Molecular Physics, 1999, Vol. 97, No. 12, 1243–1254

Recoil growth algorithm for chain molecules with continuous interactions

STYLIANI CONSTA[1], THIJS J. H. VLUGT[2], JOCHEM WICHERS HOETH[2], BEREND SMIT[2] and DAAN FRENKEL[1,2]*

[1] FOM Institute of Atomic and Molecular Physics (AMOLF), Kruislaan 407, 1098 SJ Amsterdam, The Netherlands

[2] Department of Chemical Engineering, University of Amsterdam, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands

(*Received 7 July 1999; revised version accepted 9 August 1999*)

The recoil growth (RG) scheme is a dynamic Monte Carlo algorithm that has been suggested as an improvement over the configurational bias Monte Carlo (CBMC) method (Consta, S., Wilding, N. B., Frenkel, D. and Alexandrowicz, Z., 1999, *J. chem. Phys.*, **110**, 3220). The RG method had originally been tested for hard core polymers on a lattice, and it was found that RG outperforms CBMC for dense systems and long chain molecules. In the present paper, the RG scheme is extended to the practically more relevant case of off-lattice chain molecules with continuous interactions. It is found that for longer chain molecules RG becomes over an order of magnitude more efficient than CBMC. However, other schemes are better suited to the computation of the excess chemical potential. Moreover, it is more difficult to parallelize RG than CBMC.

1. Introduction

Computer simulations help us to relate the macroscopic properties of polymers to the atomic structure of these molecules. Simulations are a useful aid in the interpretation of experimental data and allow us to gain a better insight into the validity of theoretical models. In all many-body simulations, it is essential to perform adequate sampling of the phase space of the model system. This becomes problematic for systems of long chain molecules, in particular at high densities. In fact, the slow sampling of phase space occurs also for real polymers: at high densities the natural ('reptation') dynamics of long polymer chains is very slow. Molecular dynamics simulations aim to mimic the natural dynamics of a system, and therefore suffer from the same problems. However, in Monte Carlo (MC) simulations we are not constrained to sample phase space (or actually the configuration space) using natural dynamics. This is why, for dense polymer systems, Monte Carlo methods have the potential to be more efficient than molecular dynamics.

Many MC schemes have been proposed to sample the configuration space of both isolated polymers and moderately dense polymeric systems. Among these methods we should distinguish between static Monte Carlo schemes, in which many independent polymer configurations are generated from scratch, and dynamic (Markov chain) MC schemes that accept or reject new chain conformations by comparing their 'weight' (in the simplest cases: Boltzmann weight) with that of the old configuration. Examples of static MC schemes that can sample configurations of long polymer chains are the single- [1] and double-scanning [2] methods of Meirovitch and, in particular, the pruned-enriched Rosenbluth (PERM) method of Grassberger [3]. Among the dynamic sampling schemes, the configurational bias Monte Carlo (CBMC) [4–7] has found many applications [8–10]. CBMC uses the algorithm suggested by Rosenbluth and Rosenbluth [11] for generating a configuration of a chain molecule. In CBMC a chain conformation is grown segment by segment. For each chain segment that has to be inserted several trial segments are generated, and one of these segments is selected with a probability proportional to its Boltzmann factor. This introduces a bias in the growth of the chain, which can be removed exactly by a modification of the acceptance/rejection rule [7]. In this way, chain configurations are generated with the correct Boltzmann weight. An important disadvantage of CBMC is that in the growth of the chain only one step ahead is examined, which means that one cannot avoid that the growth of the chain may lead into a 'dead-end street'. To compensate for this high attrition rate in the chain construction, one

* Author for correspondence. e-mail: frenkel@amolf.nl

is forced to use a large number of trial directions. For hard core interactions, this leads to a broad distribution of the Rosenbluth weight and therefore low acceptance rates [12]. The recoil growth (RG) [12] algorithm, first applied to polymers with hard core interactions, has been suggested as an alternative to CBMC which does not suffer from the 'short-sightedness' of CBMC. The RG method is a 'look-ahead' dynamic MC scheme (in contrast to the static schemes of [1–3]). In this algorithm, a chain is extended by one segment provided that a pathway (also called feeler) of certain length ahead of the current segment can be grown successfully. However, the feeler may fail to grow because of overlaps with other segments. In that case, the feeler can recoil up to the part of the chain that has been grown successfully. There are two parameters that one has to tune to obtain good construction and acceptance rates: the length of the feeler and the number of trial directions. For hard core chains it was found that a small number of trial directions in combination with a relatively large recoil length is optimal. This is because a small number of trial directions ensures that the distribution of the Rosenbluth weight is small, which increases the acceptance probability of the chain. The consequence of a large feeler is that the accessible part of the phase space for a segment of the chain is much larger than for CBMC, which means that it is more likely that a favourable configuration will be found. Just as for CBMC, RG introduces a bias in the generation of a chain which can be removed exactly by a modification of the acceptance/rejection rule. In [12] the RG algorithm was tested for self-avoiding walks on a lattice and its efficiency was compared with CBMC. It was found that for low densities (e.g., 30% occupancy of the lattice), CBMC performs better than RG for both short and long polymer chains. For higher densities and chain lengths longer than 20 segments, RG is an order of magnitude more efficient than CBMC.

The purpose of the present paper is to extend the RG algorithm to off-lattice chain molecules with continuous interactions. Although the algorithm is inspired by the one for hard core interactions, there are several important differences.

In section 2 the RG algorithm for the case of continuous interactions is described, while in section 2.3 it is shown that RG can be reduced to CBMC only for hard core potentials. Section 3.2 contains a study of the efficiency compared with CBMC and suggestions for the optimization of the method. In appendix A we present an alternative (but less efficient) algorithm to compute the Rosenbluth weight of a chain. In appendix B we explore the performance of RG for the computation of the excess chemical potential. We find that RG is less efficient than the 'Rosenbluth' scheme for computing the excess chemical potential and, *a fortiori* less efficient than the PERM scheme of [3]. In appendix C we show that there is little point in parallelizing the RG scheme using a 'multiple chain' algorithm.

2. Description of the algorithm

The RG algorithm for hard core chains is described in detail in [12]. Here we discuss extensions of this algorithm for continuous potentials. In what follows we describe the RG scheme for canonical MC in which M chains of length N are sampled. It is straightforward to extend this algorithm to other ensembles [7, 13].

2.1. *Construction of a chain*

The central step in both CBMC and RG is the selection of a specific polymer trial conformation from an entire 'tree' of possible conformations. The essential difference between the 'continuous-potential' RG method and the earlier schemes is that the selection of the trial conformation involves two stochastic steps: the first is the selection of a subset of open branches on the tree, the second is the selection of the trial conformation among the open branches. The crucial new concept in RG is that trial directions can be either open or closed. A trial direction that is closed will never be chosen as a part of the chain. For hard core potentials, a trial direction is closed if it leads to a configuration that has at least one hard core overlap, otherwise it is open. Therefore, the selection of the open trial directions is deterministic rather than stochastic. By contrast, for continuous potentials we use a stochastic rule to decide whether a trial direction is open or closed. The probability p_i that direction i is open depends on its energy u_i, hence $p_i = p_i(u_i)$. It is important to note that, in principle, this stochastic rule is quite arbitrary, and the only restriction is $0 < p_i \leqslant 1$ (for hard core potentials, $0 \leqslant p_i \leqslant 1$). However, it is useful to apply the following restrictions

$$\lim_{u_i \to \infty} p_i(u_i) = 0,$$
$$\lim_{u_i \to -\infty} p_i(u_i) = 1. \tag{1}$$

An obvious choice that obeys these restrictions is the standard Metropolis acceptance/rejection rule [7, 13, 14]

$$p_i(u_i) = \min(1, \exp[-\beta u_i]), \tag{2}$$

in which $\beta = 1/k_B T$. For hard core potentials, p_{open} is either equal to 0 (at least one overlap) or 1 (no overlaps). Once we have determined the set of open trial directions, the RG algorithm for a chain with continuous interactions becomes almost identical to that for a hard core chain.

(1) To start, the first segment of a chain is placed at a random position in the system. If the first position is open, we continue with the next step. Otherwise, the chain is discarded.
(2) A direction is assigned randomly to a segment i $(i > 1)$. If this direction leads to overlap with another segment in the system, another direction is tried, up to a maximum of k trial directions. In principle, k can vary with i. In fact, it can even be a stochastic variable.
(3) If an open direction is found, a new segment is added and the number of the directions for segment i that have not yet been explored is recorded. If all directions are blocked the chain retracts by one step to segment $i-1$, and the unused directions are explored. The chain is allowed to recoil up to length $(l_{\max} - l + 1)$ where $l_{\max}$ is the maximum length that the chain has attained in its growth history, and l is the recoil length, which is a fixed simulation parameter. When a chain is not allowed to recoil the entire chain is discarded.
(4) The previous steps are repeated until the complete chain has been grown. After the successful construction of a chain, the weight of the new chain $(W(\mathrm{n}))$ is computed. This weight will be needed in step 6 to determine whether or not the new conformation will be accepted. The computation of W will be discussed in the next section.
(5) For the old chain, on every segment of the old chain $k-1$ feelers of length l are grown and the number of feelers that is grown successfully is recorded. Using this information, one can compute the weight of the old configuration $(W(\mathrm{o}))$.
(6) The new configuration is accepted with a probability

$$P_{\mathrm{acc}}(\mathrm{o} \to \mathrm{n}) = \min\left(1, \frac{W(\mathrm{n})}{W(\mathrm{o})}\right). \tag{3}$$

In the next sections, we derive the form for $W(\mathrm{n})$ and $W(\mathrm{o})$ that is required in order that the MC algorithm obeys detailed balance.

2.2. *Detailed balance condition and acceptance probability*

In a Markov-chain Monte Carlo scheme we need to ensure that different points in configuration space are visited with a frequency proportional to their Boltzmann weight. Usually this is achieved by imposing the detailed balance condition

$$N(\mathrm{o})P(\mathrm{o} \to \mathrm{n}) = N(\mathrm{n})P(\mathrm{n} \to \mathrm{o}), \tag{4}$$

where $N(i)$ is the Boltzmann weight of state i and $P(\mathrm{o} \to \mathrm{n})$ is the transition probability from state o (old configuration) to state n (new configuration).

In the RG algorithm the transition probability includes the construction of a particular tree of trial segments for both the new and the original states, the stochastic choice of the subset of open segments in both trees, the selection of a particular configuration among the branches of the new tree, and finally the probability of accepting the new configuration. The transition probability from the old to the new state is given by

$$\begin{aligned} P(\mathrm{o} \to \mathrm{n}) = \sum_{t_\mathrm{o}, t_\mathrm{n}, O_\mathrm{o}, O_\mathrm{n}} & P_\mathrm{g}(t_\mathrm{n}) P_\mathrm{g}(O_\mathrm{n}|t_\mathrm{n}) P_\mathrm{g}(rw_\mathrm{n}|t_\mathrm{n}, O_\mathrm{n}) \\ & \times P_\mathrm{g}(t_\mathrm{o}|rw_\mathrm{o}) P_\mathrm{g}(O_\mathrm{o}|t_\mathrm{o}, rw_\mathrm{o}) \\ & \times P_{\mathrm{acc}}(\mathrm{o} \to \mathrm{n}), \end{aligned} \tag{5}$$

where $P_\mathrm{g}(t_\mathrm{n})$ is the probability of generating the new tree t_n. In what follows we shall consider the case that this probability is uniform. However, in general, when simulating molecules with internal (bond and torsion) potentials, it may be advantageous to generate trial segments according to the intramolecular Boltzmann distribution. Alternatively, it is also possible to take bonded intramolecular interactions into account in equation (2). However, this will lead to an inefficient algorithm because in that case many trial directions will be found closed on the basis of their internal energy. $P_\mathrm{g}(O_\mathrm{n}|t_\mathrm{n})$ is the probability of selecting a particular set of open/closed directions (O_n) on the new tree t_n. $P(rw_\mathrm{n}|t_\mathrm{n}, O_\mathrm{n})$ is the probability of generating a random walk (rw) on the set of open/closed directions O_n of the new tree t_n. This factor is equal to

$$P(rw_\mathrm{n}|t_\mathrm{n}, O_\mathrm{n}) = \frac{1}{\prod_{i=1}^{i=N} m_i}, \tag{6}$$

in which m_i is the number of successfully grown feelers at position i in the chain. m_i is always larger than zero, because if it were not the trial move would not have resulted in any new configuration, and it would have been rejected. The next two terms are related to the old configuration: $P_\mathrm{g}(t_\mathrm{o}|rw_\mathrm{o})$ is the probability of generating a tree around the old configuration (i.e., the old configuration is included in the tree, the other configurations are generated) and $P_\mathrm{g}(O_\mathrm{o}|t_\mathrm{o}, rw_\mathrm{o})$ is the probability of selecting a particular set of open/closed directions on this tree; however, the old configuration of the chain is always ‘open’. Finally, the term $P_{\mathrm{acc}}(\mathrm{o} \to \mathrm{n})$ is the probability that the transition from o to n is accepted. The transition probability from the new (n) to the old (o) state is written as equation (5) by exchanging o and n. Detailed balance is satisfied by imposing the stronger condition of super-detailed balance [7], which means that we obey detailed balance for all possible sets of trees and open/closed directions of both the new and old configurations $([t_o, t_n, O_o, O_n])$. Many factors in $P_g(O_n|t_n, rw_n)$ that are found in the transition probability from the new (n) to the old (o) state cancel with the factors of $P_g(O_n|t_n)$ which are in the expression for the transition probability from the old (o) to the new (n) state. The only remaining term is the probability of generating open directions along the backbone of the tree. Therefore, the correct acceptance/rejection rule is given by

$$P_{\mathrm{acc}}(\mathrm{o} \to \mathrm{n}) = \min\left(1, \frac{W(\mathrm{n})}{W(\mathrm{o})}\right), \tag{7}$$

in which $W(\mathrm{n})$ is

$$W(n) = \frac{\exp[-\beta u(n)] \prod_{i=1}^{i=N} \dfrac{m_i(\mathrm{n})}{p_i(\mathrm{n})}}{fk^{N-1}}. \tag{8}$$

In this equation, f is the number of trial positions for the first segment, which is equal to 1 when the first segment of the chain is placed at a random position in the system. Note that the term fk^{N-1} is present in both the numerator and denominator of equation (7) and is therefore irrelevant. We include this term to emphasize the similarity of W with the Rosenbluth weight [11]. We obtain the expression for $W(\mathrm{o})$, by exchanging n and o. If we choose equation (2) as our stochastic rule, there is complete cancellation of Boltzmann factors associated with the selected trial segments (i) as long as $u_i \geqslant 0$. For hard core interactions $p_i = 1$, in which case the algorithm reduces to the RG algorithm for hard core potentials [12].

For the simulation of branched chain molecules, there will be an additional term in equation (5) for the probability of selecting a random growth path on the branched molecule. As this probability is uniform, this does not influence the final detailed balance expression (equation (8)). For the simulation of branched molecules with bonded intramolecular interactions special techniques like the coupled-decoupled CBMC method by Siepmann and coworker [8] may be required.

An alternative scheme to compute the weights $W(\mathrm{n})$ and $W(\mathrm{o})$ can be found in appendix A. As this scheme is more complex and less efficient than the scheme presented above, we will not discuss it in the main text of the paper.

2.3. *Comparison with CBMC*

It is instructive to compare RG with CBMC for when the recoil length l is equal to 1. In [12] it was explained that, for hard core potentials, RG and CBMC become identical when $l = 1$. Below we show that this is not the case for continuous potentials. In other words, RG is not simply a generalization of CBMC. In CBMC we retain all possible trial directions and then select a particular direction i with a probability proportional to its Boltzmann weight $b_i = \exp[-\beta u_i]$. For models with continuous interactions, $b_i > 0$ (even though it may be very small). Hence, in a naive implementation of the CBMC scheme, the growth of a trial configuration will be completed, no matter how small b_i is (see, however, [15, 16]). Of course, in the acceptance step, conformations with a very low weight will most probably be rejected. By contrast, in the RG scheme unlikely configurations are weeded out at an early stage because, most probably they will be 'closed'. One might think that RG would become similar to CBMC if we do not allow trial segments to be closed (i.e., if $p_i(u_i)$ is always equal to one). However, if we do that, all generated configurations are equally likely to be selected, irrespective of their Boltzmann weight. Clearly, that would be much worse than CBMC (unless $k = 1$, in which case both schemes reduce to the worst possible algorithm, i.e., random insertion). Otherwise, RG is only equivalent to CBMC in the case that $l = 1$ provided that all configurations that have a nonzero Boltzmann weight, do in fact have the same Boltzmann weight. Clearly, this condition is fulfilled for hard core potentials. However, in general, RG and CBMC are based on different stochastic rules to generate trial configurations.

3. Simulations

3.1. *Simulation details*

To study the efficiency of the continuous-potential RG method, we have performed NVT Monte Carlo simulations of M linear chains (length N) with truncated Lennard-Jones interactions between the non-bonded segments. In reduced units (well depth $\varepsilon = 1$, Lennard-Jones diameter $\sigma = 1$), the truncated LJ potential has the following form:

$$\begin{aligned} u &= 4\left[\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right] & r &\leqslant r_{\mathrm{cut}}, \\ u &= 0 & r &> r_{\mathrm{cut}}. \end{aligned} \tag{9}$$

We have used $r_{\mathrm{cut}} = 2.5$. The bond length between two successive segments was chosen to be 1.0. Three successive segments of a molecule have a constant bond angle of 2.0 rad. Intramolecular non-bonded interactions were taken into account for segments that are separated by more than two bonds.

To enhance the efficiency of both the CBMC and the RG schemes, we have divided the intermolecular potential energy into short range and long range parts,

$$u = \bar{u}(r < r^*_{\text{cut}}) + \delta(r^*_{\text{cut}} \leqslant r < r_{\text{cut}}), \qquad (10)$$

in which r^*_{cut} is a second cutoff radius. As shown in [17, 18], the repulsive part of the potential is most important for the generation of a chain. For CBMC we generate chain segments only with the short range part of the potential and correct for the bias afterwards [17, 18]. For RG we use only the short range part of the potential to decide whether a segment is open or closed. In this study, we have used

$$p_{\text{open}} = \min(1, \exp[-\beta(\bar{u}_{\text{inter}}(r < r^*_{\text{cut}}) + u_{\text{intra}})]), \qquad (11)$$

where u_{intra} is the intramolecular energy.

Since the generation of a chain uses only the short range part of the potential, we can make a linked cell list to calculate the potential efficiently. Such a cell list has to be updated only after an accepted trial move. As the number of interactions within the short range part of the potential is usually quite small, we can calculate the energy of a trial position very quickly. It was found that for CBMC, dependent on the system, this will result in a speed-up by a factor 2–5. One consequence of this trick is that most of the CPU time will be spent in calculating the long range part of the potential energy, as the construction of a chain is very quick. As a consequence, the overall efficiency of the simulation does not depend strongly on the choice of the simulation parameters (f, k, l). This makes it difficult (but also less relevant) to find (or predict) the optimal values for (f, k, l).

When the first segment of a new chain is placed at an unfavourable position it is not very likely that this trial-move will be accepted. Therefore we use CBMC in the selection of a position for the first segment [19], for which we use f trial positions. One of these positions is selected with a probability proportional to its Boltzmann factor. For the old configuration, $f - 1$ trial positions are generated (the fth is the position of the first segment itself). In order to obey detailed balance, one has to multiply the Rosenbluth weight of the new and old configuration (equations (8) and (12)) with the sum of the f Boltzmann factors; see also equation (21) and [8, 10, 19]. Note that there are several other methods to bias the selection of the position of the first segment [10, 20]. The value of f can be chosen to be quite large for CBMC: for example, Martin *et al.* use $f = 10$ [8, 16]. RG is less sensitive to the value of f because, if the first segment is placed at a position with a very unfavourable energy, the chain growth will be terminated immediately.

3.2. *Efficiency of RG compared with CBMC*

To test the efficiency of our algorithm, we have simulated the following systems at $T = 5.0$:

(1) $N = 10, \rho = 0.4, M = 40$
(2) $N = 10, \rho = 0.2, M = 20$
(3) $N = 20, \rho = 0.4, M = 20$
(4) $N = 20, \rho = 0.2, M = 10$
(5) $N = 40, \rho = 0.4, M = 10$
(6) $N = 40, \rho = 0.2, M = 5$

Note that ρ is the segment density. In all our simulations, we have used $r^*_{\text{cut}} = 1.5$. The length of our cubic simulation box was 10.0. We have simulated all systems using CBMC $(f = 5, 10, k = 5, 10, 15)$ and RG $(f = 5, k = 1, 2, 3, 4, 5, l = 1, 2, 3, 4, 5)$. Note that, in principle, k itself can be a stochastic variable [12]; however, here we have kept it fixed. We have performed two different trial moves. (i) Displacement of a chain. A randomly chosen chain is given a random displacement. The maximum displacement is adjusted such that 50% of the trial moves are accepted. (ii) Regrowth of a chain. A randomly chosen chain is regrown at a random position using either CBMC or RG.

In every MC a trial move consists of either a trial regrowth or a trial displacement (both selected with equal probability). The amount of CPU time that is spent in the regrowth trial move is monitored during the simulation. A total simulation consists of 10^5 cycles, i.e., 10^5 trial moves per chain molecule.

First of all, we have checked if the implementation of RG and CBMC is correct. We have found exact agreement in average energies, distribution of the radius of gyration of a chain and also the radial distribution function between RG and CBMC for various simulation parameters. There are different ways to define the efficiency of a simulation. (1) Number of accepted trial moves divided by the CPU time. This definition is often used, but it does not say anything about the effectiveness of accepted trial moves in changing the molecular configuration. (2) Decay of an autocorrelation function that measures the rate of change of molecular conformations. For example, we can measure the decay of the autocorrelation function of the angle between the end-to-end vector of a chain with an arbitrary but constant vector (for example the z axis) as a function of the CPU time. The faster the decay of this function, the faster a new independent configuration is generated.

The second definition is generally preferred, because this one contains not only information about the speed of the algorithm but also information about its effectiveness in sampling configuration space.

Table 1 summarizes the efficiency by both definitions for our six model systems, with their optimal simulation parameter sets for both CBMC and RG. It is found that the ratios of efficiencies of CBMC compared with RG are equal for both definitions. This means that an accepted CBMC move is as effective in changing the molecular configuration of the system as an accepted RG trial move. This is different from lattice simulations of RG and CBMC [12]. For short chains and low density the improvement of RG over CBMC is only marginal. However, for high densities and long chain lengths, RG is an order of magnitude more efficient than CBMC. Of course, even when using RG, the efficiency of the MC scheme decreases quite rapidly at high densities and for long chain lengths. However, using RG clearly we can extend the density and chain-length regime for which MC techniques based on chain regrowth are feasible.

In figure 1 we have plotted the efficiency of the RG scheme as a function of the number of trial directions k for various values of the recoil length l. As is to be expected, the efficiency decreases quite rapidly with increasing chain length and density. The figure shows that, once k, the number of trial directions, is larger than one, more efficiency is gained by increasing the value of the recoil length l than by increasing k. However, at higher densities, it is important to optimize k. Of course, for $k = 1$, the efficiency is independent of the recoil length.

4. Conclusion

In summary, we have extended the recoil growth scheme for systems with continuous potentials. We find that in an NVT simulation RG is much more efficient than CBMC for long chains and high densities.

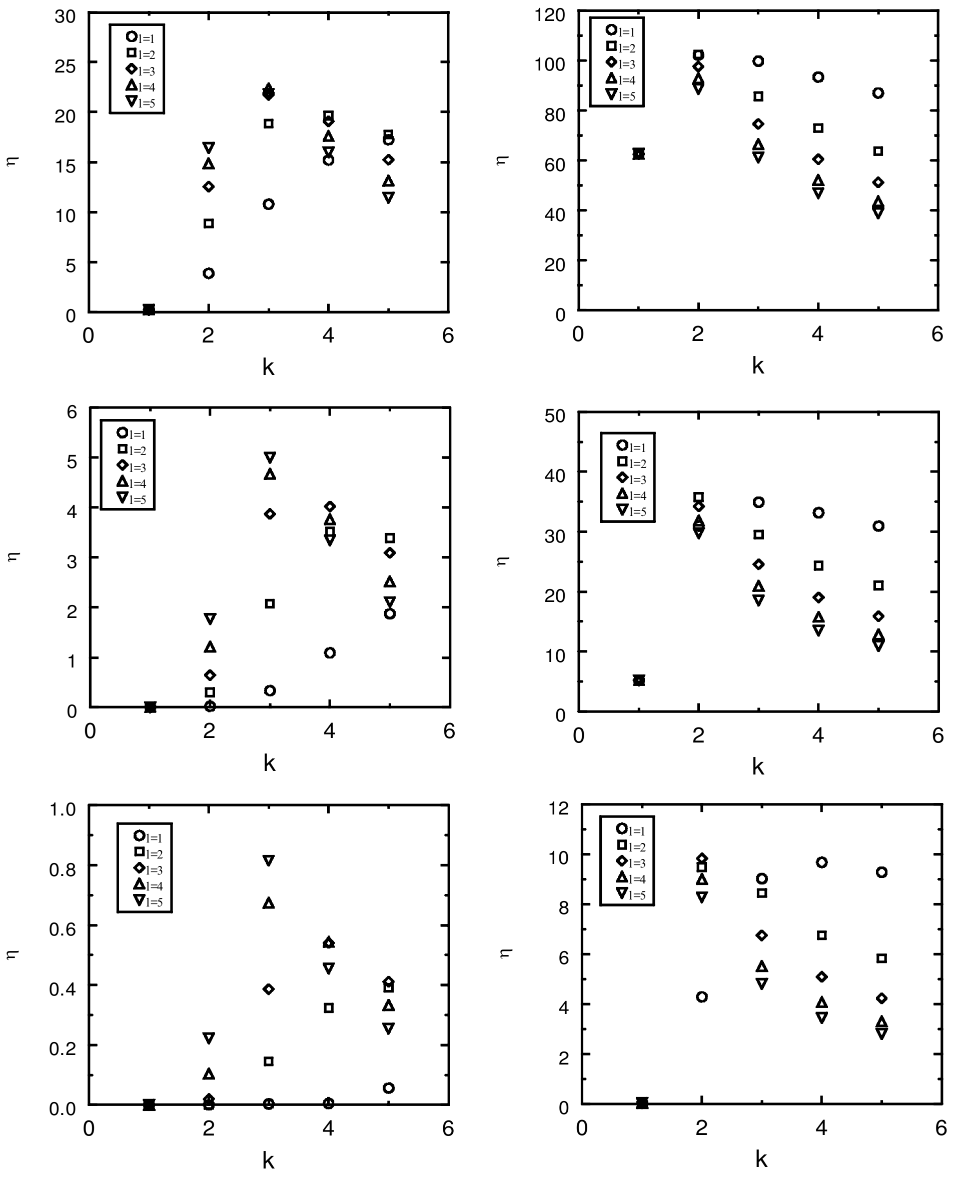

Figure 1. Efficiency (in arbitrary units) as a function of the number of trial directions k for a given recoil length l for the six systems described in section 3.2: left, reduced density $\rho = 0.4$; right, $\rho = 0.2$; top-to-bottom, chain lengths $N = 10$, $N = 20$, $N = 40$; and $f = 5$.

Table 1. Optimal simulation parameters and efficiencies (in arbitrary units) by both definitions (η_1, η_2) for both RG and CBMC. The last two columns are the efficiency ratios for RG and CBMC according to different definitions (1, 2, see text).

				CBMC			RG				
Sys	N	ρ	M	f, k	η_1	η_2	f, k, l	η_1	η_2	(RG/CBMC)$_1$	(RG/CBMC)$_2$
1	10	0.4	40	10,10	5.9	0.071	5,3,4	22.4	0.27	3.8	3.8
2	10	0.2	20	10,5	63.2	1.54	5,2,2	102.5	2.5	1.6	1.7
3	20	0.4	20	10,15	0.58	0.013	5,3,5	5.0	0.12	8.6	8.4
4	20	0.2	10	10,10	16.2	0.748	5,2,2	35.7	1.8	2.2	2.4
5	40	0.4	10	10,15	0.031	0.0017	5,3,5	0.81	0.039	26.0	23.0
6	40	0.2	5	10,10	3.6	0.35	5,2,3	9.83	1.02	2.7	2.9

However, RG is less suitable for parallelization and the computation the excess chemical potential using Widom's test particle method.

The work of the FOM Institute is part of the research program of FOM (Stichting Fundamenteel Onderzoek der Materie) and is supported by NWO (Nederlandse Organisatie voor Wetenschappelijk Onderzoek). S.C. acknowledges the EU for financial support (Contract No. 95032 CGP.4, Contract No. ERBFMBICT-972312). T.J.H.V. acknowledges financial support from NWO-CW (Nederlandse Organisatie voor Wetenschappelijk Onderzoek, Chemische Wetenschappen). B.S. acknowledges support from the Dutch Graduate School of Process Technology (OSPT). A large fraction of the computing resources used in this work were made available by SARA (Stichting Academisch Rekencentrum Amsterdam) and EPCC (Edinburgh Parallel Computing Centre).

Appendix A

Alternative algorithm to compute the weight

Instead of the method described in section 2.2, it is also possible to calculate the probability that a certain path on the tree t is followed explicitly, without having to use terms that represent the probability of generating a set of open/closed directions $(P_g(O_n|t_n), P_g(O_o|t_o, rw_o))$. This means that O_o and O_n do not appear in the super-detailed balance expression (equation (5)). When the probability to follow a path on the tree is equal to Ψ, in order to obey detailed balance and to use equation (7) as acceptance/rejection rule we have to redefine the weight $W(n)$ as

$$W(n) = \frac{\exp[-\beta u(n)]}{\Psi(n) f k^{N-1}}. \tag{A 1}$$

To obtain the correct expression for $W(o)$ we have to replace n with o. To calculate Ψ we have to extend all feelers up to the recoil length l and calculate the probabilities that each of these trial segments is open.

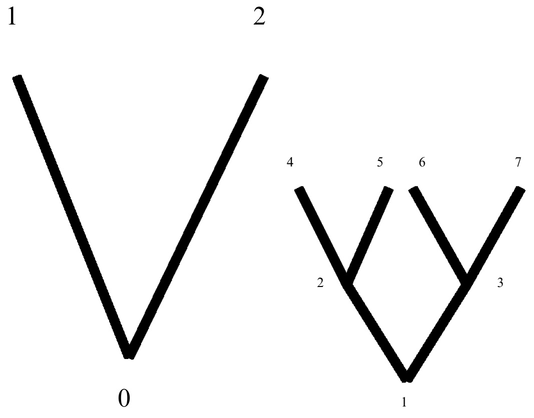

Figure A1. Schematic representation of some simple trees: left, $l = 1$, $k = 2$, and $N = 2$; right, $l = k = 2$ and $N = 3$.

It is instructive to discuss the situation $l = 1$ and $N = 2$. This system is schematically drawn in figure A 1 (left). The probability that the first segment is open is equal to p_0. For the second segment, the probability that we select trial segment 1 is equal to

$$Q = \frac{p_1 p_2}{2} + \frac{p_1(1 - p_2)}{1}, \tag{A 2}$$

in which p_i is the probability that segment i is open. The probability of generating the whole chain (segments 0, 1) then equals

$$\Psi = p_0 Q. \tag{A 3}$$

When the number of trial segments for the second segment is equal to k the number of terms in this equation will be equal to 2^{k-1}. For example, for $k = 3$ the probability of selecting trial segment 1 is

$$Q = \frac{p_1 p_2 p_3}{3} + \frac{p_1 p_2 (1 - p_3)}{2} + \frac{p_1 (1 - p_2) p_3}{2} + \frac{p_1 (1 - p_2)(1 - p_3)}{1}. \tag{A 4}$$

Table A1. Recursive Fortran90 function (F) to compute the probability to select direction `1` from `k` directions (see equations (A 2) and (A 4)). The probability that direction `i` is open is equal to `P(i)`. This function should be called with `N=k` and `Norm=1`.

```
Recursive Double Precision Function F(P,N,Norm) Result(Res)
Implicit None
Integer N,Norm
Double Precision P(*)
If(N.Eq.1) Then
  Res = P(1)/Dble(Norm)
Else
  Res = P(N)*F(P,N-1,Norm+1)
  Res = Res + (1.0d0-P(N))*F(P,N-1,Norm)
Endif

Return
End Function F
```

For arbitrary k it is possible to compute this function recursively, see table A 1. Note that, for hard core potentials, p_i is either equal to 0 or 1. This means that for arbitrary k all terms except one will be equal to zero. The expression for the probability of selecting trial segment 1 will be

$$Q = \frac{1}{m}, \qquad (A\,5)$$

in which m is the number of open directions including direction 1. It is straightforward to see that in this case the algorithm reduces to the standard RG algorithm of hard core potentials [12].

Let us now consider the case that we have to calculate the probability of generating a chain of length $N = 3$. This is different from the previous case only in the way we calculate the probabilities that parts of the tree are open. For example, consider a segment (1) with children 2 and 3. Segment 2 has children 4 and 5 and segment 3 has children 6 and 7. This situation is drawn schematically in figure A 1 (right) and it corresponds to $l = k = 2$. Let us again use p_i for the probability that segment i is open. The probability of following the path 1, 2, 4 along this tree is

$$\Psi = p_1\left[\frac{p_2^* p_3^*}{2} + \frac{p_2^*(1 - p_3^*)}{1}\right]. \qquad (A\,6)$$

In this equation, p_2^* is the probability of growing segment 4 successively from segment 2. The expression for p_2^* is similar to the previous expression:

$$p_2^* = p_2\left[\frac{p_4 p_5}{2} + \frac{p_4(1 - p_5)}{1}\right]. \qquad (A\,7)$$

The physical meaning of p_3^* is the probability of growing either 6 or 7 starting from 3. This means that segment 3 and at least 6 or 7 must be open

$$p_3^* = p_3[1 - (1 - p_6)(1 - p_7)]. \qquad (A\,8)$$

For different chain and recoil lengths we simply have to use the previous expressions in a recursive way: (i) for a part of the tree that is part of the backbone, we have to calculate the probability that the backbone is followed (see table A1); and (ii) for a part of the tree that is not part of the backbone, we have to calculate the probability that at least one trial direction is open. This is of course equal to 1 minus the probability that none of the directions is open; see for example equation (A 8).

A possible way to program this on a computer is to use a parent/child concept, in which every point of the tree has pointers to both its parent and its children. One can use the same recursive operators for a parent and all descendants of the parent.

An important difference with the algorithm in section 2.2 is that we have to extend all feelers up to length l, even if a direction is closed. This means that we have to compute many more trial directions compared with the algorithm in section 2.2. Although it is possible to reduce the fraction of CPU time that is spent in the calculation of the energy of a trial segment [17, 18], this algorithm will always be computationally more expensive than the algorithm in section 2.2. In figure A2, we have plotted the efficiencies of the method described in this appendix and the method described in section 2.2 for $N = M = 20$ and $\rho = 0.4$. It is found that for a large recoil length and number of trial directions the algorithm described in this section becomes less efficient. However, the method described in this appendix is still almost a factor of 5 faster than CBMC for the optimal simulation parameters.

We found that although this algorithm is correct in principle, this method is quite difficult to program. Although every program with recursive functions can be transformed to a program without recursion, we found that this method is extremely difficult to program without this technique. Because of the lower efficiency and the complexity we do not recommend the use of this algorithm. However, there may be other problems where an approach like the one sketched in this appendix is useful.

Appendix B
Calculation of the chemical potential

An important macroscopic quantity is the chemical potential, which can be calculated using Widom's test particle method [21]. When the Rosenbluth scheme is used to grow a test chain, the expression for the excess chemical potential equals [7]

$$\mu_{\text{ex}} = -\frac{\ln\langle W^+\rangle}{\beta}, \qquad (B1)$$

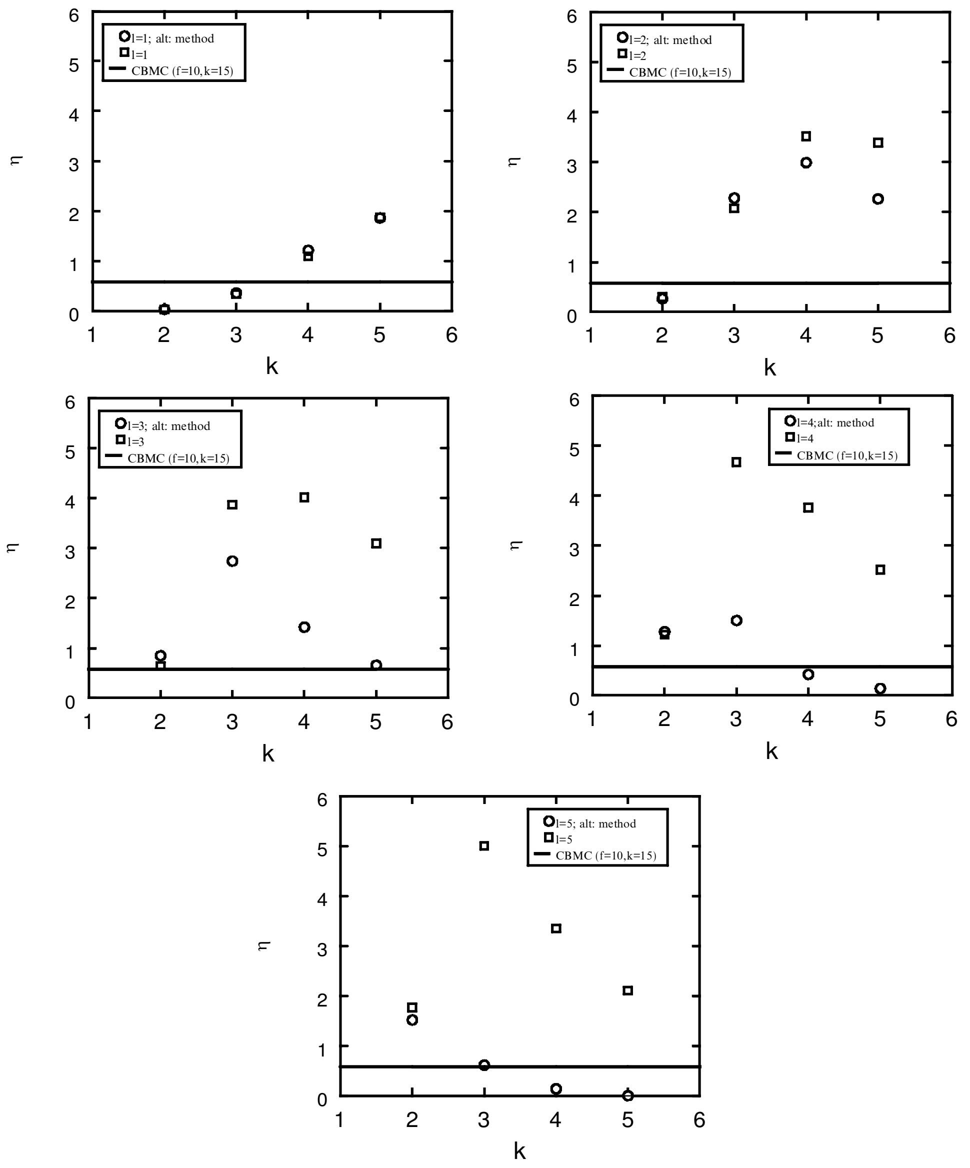

Figure A2. Efficiency (in arbitrary units) as a function of the number of trial directions k for a given recoil length l for the two different algorithms to compute the Rosenbluth weight (section 2.2 and the alternative method of appendix A): $N = M = 20$ and $\rho = 0.4$. Note that for CBMC, the number of trial directions is constant $(f = 10, k = 15)$.

in which W^+ is the normalized Rosenbluth factor for a test chain that is grown using CBMC,

$$W^+ = \frac{\left[\sum_{j=1}^{j=f} \exp[-\beta u(j,1)]\right]\left[\prod_{i=2}^{i=N}\sum_{j=1}^{j=k} \exp[-\beta u(j,i)]\right]}{f k^{N-1}}. \tag{B2}$$

One can prove that equation (B1) holds when a test chain is constructed using RG, in which equation (8) is used to compute the Rosenbluth weight. This proof is quite similar to the proof in [7], in which a similar relation is derived for CBMC.

We will derive the expression for the insertion of one segment (i); the extension to a chain of N segments is straightforward. Let us assume that the number of trial directions is equal to k. We will have to integrate over all possible sets of subtrees (t) and sum over all possible choices of sets of open/closed directions (O) and all possible choices (j) of the segment. For the average Rosenbluth factor, one can write

$$\begin{aligned}\langle W^+ \rangle &= \frac{\left\langle \exp[-\beta u_j] \dfrac{m}{p_j} \right\rangle}{k} \\ &= \int \mathrm{d}\Gamma\, P(t) \sum_O \sum_j P(O \mid t)\, P(rw \mid t, O) \frac{\exp[-\beta u_j] \dfrac{m}{p_j}}{k},\end{aligned} \tag{B3}$$

in which m is the number of open directions and p_j is the probability that segment j is open. Using

$$P(rw|t,O) = \frac{1}{m} \tag{B 4}$$

and

$$p_j = \frac{P(O|t)}{P(O|t,rw)}, \tag{B 5}$$

we obtain

$$\langle W^+ \rangle = \int \mathrm{d}\Gamma P(t) \sum_O \sum_j P(O|t,rw) \frac{\exp[-\beta u_j]}{k}. \tag{B 6}$$

Let us now focus on the term

$$\sum_O \sum_j P(O|t,rw) \exp[-\beta u_j]. \tag{B 7}$$

When $k = 1$ it is straightforward to see that this term equals $\exp[-\beta u_1]$. For $k = 2$, this term is

$$\begin{aligned} &p_2 \exp[-\beta u_1] + p_1 \exp[-\beta u_2] \\ &\quad + (1 - p_2) \exp[-\beta u_1] + (1 - p_1) \exp[-\beta u_2] \\ &\quad\quad = \exp[-\beta u_1] + \exp[-\beta u_2]. \end{aligned} \tag{B 8}$$

To prove that, for arbitrary k,

$$\sum_O \sum_j P(O|t,rw) \exp[-\beta u_j] = \sum_{j=1}^{j=k} \exp[-\beta u_j], \tag{B 9}$$

we will use induction. Suppose that this equation is valid for $k = a$, for the correct expression for $k = a + 1$ there are three contributions.

(1) Segment $a + 1$ is closed. This leads to a contribution

$$(1 - p_{a+1}) \sum_{j=1}^{j=a} \exp[-\beta u_j]. \tag{B 10}$$

(2) Segment $a + 1$ is open, but it is not selected. This leads to a contribution

$$p_{a+1} \sum_{j=1}^{j=a} \exp[-\beta u_j]. \tag{B 11}$$

(3) Segment $a + 1$ is open and it is selected. This leads to a contribution

$$\begin{aligned} &\exp[-\beta u_{a+1}] \times (p_1 p_2 \cdots p_a + (1 - p_1) p_2 \cdots p_a + \cdots \\ &+ (1 - p_1)(1 - p_2) \cdots (1 - p_a)) = \exp[-\beta u_{a+1}]. \end{aligned} \tag{B 12}$$

Clearly, the sum of these three contributions is

$$\sum_{j=1}^{j=a+1} \exp[-\beta u_j], \tag{B 13}$$

which means that we have proved equation (B 9). However, the labelling of all k trial segments in equation (B 6) is quite arbitrary. Therefore, we can write

$$\begin{aligned} \langle W^+ \rangle &= \int \mathrm{d}\Gamma P(t) \exp[-\beta u] \\ &= \frac{Q(N,V,T)}{Q_{\mathrm{id}}(N,V,T)}, \end{aligned} \tag{B 14}$$

in which $Q(N,V,T)$ is the canonical partition function and $Q_{\mathrm{id}}(N,V,T)$ is the partition function of an isolated chain. It is trivial to see that this expression ultimately leads to equation (B 1).

We have calculated the excess chemical potential of a chain of length N in a solvent of monomers $(N = 1)$ at $T = 2.0$ and various densities using test particles. The test chains were grown using either CBMC or RG. The results are presented in figure B1. In all simulations, we have used the same number of trial moves and particle insertions $(2.5 \times 10^6$ for $\rho = 0.2$ and 1.25×10^7 for $\rho = 0.4)$. As expected, we obtain the same excess chemical potential for both RG and CBMC. The excess chemical potential is an almost linear function of the

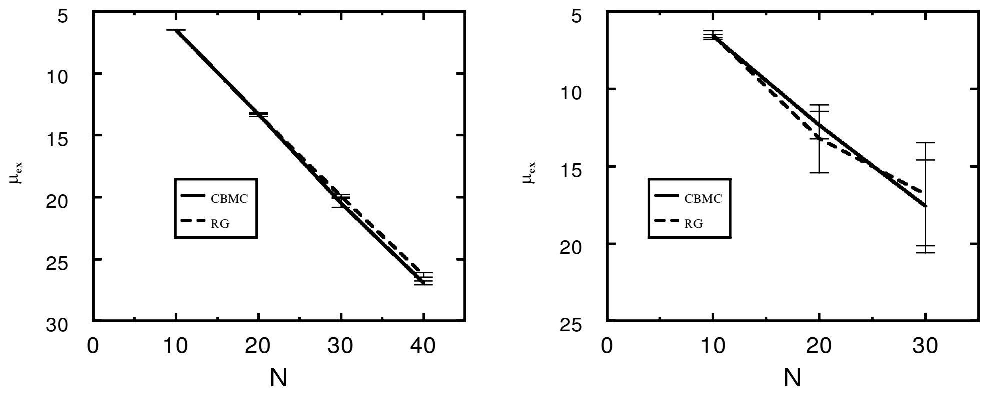

Figure B1. Excess chemical potential (μ_{ex}) of a chain of length N in a solvent of $N = 1$ at $T = 2.0$ calculated using either RG or CBMC. Left: $\rho = 0.2$, right: $\rho = 0.4$.

chain length [22]. Furthermore, the error in the estimate of the excess chemical potential as calculated by block averages [23] is quite large for high densities and large chain lengths. At lower densities, RG seems to perform slightly better than CBMC.

There are several reasons why calculating the chemical potential using RG is more complicated than using CBMC. Let us consider a discrete system of which the phase space consists of only one point with Boltzmann factor a. When CBMC is used to calculate the chemical potential of this system, the average Rosenbluth weight $\langle W^+ \rangle$ will be equal to a; because the phase space is discrete and there is only one point, the standard deviation in the estimate of $\langle W^+ \rangle$ will be equal to zero. However, when RG is used, there is a certain probability p that this discrete position is open. When this position is open, there will be a contribution a/p, otherwise this contribution will be zero. It is straightforward to see that in this case the average Rosenbluth weight is equal to a as well, but when $p \neq 1$ we will have a binomial process in which the error in the estimate of $\langle W^+ \rangle$ will be proportional to the inverse of the square-root of the number of attempts. When the number of discrete points in the phase space is larger than one, we can use the same analysis to show that the error in the estimate of $\langle W^+ \rangle$ will always be larger for RG than for CBMC when $p_i \neq 1$. The previous analysis suggests that statistical errors will be less when the probability that a segment is open is close to 1. However, when this probability is exactly equal to one, the algorithm reduces to a completely random insertion of a chain molecule,

$$\mu_{\mathrm{ex}} = -\frac{\ln \langle \exp[-\beta u^+] \rangle}{\beta}, \qquad (\mathrm{B}\,15)$$

in which u^+ is the energy of the test chain. For high densities and long chains, complete random insertion of test particles leads to very poor statistics because nearly always a position with an overlap with another segment is generated [7]. When p_i is close to 0, most of the trial chains will be found closed and no statistics at all will be collected. Therefore, one needs to fine-tune the stochastic rule to decide what is open or closed. A possible choice would be a modification of the Metropolis acceptance/rejection rule

$$p_i = \min(1, \exp[-\beta^*(u_i + u^*)]), \qquad (\mathrm{B}\,16)$$

in which β^* and u^* are constants that can be optimized to obtain good accuracy for the estimate of μ_{ex}. This function is plotted in figure B2 for a LJ potential. It turns out that β^* makes the transition from $p = 0$ to $p = 1$ larger while u^* is responsible for a shift of this transition. We have tested the use of a modified stochastic rule (equation (B 16)) and we found that this leads to only a marginal improvement in the error in the estimate of μ_{ex} for $\rho = 0.2$ and $\rho = 0.4$. Also, a full optimization of β^* and u^* is computationally too expensive compared with the improvement in the estimate of μ_{ex}. For higher densities and longer chain lengths, however, we found that such an optimization is necessary to obtain a good estimate of μ_{ex}.

Another reason why RG does not perform too well in calculating the chemical potential is that the quality of a trial direction is reduced to a binomial process: a direction is either open or closed. This is completely different for CBMC, where more detailed information about the trial directions (in this case the Boltzmann factor) is collected (see equation (B2)). For a dynamic MC simulation, the lack of detailed information about the surrounding of a segment does not create any problems. For test particles however, this is different because, ideally, one would like to sample all possible trial directions of the test chain [7].

Therefore, we conclude that other algorithms, such as the Rosenbluth scheme, but even more the pruned-enriched Rosenbluth method by Grassberger [3] are more suitable (i.e., more efficient) for computing the excess chemical potential.

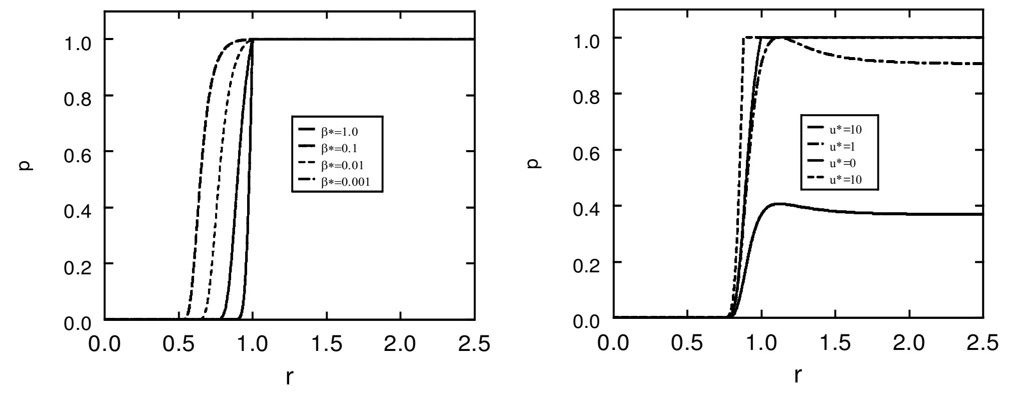

Figure B2. Probability that a segment is open as a function of the inter-atomic distance for the LJ potential ($r_{\mathrm{cut}} = 2.5$, $\sigma = 1$ and $\varepsilon = 1$) according to equation (B 16): left: $u^* = 0$; right: $\beta^* = 0.1$.

Appendix C
Parallelization

With the increasing availability of parallel computers, interest in parallel MC algorithms is growing. Esselink *et al.* [19] have designed a parallel CBMC algorithm that has been optimized and tested on a parallel computer by Vlugt [18]. It was found that, although the construction of a single chain cannot be parallelized efficiently on a large number of processors, one can construct multiple chains instead. This task can be divided readily among processors. One of these chains is selected as the new configuration, while the remaining chains are thrown away. This introduces a bias is the generation of the new configuration, which can be removed exactly by a modification of the acceptance/rejection rule. It was found that for a typical simulation this algorithm works quite well on up to 16 processors on various parallel computers [18].

In such a multiple chain algorithm, it is essential to have a good load balance. This means that every processor is doing roughly the same amount of work. Therefore, the distribution in CPU time for the construction of a chain should be as small as possible. In figure C1, we have plotted the fraction of chains that is grown successfully as a function of the maximum chain length that has been attained during the construction of a chain for various chain lengths (systems 1, 3, 5 of section 3.2). Note that in CBMC, a chain is discarded, for numerical reasons, only when the Rosenbluth weight is of the order of the machine accuracy of the computer (roughly 2×10^{-308} for the computer used in this study). It turns out that the distribution in CPU time is much wider for RG than for CBMC. The fact that in RG many configurations can be thrown away before the complete chain is constructed is one of the main reasons why RG is more efficient than CBMC. However, it also implies that multiple chain algorithms cannot be parallelized efficiently, which makes RG less suitable for parallelization.

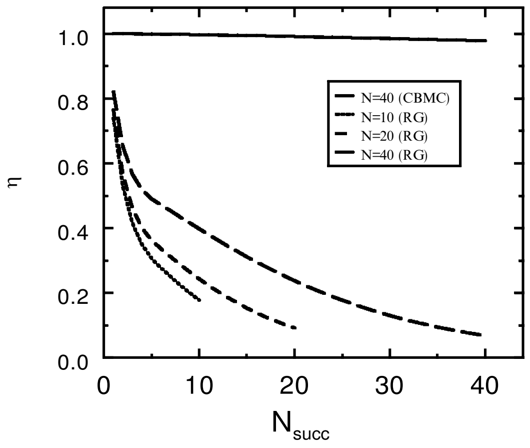

Figure C1. Fraction of chains η that are successfully grown up to length N_{succ} as a function of N_{succ} for $\rho = 0.4$ and $T = 5.0$. For RG we have used $f = 5$, $l = 5$, and $k = 3$, and for CBMC we have used $f = k = 10$.

References

[1] Meirovitch, H., 1982, *J. Phys. A*, **15**, L735.
[2] Meirovitch, H., 1988, *J. chem. Phys.*, **89**, 2514.
[3] Grassberger, P., 1997, *Phys. Rev. E*, **56**, 3682.
[4] Siepmann, J. I., and Frenkel, D., 1992, *Molec. Phys.*, **75**, 59.
[5] Frenkel, D., Mooij, G. C. A. M., and Smit, B., 1992, *J. Phys. condens Matter*, **4**, 3053.
[6] de Pablo, J. J., Laso, M., and Suter, U. W., 1992, *J. chem. Phys.*, **96**, 6157.
[7] Frenkel, D., and Smit, B., 1996, *Understanding Molecular Simulations: from Algorithms to Applications* (San Diego: Academic Press).
[8] Martin, M. G., and Siepmann, J. I., 1999, *J. phys. Chem.* B, **103**, 4508.
[9] Vlugt, T. J. H., Krishna, R., and Smit, B., 1999, *J. phys. Chem.* B, **103**, 1102.
[10] Macedonia, M. D., and Maginn, E. J., 1999, *Molec. Phys.*, **96**, 1375.
[11] Rosenbluth, M. N., and Rosenbluth, A. W., 1955, *J. chem. Phys.*, **23**, 356.
[12] Consta, S., Wilding, N. B., Frenkel, D., and Alexandrowicz, Z., 1999, *J. chem. Phys.*, **110**, 3220.
[13] Allen, M. P., and Tildesley, D. J., 1987, *Computer Simulation of Liquids* (Oxford: Clarendon Press).
[14] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. N., and Teller, E., 1953, *J. chem. Phys.*, **21**, 1087.
[15] Frenkel, D., 1993, *Computer Simulation, in Chemical Physics*, edited by M. P. Allen, and D. J. Tildesley (Dordrecht: Kluwer), p. 93.
[16] Martin, M. G., and Siepmann, J. I., 1998, *J. phys. Chem.* B, **102**, 2569.
[17] Vlugt, T. J. H., Martin, M. G., Smit, B., Siepmann, J. I., and Krishna, R., 1998, *Molec. Phys.*, **94**, 727.
[18] Vlugt, T. J. H., 1999, *Molec. Simulation* in press.
[19] Esselink, K., Loyens, L. D. J. C., and Smit, B., 1995, *Phys. Rev.* E, **51**, 1560.
[20] Snurr, R. Q., Bell, A. T., and Theodorou, D. R., 1993, *J. phys. Chem.*, **97**, 13 742.
[21] Widom, B., 1963, *J. chem. Phys.*, **39**, 2802.
[22] Spyriouni, T., Economou, I. G., and Theodorou, D. N., 1997, *Macromolecules*, **30**, 4744.
[23] Flyvbjerg, H., and Petersen, H. G., 1989, *J. chem. Phys.*, **91**, 461.