PHYSICAL REVIEW D **71,** 114506 (2005)

Heat bath efficiency with a Metropolis-type updating

Alexei Bazavov[1,2] and Bernd A. Berg[1,2]

[1]*Department of Physics, Florida State University, Tallahassee, Florida 32306-4350, USA*

[2]*School of Computational Science, Florida State University, Tallahassee, Florida 32306-4120, USA*

(Received 13 March 2005; published 17 June 2005)

We illustrate for 4D $SU(2)$ and $U(1)$ lattice gauge theory that sampling with a biased Metropolis scheme is essentially equivalent to using the heat bath algorithm. Only, the biased Metropolis method can also be applied when an efficient heat bath algorithm does not exist. For the examples discussed the biased Metropolis algorithm is also better suited for parallelization than the heat bath algorithms.

DOI: 10.1103/PhysRevD.71.114506 PACS numbers: 05.10.Ln, 11.15.Ha

I. INTRODUCTION

The possibility of constructing biased Metropolis algorithms (BMAs) has been known for quite a while [1]. Although they have occasionally been used in the statistical physics [2] and biochemical [3] literature, it appears that practitioners of Markov chain Monte Carlo (MC) simulations have not given this topic the attention which it deserves. Reasons for this seem to be that (a) general situations for which BMAs are of advantage have not been clearly identified and (b) a lack of straightforward instructions about implementing such schemes.

On the other hand, the heat bath algorithm (HBA) is one of the widely used algorithms for MC simulations. It updates a variable with the Gibbs-Boltzmann probability defined by its interaction with the rest of the system (an introduction to HBAs can, e.g., be found in Ref. [4]). But, there exist energy functions for which an efficient heat bath implementation does not exist.

In this paper we show that a BMA similar to the one used for the rugged Metroplis method of Ref. [5], can be employed whenever one would normally think about constructing a HBA. When an efficient heat bath implementation exists, the performance of the HBA and the BMA will practically be identical. However, the BMA can still be constructed when the inversion of the cumulative distribution, required for a HBA, is numerically so slow that it is not a suitable option.

In the next section we illustrate our general observation for systems from lattice gauge theory. Our first example is 4D $SU(2)$ lattice gauge theory for which the HBA was first introduced by Creutz [6] and improved in Ref. [7,8]. Our second example is 4D $U(1)$ gauge theory.

II. BIASED METROPOLIS ALGORITHMS AND PURE LATTICE GAUGE THEORY

The action which we consider is

$$S(\{U\}) = \frac{1}{N_c}\sum_{\square} \mathrm{Re\,Tr}(U_{\square}), \tag{1}$$

$U_{\square} = U_{i_1 j_1} U_{j_1 i_2} U_{i_2 j_2} U_{j_2 i_1}$, where the sum is over all plaquettes of a 4D simple hypercubic lattice, i_1, j_1, i_2 and j_2 label the sites circulating about the plaquette and U_{ji} is a $U(1)$ or a $SU(2)$ matrix ($N_c = 1$ or 2) associated with the link $\langle ij\rangle$. The reversed link is associated with the inverse matrix. The aim is to calculate expectation values with respect to the Euclidean path integral

$$Z = \int \prod_{\langle ij\rangle} dU_{ij} e^{+\beta_g S(\{U\})}, \tag{2}$$

where the integrations are over the invariant group measure. While working at a particular link $\langle ij\rangle$, we need only to consider the contribution to S, which comes from the staples containing this link. If we denote by $U_{\sqcup,k}$, $k = 1, \ldots, 6$, the products which interact with the link in question, then the probability density of this link matrix is

$$dP(U) \sim dU \exp\left[\frac{\beta_g}{N_c}\mathrm{ReTr}\left(U\sum_{k=1}^{6} U_{\sqcup,k}\right)\right]. \tag{3}$$

A. $SU(2)$

We deal first with $SU(2)$ and parametrize the matrix elements in the form

$$U = a_0 I + i\vec{a}\cdot\vec{\sigma}, \qquad a_0^2 + \vec{a}^2 = 1, \tag{4}$$

where I denotes the 2×2 identity matrix and $\vec{\sigma}$ are the Pauli matrices. A property of $SU(2)$ group elements is that any sum of them is proportional to another $SU(2)$ element. We define a $SU(2)$ matrix $U_{\sqcup}$ which corresponds to the sum of the staples in Eq. (3) by

$$s_{\sqcup} U_{\sqcup} = \sum_{k=1}^{6} U_{\sqcup,k}, \qquad s_{\sqcup} = \sqrt{\det\left(\sum_{k=1}^{6} U_{\sqcup,k}\right)}. \tag{5}$$

Using the invariance of the group measure, one finds

$$dP(UU_{\sqcup}^{-1}) \sim d\Omega\, da_0 \sqrt{1-a_0^2}\exp(\beta_g s_{\sqcup} a_0), \tag{6}$$

where $d\Omega$ is the differential solid angle of $\vec{a}$. As it is

straightforward to generate the solid angle stochastically, the problem is reduced to sampling the probability density

$$P(a_0) \sim \sqrt{1 - a_0^2}\exp(\beta_g s_\sqcup a_0) \tag{7}$$

in the interval $-1 \le a_0 \le 1$. This is the starting point for the HBA, which amounts to finding a numerically fast inversion of the cumulative distribution function

$$F(a_0) = N_0 \int_{-1}^{a_0} da_0' \sqrt{1 - a_0'}\exp(\beta_g s_\sqcup a_0'), \tag{8}$$

where N_0 ensures the normalization $F(1) = 1$. The HBA updates a_0 by converting a uniformly distributed random number $0 \le x < 1$ into $a_0 = F^{-1}(x)$.

The remark of our paper is that a crude tabulation of the function $F(a_0)$ is entirely sufficient to obtain practically the same efficiency as with the HBA. Obviously, such a tabulation can still be done when there is no numerically efficient way to calculate $F^{-1}(x)$. The procedure does still generate the canonical probabilities of the continuous theory (2) without any approximation (except by the floating point precision and limitations of the random number generator).

Let us show how this works. First we choose a discretization of the parameter $s_\sqcup$, $0 \le s_\sqcup \le 6$, into m discrete values $s_\sqcup^i$, $i = 1, \ldots, m$ so that

$$0 < s_\sqcup^1 < s_\sqcup^2 < \ldots < s_\sqcup^m \tag{9}$$

holds. We take these values equidistant. Other partitions work too and could be more efficient. For each $s_\sqcup^i$ we calculate a table of values $a_0^{i,j}$, $j = 1, \ldots, n$ defined by

$$\frac{j}{n} = F(a_0^{i,j}; s_\sqcup^i) \tag{10}$$

and we also tabulate the differences

$$\triangle a_0^{i,j} = a_0^{i,j} - a_0^{i,j-1} \quad \text{for} \quad j = 1, \ldots, n, \tag{11}$$

where we define $a_0^{i,0} = -1$, and $a_0^{i,n} = +1$ follows from Eq. (10). For $\beta_g = 2.3$ this construction is shown in Fig. 1 using a representative $s_\sqcup^i$ value.

The biased Metropolis procedure for one update of a $SU(2)$ matrix consists now of the following steps:

(1) Find the $s_\sqcup^i$ value (only i is needed) which is nearest to the actual $s_\sqcup$ value.
(2) Place the present a_0 value on the discretization grid, i.e., find the integer j through the relation $a_0^{i,j-1} \le a_0 < a_0^{i,j}$.
(3) Pick an integer value j' uniformly distributed in the range 1 to n.
(4) Propose $a_0' = a_0^{i,j'-1} + x^r \triangle a_0^{i,j'}$, where $x^r, 0 \le x^r < 1$, is a uniformly distributed random number.

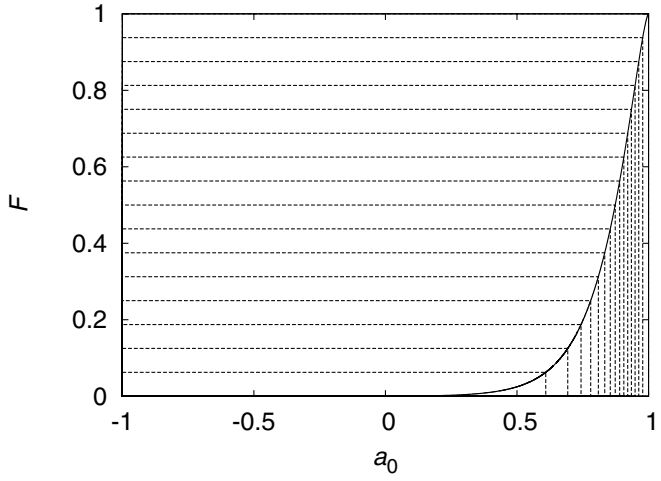

FIG. 1. Discretization of the cumulative distribution function $F(a_0; s_\sqcup^{11})$ for $SU(2)$ at $\beta_g = 2.3$ for the choices $m = 16$ (equidistant $s_\sqcup^i$ values, i.e., $s_\sqcup^{11} = 3.9375$) and $n = 2^4 = 16$.

(5) Accept a_0' with the probability

$$p_a = \frac{\exp(\beta_g s_\sqcup a_0') \triangle a_0^{i,j'}}{\exp(\beta_g s_\sqcup a_0) \triangle a_0^{i,j}}. \tag{12}$$

(6) If a_0' is accepted, calculate a random value for $\vec{a}'$ with the measure $d\Omega$ and store the new $SU(2)$ matrix. Otherwise keep the old $SU(2)$ matrix. After this step the configuration has to be counted independently of whether a_0' was accepted or rejected.

For i given each interval on the a_0 abscissa of Fig. 1 is proposed with probability $1/n$. In the limit $n > m, m \to \infty$ these are by construction the heat bath probabilities, so that the acceptance rate becomes one. For any discretization the algorithm is still exact due to the factor $\triangle a_0^{i,j'} / \triangle a_0^{i,j}$ in the acceptance probability (12). The acceptance rate remains close to one when the discretization is reasonably accurate. Therefore, the relative efficiency of a HBA versus our BMA becomes to a large extent a matter of CPU time consumption.

Only step 2 of the BMA procedure requires some thought, all others are straightforward numerical calculations. For $n = 2^{n_2}$ the interval label j of the existing a_0 can be determined in n_2 steps using the binary search recursion

$$j \to j + 2^{i_2}\text{sign}(a_0 - a_0^{i,j}), \qquad i_2 \to i_2 - 1, \tag{13}$$

where the starting values are $i_2 = n_2 - 2$ and $j = 2^{n_2-1}$, and the termination is for $i_2 = 0$ (after which one final logical decision has to be made). As long as a uniform discretization of $s_\sqcup$ is chosen, there is no slowing down of the code with an increase of the size m of the table, while there is a logarithmic slowing down with an increase of the

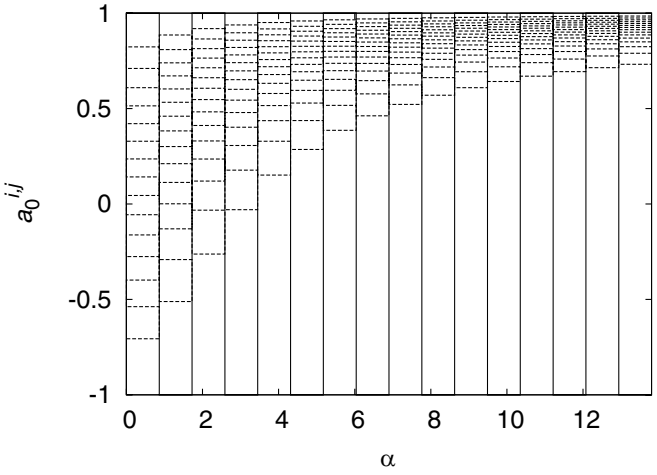

FIG. 2. Partition of the $\triangle a_0^{i,j}$ values for $SU(2)$ at $\beta_g = 2.3$, where the variable $\alpha = \beta_g s_{\sqcup}$ is used on the abscissa. The choices for m and n are the same as in Fig. 1.

$\triangle a_0^{i,j}$ discretization. For the same choice of m and n values as used in Fig. 1, the partition of all $\triangle a_0^{i,j}$ values is shown in Fig. 2. For each bin i on the abscissa the $a_0^{i,j}$ values are calculated for its central value $\alpha^i = \beta_g s_{\sqcup}^i$. For our simulations we used a finer discretization, $m = 32$ and $n = 128$.

Table I illustrates the performance of the $SU(2)$ BMA for a long run on a 4×16^3 lattice at $\beta_g = 2.3$. At this coupling the system exhibits critical slowing down, because of its neighborhood to the deconfining phase transition (see for instance [9] and references therein). Our comparison is with the Fabricius-Haan-Kennedy-Pendleton HBA [7,8], which at this coupling is more efficient than Creutz's HBA [6].

We used 16 384 sweeps for reaching equilibrium and, subsequently, $32 \times 20\,480$ sweeps for measurements. Simulations were performed on 2 GHz Athlon PCs with the $-$O2 option of the (freely available) g77 Fortran compiler. Although our programs are not thoroughly optimized, we report the runtimes in Table I, because we expect their ratios to be relatively stable under further optimization. (For instance, our runs were fully in real*8 precision. By reducing most of the code to real*4 a factor up to two might be gained.)

TABLE I. Efficiency of the $SU(2)$ algorithms on a 4×16^3 lattice at $\beta_g = 2.3$. For the same lattice size integrated autocorrelation times are also given at $\beta_g = 2.2$ and $\beta_g = 2.4$.

	HBA [7,8]	BMA
CPU time	194 873 [s]	199 244 [s]
Acceptance rate	1 (1.043 proposals)	0.975
$\langle \mathrm{Tr}(U_{\square})/2 \rangle$	0.603 147 (17)	0.603 111 (21)
τ_{int}	49.8 (3.5)	48.2 (3.8)
$\tau_{\mathrm{int}}(\beta = 2.2)$	7.1 (0.3)	8.9 (0.4)
$\tau_{\mathrm{int}}(\beta = 2.4)$	6.7 (0.4)	7.0 (1.0)

It is the high acceptance rate of 97.5% which makes the BMA almost as efficient as the HBA. In standard Metropolis procedures one gets high acceptance rates only at the price of small moves, so that acceptance rates between 30% and 50% are optimal [4]. In our BMA the high acceptance rate is achieved by proposing with an approximation of heat bath probabilities for which the acceptance rate is 100%. So, an acceptance rate close to 100% is best for the BMA. The accept/reject step corrects for the failure to approximate the heat bath probability perfectly.

Although the acceptance rate for the HBA is 100%, the $SU(2)$ HBAs use in their inner loops a repeat until accepted (RUA) step. It should be noted that this is distinct from the accept/reject step of the BMA. Like in the original Metropolis method, the latter cannot be iterated until accepted (compare, e.g., p. 137 of Ref. [4]). This would introduce an uncontrolled bias, which for the original Metropolis algorithm is towards too low energies. For the simulation of Table I the RUA step of the HBA [7,8] needs in the average 1.043 iterations to generate the new a_0 value [10]. For small β_g values the number of iterations goes up, so that the Creutz HBA becomes then more efficient than the HBA of Fabricius-Haan-Kennedy-Pendleton, see [8] for a detailed discussion. Independently of β_g the BMA acceptance rate stays always close to 100%.

The difference between a RUA procedure and the accept/reject step of a BMA becomes important for a (checkerboard) parallelization of the updating. While for a BMA the speed is uniform at all nodes, this is not the case for a RUA method, where all nodes have to wait until the last RUA step is completed. For large systems, the consequences would be disastrous, so that at the price of an arguably negligible bias workers tend to impose an upper limit on the number of RUA steps [say three for our $SU(2)$ case]. In the parallelization of BMAs one should be concerned about the speed of the table look-ups. It appears that this can be pipelined similarly as computations with lattice matrix elements, but tests are needed which go beyond the scope of this paper.

The integrated autocorrelation time τ_{int} is a direct measure for the performance of an algorithm. The number of sweeps needed to achieve a desired accuracy is directly proportional to τ_{int}. Table I gives τ_{int} for the Wilson plaquette together with the expectation value of this operator. Error bars are given in parenthesis and apply to the last digits. They are calculated with respect to 32 bins (jackknife bins in case of τ_{int}), relying on the data analysis software of [4]. We see that the expectation values are well compatible with one another ($Q = 0.18$ in a Gaussian difference test). For τ_{int} we know that the HBA should give a slightly lower value than the BMA. That the τ_{int} data at $\beta_g = 2.3$ table come out in the opposite order is attributed to a statistical fluctuation. This is confirmed by

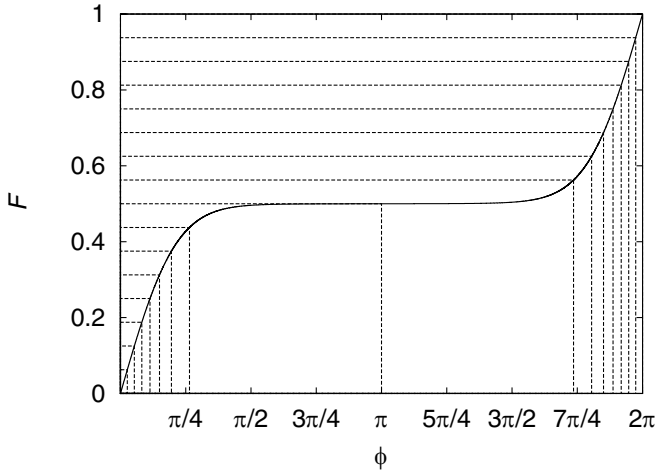

FIG. 3. Discretization of the cumulative distribution function $F(\phi; r_{\sqcup}^{11})$ for $U(1)$ at $\beta_g = 1.0$ for the choices $m = 16$ (equidistant $r_{\sqcup}^{i}$ values) and $n = 2^4 = 16$.

shorter runs which we performed at other β_g values, whose $\tau_{\rm int}$ results are also listed in the table.

B. $U(1)$

Next we consider the $U(1)$ gauge group. The ''matrices'' are then complex numbers on the unit circle, $U_{ij} = \exp(i\phi_{ij})$, and the analogue of Eq. (5) becomes

$$r_{\sqcup} e^{i\phi_{\sqcup}} = \sum_{k=1}^{6} e^{i\phi_{\sqcup,k}}, \tag{14}$$

$r_{\sqcup} = \sqrt{(\sum_{k=1}^{6} \cos\phi_{\sqcup,k})^2 + (\sum_{k=1}^{6} \sin\phi_{\sqcup,k})^2}$. We are led to the cumulative distribution function

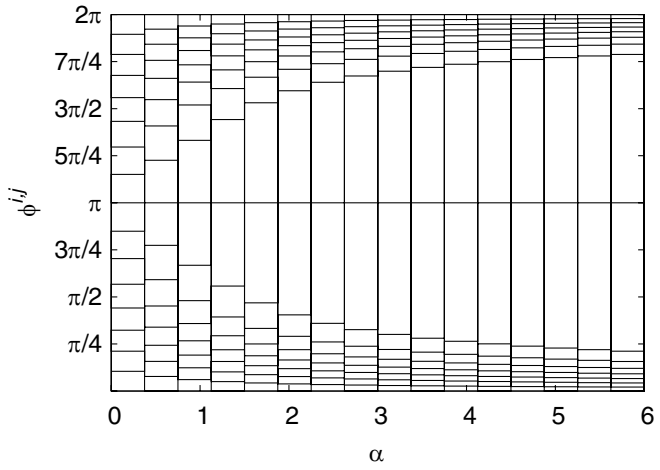

FIG. 4. Partition of the $\triangle\phi^{i,j}$ values for $U(1)$ at $\beta_g = 1.0$, where the variable $\alpha = \beta_g r_{\sqcup}$ is used on the abscissa. The choices for m and n are the same a in Fig. 3.

TABLE II. Efficiency of the $U(1)$ algorithms on a 4×16^3 lattice at $\beta_g = 1.0$.

	Metropolis	BMA
CPU time	84 951 [s]	107 985 [s]
Acceptance rate	0.286	0.972
$\langle\cos\phi_{\square}\rangle$	0.591 03 (16)	0.591 06 (12)
$\tau_{\rm int}$	341 (26)	142 (10)

$$F_1(\phi) = N_1 \int_0^{\phi} d\phi' e^{\beta_g r_{\sqcup} \cos(\phi')}, \tag{15}$$

where the normalization is $F_1(2\pi) = 1$ and the angle $(\phi + \phi_{\sqcup})\mathrm{mod}(2\pi)$ will be stored.

We test the performance of the $U(1)$ BMA for a 4×16^3 lattice at $\beta_g = 1.0$, again a coupling which puts the system close to the deconfining phase transition, which is weakly first order for $U(1)$ (see for instance [11] and references therein). HBAs have been designed in Ref. [12,13]. Both HBAs rely on a RUA step, so that the remarks made in this connection for $SU(2)$ apply also to $U(1)$. We have only tested the HBA of Ref. [13], which turns out to be about 20% slower than our BMA, while the integrated autocorrelation time is about 10% lower. Overall an advantage of 10% in favor of the $U(1)$ BMA, which reiterates that HBAs and BMAs have about equal efficiency, when efficient HBAs exist.

We compare the $U(1)$ BMA now with a conventional Metropolis algorithm, which proposes new angles uniformly in the (entire) range $[0, 2\pi)$. For the BMA we follow the same lines as previously for $F(a_0)$ of Eq. (8). Figure 3 plots $F_1(\phi)$ at $\beta_g = 1.0$ using a representative $r_{\sqcup}^{i}$ value and Fig. 4 shows the entire tabulation $\triangle\phi^{i,j}$. Table II summarizes the results. At $\beta_g = 1$ the acceptance rate of the standard Metropolis procedure is still about 30%, so that a restriction of the proposal range to increase the acceptance rate is not warranted [4]. From the data of the table we conclude that the BMA improves the Metropolis performance at $\beta_g = 1$ by a factor of about two.

When comparing with a full-range Metropolis algorithm an upper bound on the improvement factor is given by the ratio of the acceptance rates, in the present case $0.972/0.282 = 3.45$. This applies also to comparisons of such Metropolis algorithms with HBAs, substituting then one for the acceptance rate. The bound will normally not be saturated, because rms deviations of the new variables from the old variables are smaller for a BMA or HBA than for a full-range Metropolis algorithm. Larger gains can be achieved when the Metropolis acceptance rates are small. For $U(1)$ this happens for $\beta_g \gg 1$.

III. SUMMARY AND CONCLUSIONS

In summary, BMAs are an alternative to HBAs. BMAs work still in situations for which HBAs fail, because there

is no efficient inversion of the cumulative distribution function in question. In lattice gauge theory this is the case for some Higgs system and for actions which are nonlinear in the Wilson plaquette operator (see, e.g., Ref. [14] and references therein). Obviously, similar situations ought to exist for energy functions in many other fields. We leave it to the reader to identify whether her or his simulations would benefit from using a BMA. Finally, let us mention that BMAs may be combined with overrelaxation moves [15–17] in the same way as one does for HBAs or standard Metropolis algorithms (the subject of overrelaxation algorithms for lattice gauge theories deserves further study [18]).

ACKNOWLEDGMENTS

This work was in part supported by the U.S. Department of Energy under Contract No. DE-FG02-97ER41022. We would like to thank Urs Heller for useful correspondence.

[1] W. K. Hastings, Biom. J. **57**, 97 (1970).
[2] A. D. Bruce, J. Phys. A **18**, L873 (1985); A. Milchev, D. W. Heermann, and K. Binder, J. Stat. Phys. **44**, 749 (1986).
[3] M. W. Deem and J. S. Bader, Mol. Phys. **87**, 1245 (1996); G. Favrin, A. Irbäck, and F. Sjunnesson, J. Chem. Phys. **114**, 8154 (2001); J. P. Ulmschneider and W. L. Jorgensen, J. Chem. Phys. **118**, 4261 (2003).
[4] B. A. Berg, *Markov Chain Monte Carlo Simulations and Their Statistical Analysis* (World Scientific, Singapore, 2004).
[5] B. A. Berg, Phys. Rev. Lett. **90**, 180601 (2003); B. A. Berg and H.-X. Zhou, cond-mat/0502113.
[6] M. Creutz, Phys. Rev. D **21**, 2308 (1980).
[7] K. Fabricius and O. Haan, Phys. Lett. B **143**, 459 (1984).
[8] A. D. Kennedy and B. J. Pendleton, Phys. Lett. B **156**, 393 (1985).
[9] J. Fingberg, U. Heller, and F. Karsch, Nucl. Phys. **B392**, 493 (1993).
[10] In some of the literature the quantity 1/(average number of RUA heat bath iterations per update) is also called acceptance rate. It should not be confused with the acceptance rate defined by our Eq. (12).
[11] G. Arnold, K. Schilling, and Th. Lippert, Phys. Rev. D **59**, 054509 (1999).
[12] R. J. Wensley, Ph.D. thesis, University of Illinois [Institution Report No. ILL-TH-89-25, 1989 (unpublished)].
[13] T. Hattori and H. Nakajima, Nucl. Phys. B Proc. Suppl. **26**, 635 (1992).
[14] P. Damgaard and U. Heller, Nucl. Phys. **B304**, 63 (1988); P. W. Stephenson, hep-lat/9604008.
[15] S. L. Adler, Phys. Rev. D **23**, 2901 (1981).
[16] F. R. Brown and T. J. Woch, Phys. Rev. Lett. **58**, 2394 (1987).
[17] M. Creutz, Phys. Rev. D **36**, 515 (1987).
[18] I. Horváth and A. D. Kennedy, Nucl. Phys. **B510**, 367 (1998) and references therein.