6}
$$

This hypothesis has to be corroborated by data and, in some cases, more than one exponential term is required [14]. One point worth mentioning is that the autocorrelation function is not well behaved for long times, due to bad statistics (this is evident from Eq. (5), since few "measurements" are available for long times). Then, one has to choose the region where the straight line will be adjusted very carefully and it turns out that the value of τ so obtained is strongly dependent on this choice. One other possible way to measure τ is to integrate $\rho(t)$, assuming a single exponential dependence on (past and forward) time, and obtain:

$$
\tau = \frac{1}{2} \int_{-\infty}^{\infty} \frac{\rho(t)}{\rho(0)} dt, \tag{7}
$$

with:

$$
\rho(t) \equiv e^{-|t|/\tau}. \tag{8}
$$

Eq. (7), when discretized, leads to Ref. [15]:

$$
\tau = \frac{1}{2} + \sum_{t=1}^{\infty} \frac{\rho(t)}{\rho(0)}. \tag{9}
$$

The sum in the previous equation cannot be carried out for large values of t, since bad statistics would lead to unreliable results. In order to truncate the sum at some point, we use a cutoff (see Ref. [15] and references therein), defined as the value in time where the noise in the data is clearly greater than the signal itself. We then obtain a first estimate of τ using Eq. (9) and then make the integral of $\rho(t)/\rho(0)$ from the value of the cutoff to infinity. A criterion to accept the cutoff is that the value of this integral is smaller than the statistical uncertainty in calculating τ.

A note on the definition of "time" is worth stressing here. In the Metropolis algorithm, time is measured in Monte Carlo steps (*MCS*); one *MCS* is defined as the attempt to flip N spins, where N is the number of spins in the (finite) lattice being simulated (in our case, $N = L^2$, where L is the linear size of the lattice). For cluster algorithms, one unity of time is defined as the "time" taken to build and possibly change a cluster, t. A rescaling of this quantity is necessary, to be able to compare the results for different values of E_0, namely:

$$
t_{MCS} = t \frac{\langle n \rangle}{N}, \tag{10}
$$
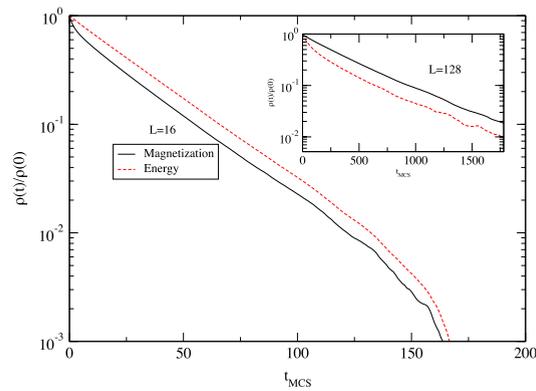
Fig. 1. Magnetization and energy autocorrelation functions versus time (in *MCS*) for the Niedermayer algorithm with $E_0 = -0.25$ and $q = 3$. The main window represents the behavior for linear size $L = 16$, while the inset applies to $L = 128$.

where t_{MCS} is the time measured in *MCS* and $\langle n \rangle$ is the mean cluster size. Note that, for Metropolis, $\langle n \rangle = 1$ and 1 *MCS* is the "time" taken to try to flip N spins, as usual. In this work, this rescaling has been done and all times are expressed in *MCS*.

Our first attempt was to fit the autocorrelation time to the expected behavior, namely $\tau \sim L^z$, in the critical region, where z is the dynamic exponent. We have taken as the chosen quantity to calculate τ the one which better adjusts to Eq. (6). Usually, different quantities lead to autocorrelation functions which behave differently as a function of time. A typical example is shown in Fig. 1, where both the magnetization and the energy autocorrelation functions for the $q = 3$ Potts model are depicted as functions of time, for the Niedermayer algorithm with $E_0 = -0.25$ and linear sizes $L = 16$ (main graph) and $L = 128$ (inset). The magnetization autocorrelation function presents an abrupt drop for small times and $L = 16$. This shows that this function is not properly described by a single exponential. On the other hand, the energy autocorrelation time follows a straight line even for the smallest times. Therefore, we should calculate τ from the latter, for $L = 16$, using Eq. (9). Note, however, that, when L is increased, the picture changes and now the magnetization autocorrelation function is well described by a single exponential (for small and intermediate values of time), as depicted in the inset of Fig. 1. Whenever a crossover like this is present, we measure the dynamic exponent from the behavior for large values of L and for the function which is well described by a single exponential for this range of L, using Eq. (9). The fact that, for intermediate values of t, the slopes of both curves in Fig. 1 (main graph and inset) appear to be the same, is an indication that the autocorrelation times for both are equal. However, we have already commented on the drawback of calculating τ from the slope of the autocorrelation function on a semi-log graph. As a final note, we would like to mention that we used helical boundary conditions and 20 independent runs (each with a different seed for the pseudo random number generator) were made for each E_0 and L. For each seed, at least 4×10^6 trial changes were made, in order to calculate the autocorrelation functions and their respective autocorrelation times. The values we quote are the average of the values obtained for each seed of the pseudo random number generator and the uncertainty in τ is the standard deviation of these 20 values.

4. Results and discussion

We have simulated the case $E_0 = -1$ (the Metropolis algorithm) as a test to our code. The result is presented in Fig. 2 for the magnetization autocorrelation time and, as we can see, all three cases have the same qualitative behavior. For $q = 2$ and 3 the dynamic exponent z takes approximately the same value ($z \simeq 2.16$), while for $q = 4$ it assumes a higher value, namely $z = 2.21 \pm 0.02$. These values are consistent with those in the literature [16].

From now on we will not comment on $q = 2$, since this case has been treated in Ref. [11]. Our simulation for $E_0 = -0.75$ is presented in Fig. 3, where the magnetization and energy autocorrelation times are depicted as functions of L. For $q = 3$ and $L \geq 32$ and for $q = 4$ and $L \geq 64$, the dynamic behavior (i.e., z) is the same as for the Metropolis algorithm. In Fig. 3(b) ($q = 3$) and Fig. 3(d) ($q = 4$) we present the behavior of the average cluster size $\langle n \rangle$ versus lattice size L: in both cases, for $L \geq \tilde{L} = 64$, $\langle n \rangle$ is constant. As already discussed, the autocorrelation function which is well described by only one exponential presents the greatest autocorrelation time and for, $E_0 = -0.75$, this happens for the magnetization's autocorrelation function. The dynamic exponent is calculated for $L \geq \tilde{L}$ and we obtain $z = 2.16 \pm 0.05$ and $z = 2.18 \pm 0.09$ for $q = 3$ and $q = 4$, respectively. These results agree, within error bars, with the values quoted in the literature [16] and with our values obtained for $E_0 = -1$.

For $E_0 = -0.25$, our result is depicted in Fig. 4 for $q = 3$ (the result for $q = 4$ is qualitatively the same); the behavior follows the same overall trend observed for $E_0 = -0.75$, with a different value of $\tilde{L}$. For $L < 128$ the autocorrelation time for the energy is greater than for the magnetization and the average cluster size $\langle n \rangle$ increases with L. For $L \geq \tilde{L} = 128$, the picture changes and the autocorrelation time for the magnetization is the greater one and $\langle n \rangle$ is constant. The value of z is consistent with the one for the Metropolis algorithm (again, z is calculated for $L \geq \tilde{L}$).
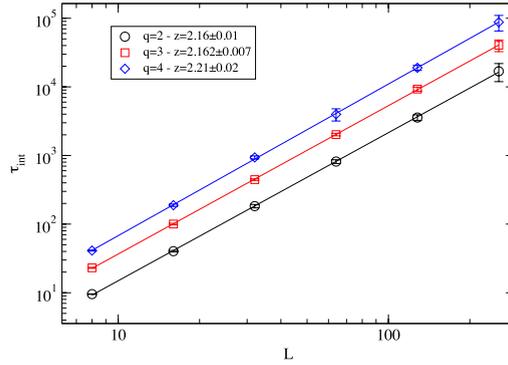
Fig. 2. Log–log graph of the magnetization autocorrelation time (in *MCS*) versus linear size *L* for the Metropolis algorithm with $q = 2$, 3 and 4. The quoted value for *z* is obtained from the slope of fitted straight lines.
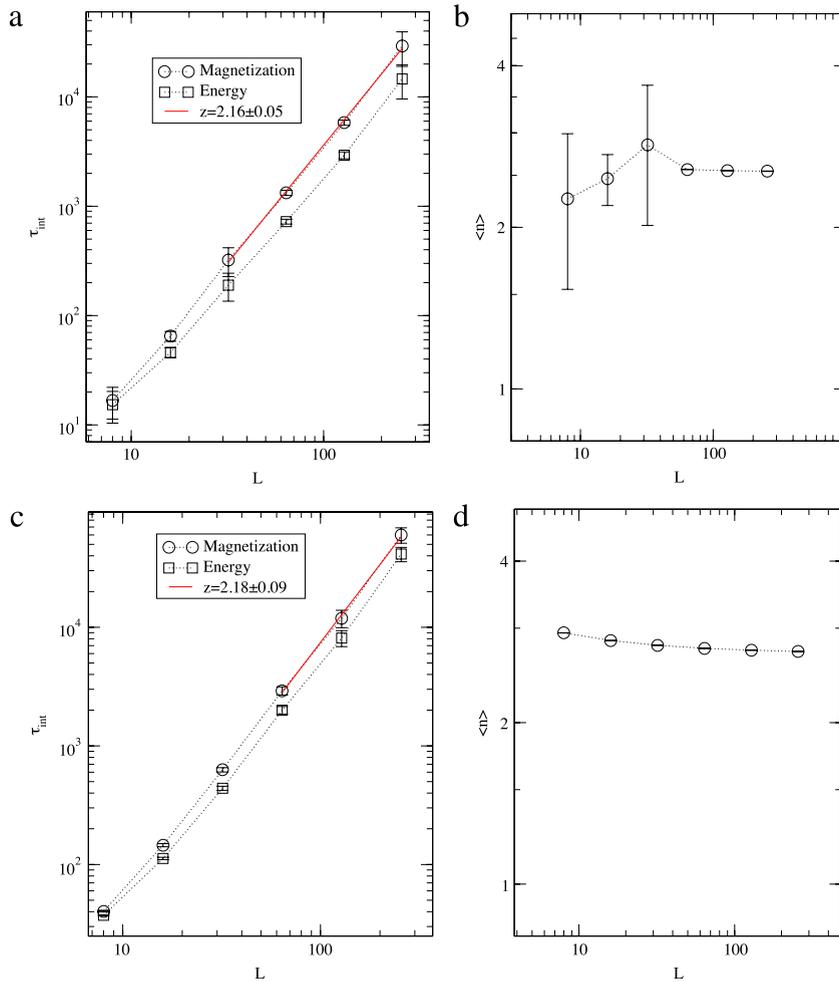
Fig. 3. Log–log graphs of magnetization (circle) and energy (square) autocorrelation times (in *MCS*) versus linear size *L* for the Niedermayer algorithm with $E_0 = -0.75$ for (a) $q = 3$ and (c) $q = 4$. Log–log graph of average cluster size $\langle n \rangle$ versus *L* for (b) $q = 3$ and (d) $q = 4$.

For other values of $-1 \leq E_0 < 0$ we observe the same qualitative behavior. There is always a $\tilde{L}$, such that, for $L < \tilde{L}$, $\langle n \rangle$ increases with *L* and, for $L > \tilde{L}$, $\langle n \rangle$ is constant and the magnetization's autocorrelation time is the one well described by a single exponential. The exponent *z*, always calculated for $L \geq \tilde{L}$, is the same as for the Metropolis algorithm. This can be linked to the constancy of $\langle n \rangle$ in this interval: since *L* increases and $\langle n \rangle$ remains the same, the fraction $\langle n \rangle / L^2$ decreases and the behavior is the same as a single-spin algorithm.
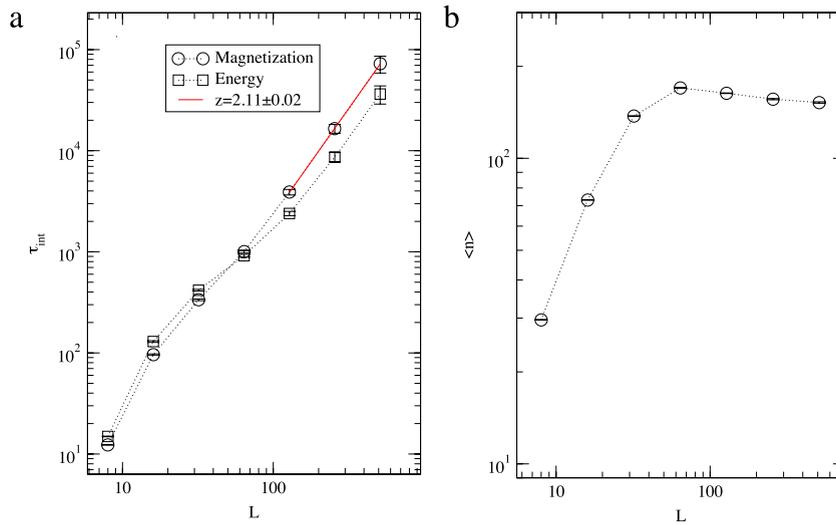
Fig. 4. (a) Log–log graphs of magnetization (circle) and energy (square) autocorrelation time (in *MCS*) versus linear size L for the Niedermayer algorithm with $E_0 = -0.25$ for $q = 3$. (b) Log–log graph of average cluster size $\langle n \rangle$ versus L for $q = 3$.

We simulated the Wolff algorithm to perform another check of our algorithm and to compare with our results for $E_0 \neq 0$. In Fig. 5 we present the autocorrelation time and $\langle n \rangle$ versus L for $q = 3$ and $q = 4$. To calculate z we used a different approach here [17,18]. We perform a power-law fitting using three consecutive lattice sizes (e.g $L = 512$, 1024, and 2048) and call L_{min} the smallest size. We then plot z versus $1/L_{min}$, as seen in Fig. 6 [17,18]. The power-law fitting for $q = 3$ is a good fit for all lattice sizes and we obtain $z = 0.55 \pm 0.02$, which agrees with a previous estimate [19]. For $q = 4$, only for $L \geq 128$ we obtain a good fit, with $z = 1.00 \pm 0.02$. This value is somewhat above a previous calculation [20] for the Swendsen–Wang algorithm. If we restrict our data to $L \leq 256$, as in the previous reference, we obtain $z = 0.94 \pm 0.01$, which just overlaps with the result of Ref. [20] (namely, $z = 0.89 \pm 0.05$). We plot the data for $q = 2$ for comparison with our results for $q = 3$ and 4: it seems clear that, for that value of q, the asymptotic regime has not yet been reached, for the lattice sizes we simulated (in fact, it is not clear if there is an asymptotic regime) [11]. We would like to note that a good indication of the result quality is the relation $\langle n \rangle \propto L^{\gamma/\nu}$. As we can see in Fig. 5(b) and (d), our estimates of γ/ν from a log–log plot of $\langle n \rangle$ vs. L (namely, $\gamma/\nu = 1.734 \pm 0.001$ and 1.7498 ± 0.0009 for $q = 3$ and 4, respectively) are, within error bars, the same of the (conjectured) exact results (namely, $\gamma/\nu = 26/15 = 1.73333\ldots$ for $q = 3$ [13] and $\gamma/\nu = 7/4 = 1.75$ for $q = 4$ [13,21]).

In order to compare our results for different values of E_0, we have plotted z vs. E_0 for both $q = 3$ and $q = 4$ (see Fig. 7): we see that z is approximately constant for any $E_0 \neq 0$ and strongly decreases for $E_0 = 0$. In fact, the subtle decrease of z as E_0 approaches 0 may be a crossover effect: the closer we get to the Wolff algorithm, the greater the value of $\tilde{L}$ and one has to go to larger and larger lattices to obtain the correct dynamic behavior.

Therefore, we have shown that the Wolff algorithm is still the most efficient procedure, when compared to the generalizations for $E_0 < 0$. But we still have to check the dynamic behavior for $E_0 > 0$. We know that for $E_0 \to \infty$ the algorithm is not ergodic. So we expect that, if a better value for E_0 exists, when compared to $E_0 = 0$, it is not much greater than this last value. To address this question we simulate the cases $E_0 = 0.05$ and 0.1. In Fig. 8 we show the results for $q = 3$ (a) and $q = 4$ (b), both calculated from the energy autocorrelation function. As we can see, for $q = 3$ the autocorrelation time τ for $E_0 = 0.05$ is much greater than for $E_0 = 0$ and grows faster than for the latter. For $q = 4$, τ is slightly greater for $E_0 = 0.05$ than for $E_0 = 0$, although our result is consistent with the same value of z for both cases. However, one has to consider that the implementation of the algorithm for $E_0 = 0.05$ is more complex than for $E_0 = 0$.

5. Conclusion

In this work we studied the Niedermayer algorithm applied to the two-dimensional Potts model with 2, 3, and 4 states. Our goal was to determine which value of E_0 leads to the optimal algorithm, i.e., to the smallest value of the dynamic exponent z. We observe that for $-1 \leq E_0 < 0$ there is a lattice size $\tilde{L}$, such that, for $L \geq \tilde{L}$, the average size of updated clusters is constant and the dynamic behavior of the algorithm is the same as for Metropolis'. The value of $\tilde{L}$ increases with E_0 and diverges for $E_0 = 0$ (the Wolff algorithm). When we look to the autocorrelation function, we notice that for $L < \tilde{L}$ the autocorrelation time of the energy is greater than for the magnetization and the opposite happens for $L \geq \tilde{L}$. As we show in Fig. 1, the quantity with greater autocorrelation time have the autocorrelation function well described by a single exponential. For $E_0 = 0$ we regain the Wolff algorithm.

In Table 1 we summarize our findings, which show that the Wolff algorithm, $E_0 = 0$, is more efficient than its generalization for $E_0 < 0$. There is still the possibility that some value of $E_0 > 0$ may present a lower value of z, when
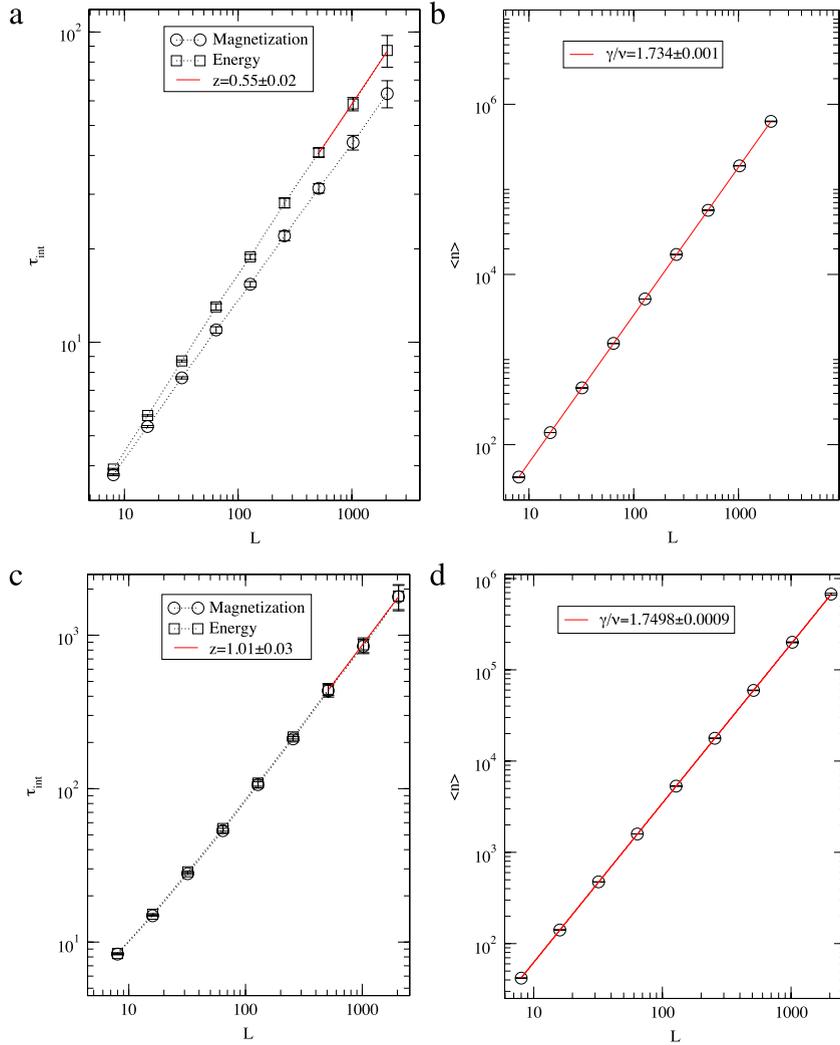
Fig. 5. Log–log graphs of magnetization (circle) and energy (square) autocorrelation time (in *MCS*) versus linear size *L* for the Niedermayer algorithm with $E_0 = 0.0$ (the Wolff algorithm) for (a) $q = 3$ and (c) $q = 4$. Note that the values we quote for the dynamic exponent z are those obtained from the greatest three values of *L* (see text). Log–log graph of average cluster size $\langle n \rangle$ versus *L* for (b) $q = 3$ and (d) $q = 4$.
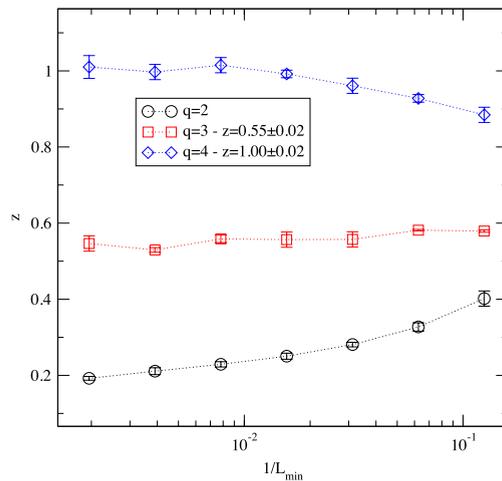
Fig. 6. Semi-log graphs of z versus $1/L_{min}$ (see the text) for $q = 2$, 3 and 4.
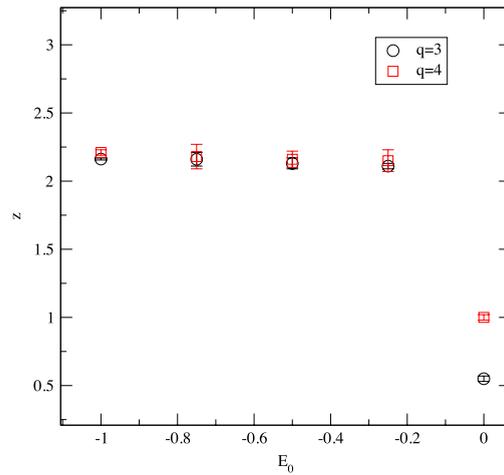
Fig. 7. Dependence of the dynamic exponent z on E_0, for $-1 \leq E_0 \leq 0$ and $q = 3$ (circles) and $q = 4$ (squares).
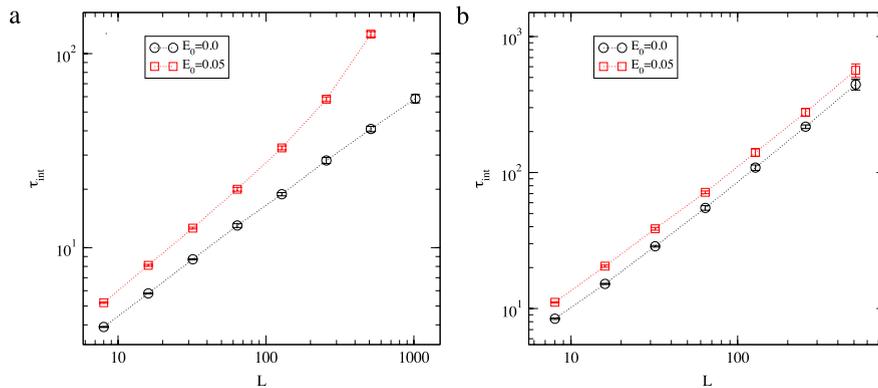
Fig. 8. Log–log graphs of energy autocorrelation time (in *MCS*) versus linear size L for the Niedermayer algorithm with $E_0 = 0$ (circle) and $E_0 = 0.05$ (square) for (a) $q = 3$ and (b) $q = 4$.

Table 1
Values for the dynamic exponent z for the three models studied here and for $-1 \leq E_0 \leq 0$.

q	$-1 \leq E_0 < 0$	Wolff ($E_0 = 0$)
2	2.16(1)	Undefined
3	2.162(7)	0.55(2)
4	2.21(2)	1.00(2)

compared to $E_0 = 0$. We show that, if this value exists, it is lower than $E_0 < 0.05$ and the complexity of the algorithm is greater than the improvement in the dynamic behavior.

Acknowledgments

References

[1] D.P. Landau, K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics, Cambridge University Press, New York, USA, 2000.
[2] M. Newman, G. Barkema, Monte Carlo Methods in Statistical Physics, Oxford University Press, USA, 1999.
[3] F.G. Wang, D.P. Landau, Phys. Rev. Lett. 86 (2001) 2050.
[4] T. Penna, P. de Oliveira, J. Stat. Phys. 61 (1990) 933.
[5] R.H. Swendsen, J.-S. Wang, Phys. Rev. Lett. 58 (1987) 86.
[6] U. Wolff, Phys. Rev. Lett. 62 (1989) 361.
[7] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087.
[8] M.P. Nightingale, H.W. Blöte, Phys. Rev. Lett. 76 (1996) 4548.
[9] C. Baillie, P. Coddington, Phys. Rev. B 43 (13) (1991) 10617.

[10] P.D. Coddington, C.F. Baillie, Phys. Rev. Lett. 68 (1992) 962.
[11] D. Girardi, N.S. Branco, J. Stat. Mech. Theory Exp. 2010 (2010) P04012.
[12] F. Niedermayer, Phys. Rev. Lett. 61 (1988) 2026.
[13] F.Y. Wu, Rev. Modern Phys. 54 (1982) 235.
[14] S. Wansleben, D.P. Landau, Phys. Rev. B 43 (1991) 6006.
[15] J. Salas, A.D. Sokal, J. Stat. Phys. 87 (1997) 1.
[16] K. Binder, J. Stat. Phys. 24 (1981) 69–86.
[17] D. Girardi, N.S. Branco, Phys. Rev. E 83 (2011) 061127.
[18] M. Picco, preprint. arxiv:cond-mat/9802092.
[19] P. Coddington, C. Baillie, Nuclear Phys. B Proc. Suppl. 26 (1992) 632.
[20] X. Li, A. Sokal, Phys. Rev. Lett. 63 (1989) 827.
[21] R.J. Creswick, S.Y. Kim, J. Phys. A. Math. Gen. 30 (1997) 8785.