Multiple "time step" Monte Carlo

Balázs Hetényi[a)]
Department of Chemistry, Princeton University, Princeton, NJ 08544 and Department of Chemistry and Center for Biomolecular Simulation, Columbia University, New York, New York 10027

Katarzyna Bernacki, and B. J. Berne
Department of Chemistry and Center for Biomolecular Simulation, Columbia University, New York, New York 10027

(Received 20 June 2002; accepted 15 August 2002)

We propose a sampling scheme to reduce the CPU time for Monte Carlo simulations of atomic systems. Our method is based on the separation of the potential energy into parts that are expected to vary at different rates as a function of coordinates. We perform n moves that are accepted or rejected according to the rapidly varying part of the potential, and the resulting configuration is accepted or rejected according to the slowly varying part. We test our method on a Lennard-Jones system. We show that use of our method leads to significant savings in CPU time. We also show that for moderate system sizes the scaling of CPU time with system size can be improved (for $n=40$ the scaling is predominantly linear up to 1000 particles). © *2002 American Institute of Physics.* [DOI: 10.1063/1.1512645]

I. INTRODUCTION

The most often used tools in obtaining expectation values for classical many-body systems at equilibrium are the Monte Carlo[1–3] (MC) and molecular dynamics (MD) methods.[2,3] MC and MD are methods which generate equilibrium configurations for various ensembles. In MD the particles of the system are propagated in real time according to Newton's equations of motion, whereas in MC configurations are sampled by a Markov process from the Boltzmann distribution (the propagation takes place in "Markovian time").

In the case of MD, significant savings in CPU time can be achieved by using multiple time-step methods. When time-reversible MD integrators are used, propagators for multiple time-step algorithms can be derived from the Liouville formulation of classical mechanics.[4,5] If the Hamiltonian is separated into two parts, a rapidly and a slowly varying part, than an MD propagator can be derived in which the slowly varying part of the Hamiltonian is evaluated less frequently than the rapidly varying part, hence savings in CPU time can be achieved. Common criteria of separation into rapidly and slowly varying parts[6] are differences in the masses of the particles of the system,[4] temperature differences among different subunits of a system,[7] or differences in the spatial range of potentials[4] (short range varying rapidly, long range varying slowly).

In this paper, we propose a sampling algorithm that is similar in spirit to multiple time-scale molecular dynamics, and is generally applicable to any type of system that can be treated by MD or MC. The only assumption of our method is the separability of the potential of the system into a part that varies rapidly and a part that varies slowly as a function of coordinates. In our method, which we call the multiple "time-step" Monte Carlo (MTS–MC), the potential is separated into a long-range and a short-range part, and the long-range part is evaluated less frequently than the short-range part. A particle is moved and accepted or rejected n times (we call n the splitting parameter) according to the short-range potential. The configuration resulting from the n short-range moves is accepted or rejected using a Metropolis criterion based on the long-range potentials before and after the n short-range moves.

Our procedure is similar in spirit but different in significant ways from the adiabatic nuclear and electronic sampling Monte Carlo (ANES–MC) method developed by Siepmann and co-workers[8,9] to sample systems in which the configuration space can be subdivided into "fast" and "slow" degrees of freedom, such as electronic and nuclear degrees of freedom respectively. ANES–MC was particularly designed to treat polarizable systems and was applied to the simple point charge fluctuating charge (SPC–FQ) model,[10] where the fluctuating charges were thermostated at low temperature and the nuclear motions were thermostated at the temperature of interest. In ANES–MC, the "slow" coordinates are updated less frequently than "fast" coordinates. In MTS–MC the configuration space need not be subdivided into fast and slow coordinates, but instead the potential is subdivided into a short- and long-range part.

In this study, we implement the MTS–MC scheme for a system of Lennard-Jones particles. In a sequel paper,[11] we apply the new algorithm to a real long-range potential, the SPC water model combined with the Ewald sum. Implementation in the case of the Lennard-Jones system requires use of a split neighbor list that formed the basis of earlier multiple time-step methods.[12,13] We demonstrate that for systems described by pair potentials, the CPU time can be made predominantly linear for the system sizes that we have studied (maximum 1000), an improvement over the quadratic scaling of regular MC. In order to illustrate this point, we use a Lennard-Jones potential in which the cutoff is set to one-half of the edge of the simulation box.

[a)]Present address: SISSA, International School for Advanced Study, via Beirut 2-4, 34014, Trieste, Italy.

In Sec. II, we introduce the new Monte Carlo method, multiple ''time step'' Monte Carlo (MTS–MC), and show that the new algorithm obeys the detailed balance condition. In Sec. III, we present computational and simulation details for the Lennard-Jones system. In Sec. IV, we discuss results for the Lennard-Jones simulations. Results are summarized in Sec. V.

II. METHOD

Our sampling procedure is as follows: Given a system of particles interacting via a potential energy function $V(\mathbf{x})$, where $\mathbf{x}$ denotes the coordinates in configuration space. The partition function of the system is given by

$$Q=\int d\mathbf{x}\exp[-\beta V(\mathbf{x})], \tag{1}$$

where β denotes the inverse temperature. Suppose that $V(\mathbf{x})$ can be written as the sum of two additive parts as

$$V(\mathbf{x})=V_{\mathrm{I}}(\mathbf{x})+V_{\mathrm{II}}(\mathbf{x}). \tag{2}$$

To evaluate the partition function for such a system, we propose the following sampling procedure:

Step 1: For a given configuration $\mathbf{x}$, we evaluate $V_{\mathrm{I}}(\mathbf{x})$ and $V_{\mathrm{II}}(\mathbf{x})$. We store the configuration $\mathbf{x}$ as $\mathbf{x}_{\mathrm{old}}$ and $\mathbf{y}_{\mathrm{old}}$.

Step 2: Using $\mathbf{y}_{\mathrm{old}}$ as a starting configuration we generate a new configuration by making n uniform random moves and accepting or rejecting each move by an acceptance probability constructed using only V_{I},

$$\mathrm{acc}_{\mathrm{I}}(\mathbf{y}_{\mathrm{old}}\rightarrow\mathbf{y}_{\mathrm{new}})=\min[1,\exp\{-\beta(V_{\mathrm{I}}(\mathbf{y}_{\mathrm{new}})-V_{\mathrm{I}}(\mathbf{y}_{\mathrm{old}}))\}]. \tag{3}$$

The configuration resulting after the nth move is stored as $\mathbf{x}_{\mathrm{new}}$.

Step 3: We accept or reject the resulting configuration by an acceptance probability constructed from V_{II} as

$$\mathrm{acc}_{\mathrm{II}}(\mathbf{x}_{\mathrm{old}}\rightarrow\mathbf{x}_{\mathrm{new}})=\min[1,\exp\{-\beta(V_{\mathrm{II}}(\mathbf{x}_{\mathrm{new}})-V_{\mathrm{II}}(\mathbf{x}_{\mathrm{old}}))\}]. \tag{4}$$

Note that configurations that have been accepted or rejected using Eq. (4) obey detailed balance, hence if intermediate configurations are used in calculating observables an error is introduced.

A. Detailed balance

In this section, we show that configurations generated according to Eq. (4) are distributed according to the desired (in this case Boltzmann) distribution. We do this by demonstrating that our procedure obeys the detailed balance condition. Our goal is to prove that the configurations generated by the above procedure obey

$$P(\mathbf{x})T(\mathbf{x}\rightarrow\mathbf{x}')=P(\mathbf{x}')T(\mathbf{x}'\rightarrow\mathbf{x}), \tag{5}$$

where $P(\mathbf{x})$ is the unnormalized probability distribution

$$P(\mathbf{x})=\exp[-\beta V(\mathbf{x})], \tag{6}$$

and $T(\mathbf{x}\rightarrow\mathbf{x}')$ is the transition probability of arriving in state $\mathbf{x}'$ starting from state $\mathbf{x}$ after performing all three steps of the sampling procedure.

Using Eq. (2) we write the probability distribution as

$$P(\mathbf{x})=P_{\mathrm{I}}(\mathbf{x})P_{\mathrm{II}}(\mathbf{x}), \tag{7}$$

where

$$\begin{aligned}P_{\mathrm{I}}(\mathbf{x})&=\exp[-\beta V_{\mathrm{I}}(\mathbf{x})],\\ P_{\mathrm{II}}(\mathbf{x})&=\exp[-\beta V_{\mathrm{II}}(\mathbf{x})].\end{aligned} \tag{8}$$

We define $T_{\mathrm{I}}^{(n)}(\mathbf{x}\rightarrow\mathbf{x}')$ as the probability of arriving in state $\mathbf{x}'$ from state $\mathbf{x}$, after executing the n moves of step 2 of the sampling procedure [i.e., making n uniform random moves and accepting or rejecting according to Eq. (3)]. Note that according to our procedure

$$T(\mathbf{x}\rightarrow\mathbf{x}')=T_{\mathrm{I}}^{(n)}(\mathbf{x}\rightarrow\mathbf{x}')\mathrm{acc}_{\mathrm{II}}(\mathbf{x}\rightarrow\mathbf{x}'). \tag{9}$$

The transition probability corresponding to one of the n moves of step 2 can be written as

$$T_{\mathrm{I}}(\mathbf{x}\rightarrow\mathbf{x}')=\alpha(\mathbf{x}\rightarrow\mathbf{x}')\mathrm{acc}_{\mathrm{I}}(\mathbf{x}\rightarrow\mathbf{x}'), \tag{10}$$

where $\alpha(\mathbf{x}\rightarrow\mathbf{x}')$ denotes the probability of arriving in state $\mathbf{x}'$ starting in state $\mathbf{x}$ by making one uniform random move, and the form of the acceptance probability $\mathrm{acc}_{\mathrm{I}}$ is given in Eq. (3). By construction, $T_{\mathrm{I}}(\mathbf{x}\rightarrow\mathbf{x}')$ satisfies

$$P_{\mathrm{I}}(\mathbf{x})T_{\mathrm{I}}(\mathbf{x}\rightarrow\mathbf{x}')=P_{\mathrm{I}}(\mathbf{x}')T_{\mathrm{I}}(\mathbf{x}'\rightarrow\mathbf{x}). \tag{11}$$

Using $T_{\mathrm{I}}(\mathbf{x}\rightarrow\mathbf{x}')$, one can write $T_{\mathrm{I}}^{(n)}(\mathbf{x}\rightarrow\mathbf{x}')$ as

$$T_{\mathrm{I}}^{(n)}(\mathbf{x}\rightarrow\mathbf{x}')=\int d\mathbf{x}_1\cdots d\mathbf{x}_{n-1}\prod_{i=0}^{n-1}T_{\mathrm{I}}(\mathbf{x}_i\rightarrow\mathbf{x}_{i+1}), \tag{12}$$

where $\mathbf{x}_0=\mathbf{x}$ and $\mathbf{x}_n=\mathbf{x}'$. It follows from Eqs. (11) and (12) that

$$P_{\mathrm{I}}(\mathbf{x})T_{\mathrm{I}}^{(n)}(\mathbf{x}\rightarrow\mathbf{x}')=P_{\mathrm{I}}(\mathbf{x}')T_{\mathrm{I}}^{(n)}(\mathbf{x}'\rightarrow\mathbf{x}). \tag{13}$$

In order to construct the acceptance probability $\mathrm{acc}_{\mathrm{II}}$ such that Eq. (5) is obeyed, we define the quantity

$$q(\mathbf{x}\rightarrow\mathbf{x}')=\frac{P(\mathbf{x})T_{\mathrm{I}}^{(n)}(\mathbf{x}\rightarrow\mathbf{x}')}{P(\mathbf{x}')T_{\mathrm{I}}^{(n)}(\mathbf{x}'\rightarrow\mathbf{x})}. \tag{14}$$

Using Eq. (13), we obtain for $q(\mathbf{x}\rightarrow\mathbf{x}')$,

$$q(\mathbf{x}\rightarrow\mathbf{x}')=\frac{P_{\mathrm{II}}(\mathbf{x})}{P_{\mathrm{II}}(\mathbf{x}')}. \tag{15}$$

The acceptance probability can thus be written

$$\mathrm{acc}_{\mathrm{II}}(\mathbf{x}\rightarrow\mathbf{x}')=\min[1,q(\mathbf{x}\rightarrow\mathbf{x}')] \tag{16}$$

which is identical to Eq. (4), therefore our procedure obeys detailed balance. That completes the proof of detailed balance.

III. LENNARD-JONES SYSTEM

The potential energy for a system of particles interacting via the Lennard-Jones potential is given by

$$V(\mathbf{x})=\sum_{i<j}V_{\mathrm{LJ}}(r_{ij}), \tag{17}$$

where r_{ij} is the distance between two particles of the system and

$$V_{\mathrm{LJ}}(r)=4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12}-\left(\frac{\sigma}{r}\right)^{6}\right). \tag{18}$$

We separate the potential into long- and short-range contributions[14]

$$V_{\mathrm{LJ}}(r)=V_s(r)+V_l(r), \tag{19}$$

where

$$V_s(r)=S(r)V(r), \tag{20}$$

$$V_l(r)=(1-S(r))V(r), \tag{21}$$

and $S(r)$ is the step function

$$S(r)=\begin{cases}1 & \text{for } r<r_s,\\ 0 & \text{for } r\geqslant r_s.\end{cases} \tag{22}$$

Note that short- (long-) range part of the potential corresponds to $V_{\mathrm{I}}(V_{\mathrm{II}})$.

We implement our sampling procedure according to the split potential in Eq. (19). Thus, the ith particle is moved n times and accepted or rejected according to the sum over the short-range potential

$$V_{\mathrm{I}}^{(i)}=\sum_{j\neq i, r_{ij}\leqslant r_s} V_s(r_{ij}). \tag{23}$$

Note that in Eq. (23) all contributions come from particles that are closer to particle i than r_s. The final configuration is accepted or rejected using the potential

$$V_{\mathrm{II}}^{(i)}=\sum_{j\neq i, r_s<r_{ij}<r_c} V_l(r_{ij}), \tag{24}$$

where r_c denotes the potential cutoff.

The fact that in Eq. (23) only particles that are within r_s contribute to the sum enables use of a neighbor list, which makes the implementation efficient. Every m_s'' long-range steps, we make an array listing the indices of all atoms that are within a certain distance r_s'' of each atom. Every m_s' short-range steps we use the long-range neighbor list to make a list of atoms within r_s' of each atom. Note that $r_s<r_s'<r_s''$. Since in this study we want to investigate the dependence of CPU time on system size, our potential cutoff r_c is set to one-half of the box size for all simulations. Note also that the two neighbor lists are used for the short-range steps only. For the long-range step, we evaluate the potential for all particles that are within one half of the box length from the central particle. Of course, for the Lennard-Jones 6–12 potential, it is not necessary to adopt minimum image cutoffs, but this is done here to illustrate the power that might be expected of the MTS–MC method when it is applied to longer range potentials.

We ran simulations of various system sizes in a cubic box using periodic boundary conditions. We started our simulations from a simple cubic configuration. We computed all quantities using the standard Monte Carlo method as well as MTS–MC. In both the regular MC and the MTS–MC we used a cutoff distance of $r_c=L/2$ where L is the length of the edge of the simulation box. In the MTS–MC we used an inner cutoff r_s of 1.6σ, and an inner neighbor list cutoff r_s' of 2.0σ and an outer one of $r_s''=3.5\sigma$ for runs with $n=10$, 20 and $r_s''=3.7\sigma$ for ones with $n=40$. We generated equilibrated configurations by runs of 50 000 steps. Our average quantities are based on calculations on the order of 10^5 MC steps in the case of regular MC, and the same number of short range steps for MTS–MC. For standard MC we optimized the diffusion as a function of step size (0.7σ), for MTS–MC we used a lower step size (0.5σ) due to the inner neighbor list.

TABLE I. Average of the potential energy and average of the potential energy squared in reduced units for three system sizes from regular MC and MTS–MC ($n=10$) simulations.

No. of particles	Method	$\langle V\rangle$	$\langle V^2\rangle$
343	MC	-0.8426 ± 6	0.712 ± 2
	MTS–MC	-0.8429 ± 9	0.712 ± 4
512	MC	-0.8527 ± 5	0.7282 ± 8
	MTS–MC	-0.8534 ± 5	0.7294 ± 8
1000	MC	-0.8629 ± 4	0.7451 ± 8
	MTS–MC	-0.8631 ± 4	0.7455 ± 7

IV. RESULTS

In Table I we compare the average potential energies and average potential energies squared for MC and MTS–MC for three system sizes. The quantities shown were averaged using the same number of data points; a calculation of each quantity was made every 20 steps in the case of MC, and every two long-range steps (or 20 short-range steps) in the case of MTS–MC. The regular MC run was 100 000 steps long. MTS–MC was run with a splitting parameter of $n=10$. The results for the observables shown are almost indistinguishable, indicating that the implementation does not affect the accuracy of the method. The error bars for the two methods are also comparable. The radial distribution functions (not shown) are also indistinguishable for the two cases.

In Table II we present the speedup as a function of system size. In order to compare CPU times of the two methods, we ran both methods for four different system sizes. For a given system size, the comparison between MC and MTS–MC were made between runs in which the number of MC steps equaled the number of short-range MTS–MC steps. In the second column of Table II, we show the CPU time for MC divided by the CPU time for MTS–MC. The ratios of the raw CPU times indicate that MTS–MC is significantly faster than standard MC.

While both methods sample the same distribution, the trajectories are not equivalent trajectories in the sense that correlation functions calculated in MC time are not necessarily equal. We found diffusion to be slower with increasing splitting parameter n in MTS–MC simulations. We attribute this behavior to the fact that if a long-range move is rejected, the system effectively stays in the same place for n equiva-

TABLE II. Speedup as a function of system size (for the meaning of raw and diffusion adjusted speedup, see text).

No. of particles	Raw CPU speedup	Diffusion adjusted speedup
343	4.1	3.0
512	5.0	3.7
729	5.6	4.2
1000	6.3	4.7

TABLE III. Diffusion adjusted speedup (DAS) for the MTS–MC as a function of the splitting parameter n.

No. of particles	Diffusion adjusted speedup (DAS)		
	MTS–MC ($n=10$)	MTS–MC ($n=20$)	MTS–MC ($n=40$)
343	3.0	3.0	2.6
512	3.7	4.0	3.5
729	4.2	4.8	4.5
1000	4.7	5.6	5.5

lent regular MC moves. Furthermore the sharp switch function used in separating the potential into long- and short-range part may also hinder diffusion.

Rao, Pangali, and Berne[15,16] proposed using the diffusion of a system as a good measure of MC simulation efficiency. Also, the correlation time of energy can be used as an efficiency parameter. They showed that these quantities provide an intuitive measure for optimization as well as efficiency assessment of different MC algorithms. In this paper, we use diffusion coefficients as the score variables to compare the MTS–MC scheme with the standard MC method. In Table II we show the diffusion adjusted speedup (DAS) for four system sizes. We define DAS as the CPU speedup multiplied by the ratio $D_{\mathrm{MTS}}/D_{\mathrm{MC}}$, where $D_{\mathrm{MTS}}(D_{\mathrm{MC}})$ is the diffusion constant obtained using MTS–MC(MC). Since the diffusion constant is size independent, the DAS is different from the raw ratio of CPU times by a constant factor (ratio $=0.74$ for $n=10$). The overall speedup is still considerable (4.7 for a system of 1000 particles).

In Table III we present comparisons of DAS for different values of the splitting parameter n. While the diffusion slows down with increasing n, the scaling with system size improves. Since the diffusion constant is independent of N, using high values of n becomes more favorable at large system sizes.

In order to further quantify the advantage of using MTS–MC over standard MC, we compare the system size scaling of the two methods. A standard MC algorithm for a system interacting through pair potentials is expected to scale as the number of potential evaluations, i.e., as the number of pairs of particles. We can therefore write the CPU time as the sum of a linear and a quadratic term ($aN+bN^2$) where N is the number of particles in the system. When a cutoff is used the linear term is expected to dominate at typical system sizes (since the number of potential evaluations reduces to $N\,N_{ngh}$ where N_{ngh} is the number of neighbors, a size-independent quantity). Since the application of MTS–MC enables use of a neighbor list even in cases where a potential cutoff cannot be used, it is expected that the range of system sizes where the linear term dominates the scaling behavior should increase compared to regular MC. In Table IV the result of fitting the raw CPU time versus system size to the function $aN+bN^2$ is presented and in Fig. 1 we show this CPU time as a function of system size, including the results of our fits. In the case of $n=40$ the linear coefficient is three orders of magnitude larger than the quadratic one, and the linear term is the dominant term in the scaling of CPU time with system size for all system sizes that we have studied.

TABLE IV. Scaling parameters resulting from fitting the raw CPU time vs system size to the function $aN+bN^2$.

Method	Splitting parameter n	Scaling parameters	
		a	b
MC		0.3	0.0400
MTS–MC	10	2.1	0.0051
MTS–MC	20	2.1	0.0027
MTS–MC	40	2.1	0.0014

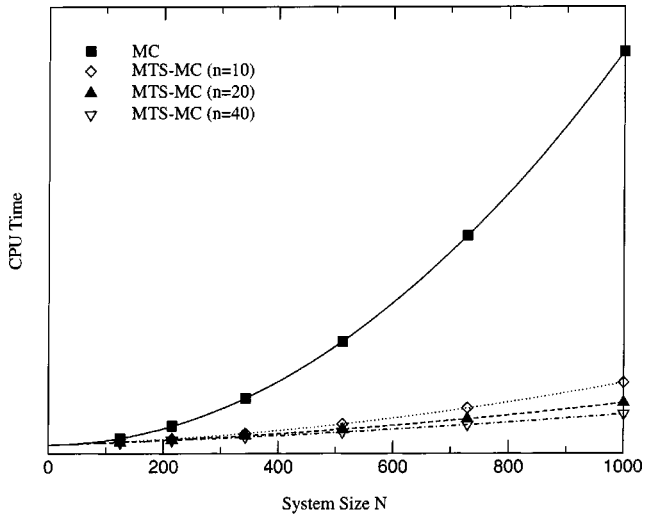

FIG. 1. CPU time vs system size for regular MC and MTS–MC with different values of the splitting parameter. The lines are least squares fits of the function $aN+bN^2$ (see Table IV).

V. CONCLUSIONS

We have devised a modified MC algorithm similar in spirit to multiple time-step MD methods. Our method is based on splitting the potential into two parts, one slowly varying and the other quickly varying as a function of coordinates. As in multiple time-step MD, the slowly varying part of the potential is evaluated less frequently than the short-range part, leading to savings in CPU time.

We have demonstrated that our procedure obeys detailed balance, and have implemented the method for a Lennard-Jones system. Our calculations indicate savings in CPU time up to factors of 5.6 for our largest system of 1000 particles. We have also studied the scaling of CPU time as a function of system size, and have shown that for the sizes investigated here the linear term can be made to be the dominant term. In a future investigation, we apply MTS–MC to systems with long-range interactions like water using the Ewald and particle mesh Ewald methods.[11]

ACKNOWLEDGMENTS

The authors thank Dr. H. A. Stern for useful discussions. This work was supported by Grant No. CHE-00-76279 from the National Science Foundation.

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. M. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
[2] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford Science, Oxford, 1987).
[3] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications* (Academic, New York, 1996).
[4] M. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **97**, 1990 (1992).
[5] J. C. Sexton and D. H. Weingarten, Nucl. Phys. B **380**, 665 (1992).
[6] S. J. Stuart, R. Zhou, and B. J. Berne, J. Chem. Phys. **105**, 1426 (1996).
[7] R. Zhou, S. J. Stuart, and B. J. Berne, J. Chem. Phys. **105**, 235 (1996).
[8] M. G. Martin, B. Chen, and J. I. Siepmann, J. Chem. Phys. **108**, 3383 (1997).
[9] B. Chen and J. I. Siepmann, Theor. Chem. Acc. **103**, 87 (1999).
[10] S. W. Rick, S. J. Stuart, and B. J. Berne, J. Chem. Phys. **101**, 6141 (1994).
[11] K. Bernacki, B. Hetényi, and B. J. Berne (unpublished).
[12] W. B. Streett, D. J. Tildesley, and G. Saville, ''Computer modelling of matter,'' Vol. 86 of *ACS Symposium Series* (American Chemical Society, Washington, D.C., 1978), p. 144.
[13] W. B. Streett, D. J. Tildesley, and G. Saville, Mol. Phys. **35**, 639 (1978).
[14] M. E. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. **47**, 6811 (1991).
[15] M. Rao, C. Pangali, and B. J. Berne, Mol. Phys. **37**, 4056 (1979).
[16] M. Rao, C. Pangali, and B. J. Berne, Mol. Phys. **40**, 661 (1980).