HEAT BATH EFFICIENCY WITH METROPOLIS-TYPE UPDATING

Alexei Bazavov\textsuperscript{a,b} and Bernd A. Berg\textsuperscript{a,b}
\textsuperscript{a)} Department of Physics, Florida State University, Tallahassee, FL 32306-4350
\textsuperscript{b)} School of Computational Science, Florida State University, Tallahassee, FL 32306-4120
(April 28, 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.

PACS: 05.10.Ln, 11.15.Ha

I. INTRODUCTION

The possibility of constructing biased Metropolis algorithms (BMAs) is known since quite a while [1]. Although they have occasionally been used in the statistical physics [2] and bio-chemical [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 distribution, required for a HBA, is numerically so slow that it is not a suitable option. While working at a particular link $(ij)$, we need only to consider the contribution to $S$, which comes from the staples containing this link. If we denote by $U_{\perp,k}$, 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 $U_{ij}$ is a $U(1)$ or a $SU(2)$ matrix ($N_c = 1$ or $2$) associated with the link $(ij)$. The reversed link is associated with the inverse matrix. The aim is to calculate expectation values with respect to the Euclidean path integral

\begin{equation}
Z = \int \prod_{(ij)} dU_{ij} e^{+\beta g S(U)}
\end{equation}

where the integrations are over the invariant group measure. Using the invariance of the group measure, one finds

\begin{equation}
dP(U) \sim dU \exp \left[\frac{\beta g}{N_c} \text{Re Tr} \left( U \sum_{k=1}^{6} U_{\perp,k} \right) \right].
\end{equation}

A. $SU(2)$

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

\begin{equation}
U = a_0 I + i \vec{a} \cdot \vec{\sigma}, \quad a_0^2 + \vec{a}^2 = 1,
\end{equation}

where $I$ denotes the $2 \times 2$ 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 $U_{\perp}$ which corresponds to the sum of the staples in equation (3) by

\begin{equation}
s_{\perp} U_{\perp} = \sum_{k=1}^{6} U_{\perp,k}, \quad s_{\perp} = \det \left( \sum_{k=1}^{6} U_{\perp,k} \right).
\end{equation}

Using the invariance of the group measure, one finds

\begin{equation}
dP \left( U U_0^{-1} \right) \sim d\Omega \, da_0 \sqrt{1 - a_0^2} \exp (\beta g s_{\perp} a_0)
\end{equation}
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

$$P(a_0) \sim \sqrt{1 - a_0^2} \exp(\beta_g s_{\|} a_0)$$  \hspace{1cm} (7)

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

$$F(a_0) = N_0 \int_{-1}^{a_0} da'_0 \sqrt{1 - a'_0^2} \exp(\beta_g s_{\|} a'_0)$$  \hspace{1cm} (8)

where $N_0$ ensures the normalization $F(1) = 1$. The HBA updates $a_0$ by converting a uniformly distributed random number $0 \leq x < 1$ 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 $s_{\|}$, $0 \leq s_{\|} \leq 6$, into $m$ discrete values $s_{\|}^i$, $i = 1, \ldots, m$ so that

$$0 < s_{\|}^1 < s_{\|}^2 \ldots < s_{\|}^m$$  \hspace{1cm} (9)

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

$$\frac{j}{n} = F(a_0^{i,j} : s_{\|}^i)$$  \hspace{1cm} (10)

and we also tabulate the differences

$$\Delta a_0^{i,j} = a_0^{i,j} - a_0^{i,j-1} \text{ for } j = 1, \ldots, n$$  \hspace{1cm} (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 $s_{\|}^i$ value.

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

1. Find the $s_{\|}^i$ value (only $i$ is needed) which is nearest to the actual $s_{\|}$ value.

2. Place the present $a_0$ value on the discretization grid, i.e., find the integer $j$ through the relation $a_0^{i,j-1} \leq a_0 < a_0^{i,j}$.

3. Pick an integer value $j'$ uniformly distributed in the range $1$ to $n$.

4. Propose $a_0' = a_0^{i,j-1} + x' \Delta a_0^{i,j}$, where $x'$, $0 \leq x' < 1$, is a uniformly distributed random number.

5. Accept $a_0'$ with the probability

$$p_a = \frac{\exp(\beta_g s_{\|} a'_0) \Delta a_0^{i,j}}{\exp(\beta_g s_{\|} a_0) \Delta a_0^{i,j}}$$  \hspace{1cm} (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 a reasonably accurate discretization the algorithm is still exact due to the factor $\Delta a_0^{i,j} / \Delta a_0^{i,j}$ in the acceptance probability (12), and the acceptance rate remains close to one. 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 $n = 2^n$, the interval label $j$ of the existing $a_0$ can be determined in $n_2$ steps using the binary search recursion

$$j \to j + 2^i \text{ sign}(a_0 - a_0^{i,j}), \quad i_2 \to i_2 - 1$$  \hspace{1cm} (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 $s_{\|}$ 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 $\Delta a_0^{i,j}$ discretization. For the same choice of $m$ and $n$
values as used in Fig. 1, the partition of all $\Delta a^{\alpha, j}_{\beta}$ values is shown in Fig. 2. For each bin $i$ on the abscissa the $a^{\alpha, j}_{\beta}$ values are calculated for its central value $\alpha^i = \beta g s^i_j$. 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 $4 \times 16^3$ 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 [7,8], which at this coupling is more efficient than the HBA of Fabricius-Haan-Kennedy-Pendleton, see [8] for a detailed discussion. Independently of $\beta_g$ 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).

The integrated autocorrelation time $\tau_{int}$ is a direct measure for the performance of an algorithm. The number of sweeps needed to achieve a desired accuracy is directly proportional to $\tau_{int}$. Table I gives $\tau_{int}$ 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 $\tau_{int}$), 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 $\tau_{int}$ we know that the HBA should give a slightly lower value than the BMA. That the $\tau_{int}$ data at $\beta_g = 2.3$ table come out in the opposite order is attributed to a statistical fluctuation. This is confirmed by shorter runs which we performed at other $\beta_g$ values, whose $\tau_{int}$ results are also listed in the table.

![FIG. 2. Partition of the $\Delta a^{\alpha, j}_{\beta}$ values for $SU(2)$ at $\beta_g = 2.3$, where the variable $\alpha = \beta g s^i_j$ is used on the abscissa. The choices for $m$ and $n$ are the same as in Fig. 1.](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$HBA$ [7,8]</th>
<th>$BMA$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>194,873 [s]</td>
<td>199,244 [s]</td>
</tr>
<tr>
<td>Acceptance rate</td>
<td>1 (1.043 proposals)</td>
<td>0.975</td>
</tr>
<tr>
<td>$\langle \text{Tr}(U^\dagger)/2 \rangle$</td>
<td>0.603147 (17)</td>
<td>0.603111 (21)</td>
</tr>
<tr>
<td>$\tau_{int}$</td>
<td>49.8 (3.5)</td>
<td>48.2 (3.8)</td>
</tr>
<tr>
<td>$\tau_{int}(\beta = 2.2)$</td>
<td>7.1 (0.3)</td>
<td>8.9 (0.4)</td>
</tr>
<tr>
<td>$\tau_{int}(\beta = 2.4)$</td>
<td>6.7 (0.4)</td>
<td>7.0 (1.0)</td>
</tr>
</tbody>
</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

$$r_{ij} e^{i \phi_{ij}} = \sum_{k=1}^{6} e^{i \phi_{i,j,k}},$$

(14)

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

$$F_1(\phi) = N_1 \int_0^{\phi} d\phi' e^{\beta_g r_{ij} \cos(\phi')}$$

(15)

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

We test the performance of the $U(1)$ BMA for a $4 \times 16^3$ 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 re-iterates 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). Fig. 3 plots $F_1(\phi)$ at $\beta_g = 1.0$ using a representative $r_{ij}$ value and Fig. 4 shows the entire tabulation $\Delta \phi^{i,j}$. 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

![Fig. 3](image-url) FIG. 3. Discretization of the cumulative distribution function $F(\phi; r_{ij}^1)$ for $U(1)$ at $\beta_g = 1.0$ for the choices $m = 16$ (equidistant $r_{ij}$ values) and $n = 2^4 = 16$.

![Fig. 4](image-url) FIG. 4. Partition of the $\Delta \phi^{i,j}$ values for $U(1)$ at $\beta_g = 1.0$, where the variable $\alpha = \beta_g r_{ij}$ is used on the abscissa. The choices for $m$ and $n$ are the same as in Fig. 3.

<table>
<thead>
<tr>
<th>value</th>
<th>Metropolis</th>
<th>BMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>84.951 [s]</td>
<td>107.985 [s]</td>
</tr>
<tr>
<td>Acceptance rate</td>
<td>0.286</td>
<td>0.972</td>
</tr>
<tr>
<td>$\langle \cos \phi \rangle$</td>
<td>0.59103 (16)</td>
<td>0.59106 (12)</td>
</tr>
<tr>
<td>$\tau_{int}$</td>
<td>341 (26)</td>
<td>142 (10)</td>
</tr>
</tbody>
</table>
function in question. In lattice gauge theory this is the case for some Higgs system and for actions which are non-linear 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.

**ACKNOWLEDGMENTS**

This work was in part supported by the US Department of Energy under contract DE-FG02-97ER41022. We like to thank Urs Heller for useful correspondence.

[10] In some of the literature the quantity 1/(average number of RUA heat bath iterations per update) is is also called acceptance rate. It should not be confused with the acceptance rate defined by our Eq. (12).