MOLECULAR PHYSICS, 1998, VOL. 94, NO. 4, 727–733

Improving the efficiency of the configurational-bias Monte Carlo algorithm

By T. J. H. VLUGT[1], M. G. MARTIN[2], B. SMIT[1], J. I. SIEPMANN[2] and R. KRISHNA[1]

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

[2] Department of Chemistry, University of Minnesota, 207 Pleasant Street SE, Minneapolis, MN 55455-0431, USA

(*Received 17 November 1997; revised version accepted 24 February 1998*)

Algorithms are presented to improve the efficiency for the generation of trial orientations and for the calculation of the Rosenbluth weight in a configurational-bias Monte Carlo (CBMC) simulation. These algorithms were tested for NpT and NVT simulations of n-octane, 3-methylheptane, and 3,4-dimethylhexane at different temperatures and densities using a preliminary version of the TraPPE united-atom representation for the CH_3, CH_2 and CH groups. It was found that for a system of 144 molecules these algorithms speed up the calculation three times for n-octane and almost four times for 3,4-dimethylhexane, resulting in a decreased difference in simulation time between a branched molecule and a linear isomer. For larger systems the speedup is even greater. It is shown that the excluded volume of an atom is the dominant term for the selection of a trial orientation, which leads to an improved CBMC algorithm called dual cutoff configurational-bias Monte Carlo (DC-CBMC).

1. Introduction

Configurational-bias Monte Carlo (CBMC) [1–5] is an advanced simulation technique for complex molecules. CBMC simulations are used frequently for the calculation of vapour–liquid equilibria of linear alkanes [6–8], branched alkanes [9–11], and for the simulation of adsorption of alkanes within porous structures [12–16]. These simulations are still computationally expensive. The architecture of the molecule has a large influence on the simulation time and the acceptance probability for a CBMC move [9]. Therefore we have developed new algorithms which not only give an overall speedup, but also decrease the difference in CPU time between a linear molecule and its branched isomers.

To explain the improved algorithms, the basics of CBMC are summarized below. A more detailed discussion can be found in [4]. In the CBMC scheme it is convenient to split the total potential energy of a trial site into two parts. The first part is the internal, bonded, intramolecular potential (u^{internal}) which is used for the generation of trial orientations. The internal potential often has the form [17, 18]

$$u^{\text{internal}} = \sum u^{\text{bend}}(\theta) + \sum u^{\text{tors}}(\phi), \tag{1}$$

$$u^{\text{bend}}(\theta) = \tfrac{1}{2} k_\theta \left[\theta - \theta_0\right]^2, \tag{2}$$

$$u^{\text{tors}}(\phi) = c_0 + c_1 \cos(\phi) + c_2 \cos^2(\phi) + c_3 \cos^3(\phi), \tag{3}$$

where $u^{\text{bend}}(\theta)$ is the bond-bending energy and $u^{\text{tors}}(\phi)$ is the torsion energy. The second part of the potential, the external potential (u^{ext}), is used to bias the selection of a site from the set of trial sites. Note that this split into u^{internal} and u^{ext} is completely arbitrary and can be optimized for a particular application.

For a new configuration, a randomly chosen molecule is regrown segment by segment. If the entire molecule is being regrown than f trial sites for the first bead are placed at random positions in the simulation box [19]. The Rosenbluth weight of this segment is

$$w_1(n) = \sum_{j=1}^{f} \exp\left(-\beta u_{1j}^{\text{ext}}\right), \tag{4}$$

where $\beta = 1/(k_B T)$, and one trial site is selected with probability

$$p_{1i}^{\text{selecting}}(\mathbf{b}_i) = \frac{\exp\left(-\beta u_{1i}^{\text{ext}}\right)}{w_1(n)}. \tag{5}$$

For the other segments l of the molecule, k trial orientations $\mathbf{b}_i$ are generated according to the Boltzmann weight of the internal potential of that segment:

$$p_{li}^{\text{generating}}(\mathbf{b}_i)\,d\mathbf{b} = \frac{\exp\left(-\beta u_{li}^{\text{internal}}\right) d\mathbf{b}}{\int \exp\left(-\beta u_l^{\text{internal}}\right) d\mathbf{b}}. \tag{6}$$

Out of these k trial orientations one is chosen according to the Boltzmann weight of its external potential

$$P_{li}^{\text{selecting}}(\mathbf{b}_i) = \frac{\exp(-\beta u_{li}^{\text{ext}})}{\sum_{j=1}^{k} \exp(-\beta u_{lj}^{\text{ext}})}. \quad (7)$$

This procedure is repeated until the entire chain has been grown. The Rosenbluth weight $W(n)$ of the new configurations is defined as

$$W(n) = w_1(n) \prod_{l \neq 1} \left[\sum_{j=1}^{k} \exp(-\beta u_{lj}^{\text{ext}}) \right]. \quad (8)$$

The old configuration is retraced in a similar way, except that for each segment only $k - 1$ ($f - 1$ for the first bead) trial orientations are generated with a probability according to equation (6) (randomly for the first bead). The kth (fth) trial orientation is the old orientation. The Rosenbluth weight $W(o)$ of the old configuration is defined as

$$W(o) = w_1(o) \prod_{l \neq 1} \left[\sum_{j=1}^{k} \exp(-\beta u_{lj}^{\text{ext}}) \right], \quad (9)$$

where $w_1(o)$ is th Rosenbluth weight of the first segment of the old configuration. In order to satisfy the detailed balance condition, the new configuration is accepted with probability

$$\text{acc}(o \rightarrow n) = \min\left(1, \frac{W(n)}{W(o)}\right). \quad (10)$$

The CBMC algorithm greatly improves the conformational sampling for molecules with an articulated structure, and increases the efficiency of chain insertions (required for the calculation of chemical potentials, grand canonical, and Gibbs ensemble simulations) by several orders of magnitude.

There are two parts in the CBMC algorithm which dominate the computational expense.

1. *Generation of trial configurations.* This part becomes computationally expensive when the distribution $P_{li}^{\text{generating}}$ is narrow, which is the case for branched alkanes and/or at low temperatures. The difference in simulation time between a branched molecule and its linear isomer is due to differences in the widths of these distributions. In section 2, a more efficient algorithm will be presented to generate trial positions with a distribution according to equation (6).

2. *Calculation of external potential energies.* To calculate the external potential of a trial segment, a loop over all molecules has to be performed. In section 3 it will be explained how to reduce the cost of such a loop.

```
10  generate vector b_i random on a sphere [4, 20, 21]
    calculate u_i^bend(θ), u_i^tors(φ) and fac = exp[-β(u_i^bend(θ) + u_i^tors(v))]
    generate a random number rm between 0 and 1
    if(rm .gt. fac) goto 10
```

Figure 1. Conventional algorithm for the generation of trial orientations in accordance with equation (6).

2. Generation of trial orientations

The conventional algorithm for generating a trial orientation in accordance with equation (6) is to generate a random vector on a sphere, compute the internal potential energy, and accept or reject it using the acceptance/rejection scheme [4, 20, 21]. This algorithm is shown schematically in figure 1.

This algorithm is not very efficient because a large number of trial orientations with an unfavourable θ will be generated which are all rejected during the acceptance/rejection step. When the probability distribution according to equation (6) is very narrow, this becomes the rate limiting step in a simulation. A better algorithm will generate only values of θ which are close to θ_0 [22]. A simple way to do this is to split the internal potential into $u^{\text{bend}}(\theta)$ and $u^{\text{tors}}(\phi)$. If a single bond-bending angle is considered then the probability of generating an angle θ is

$$P(\theta)\,d\theta = \exp\left[-\beta u_i^{\text{bend}}(\theta)\right] d\mathbf{b}$$
$$= \exp\left[-\tfrac{1}{2}\beta k_\theta(\theta - \theta_0)^2\right] \sin(\theta)\,d\theta. \quad (11)$$

This is close to a Gaussian distribution with mean θ_0 and standard deviation $1/(k_\theta\beta)$. A random angle θ is generated from a Gaussian distribution and then corrected for the $\sin(\theta)$ term via the acceptance/rejection scheme [4, 20, 21]. The assumption that the bond-bending part is Gaussian is valid only when the probability density $P(\theta)$ is negligible near $\theta = 0$ and $\theta = \pi$. Tests show that this assumption is valid for alkanes, even at extremely high temperatures.

Once an angle is accepted the problem is reduced to that of finding a site on the cone defined by θ that satisfies the torsional angles, and any remaining bond-bending angles. The orientation of $\mathbf{b}_i$ is chosen randomly upon a cone with angle θ, $\sum(u^{\text{tors}} + u^{\text{bend}})$ is computed (excluding the bond-bending angle used to generate the cone), and the acceptance/rejection scheme is again used. This algorithm is shown schematically in figure 2.

Simulations of a single ideal chain (an ideal chain is defined as a chain with only internal intramolecular interactions) have been performed to compare the two algorithms for a range of molecular architectures. Table 1 shows the CPU time for generating 10^5 ideal chains at 400 K and at 800 K. It is clear that for both temperatures the new algorithm is much faster than the old one, and that the difference in CPU time between the genera-

```
10  generate an angle θ from a Gaussian distribution with mean θ0
    and standard deviation 1/√(kθβ) [4, 20, 21]
    generate a random number rm between 0 and 1
    if (rm.gt.sin(θ)) goto 10
    generate a random position on a cone with angle θ
    calculate fac = exp[βu_i^tors(φ)]
    generate a random number rm between 0 and 1
    if(rm.gt.fac) goto 10
```

Figure 2. Improved algorithm for the generation of trial orientations in accordance with equation (6).

Table 1. CPU time (arbitrary units) for generating 10^5 ideal chains. The forcefield is described in appendix A.

	Algorithm 1		Algorithm 2	
	400 K	800 K	400 K	800 K
Propane	2.9	2.1	1.0	1.0
Butane	18	7.9	4.1	2.8
2-Methylpropane	34	17	8.6	6.4
Hexane	48	20	11	6.4
2-Methylpentane	74	34	15	10
3-Methylpentane	156	52	30	15
Decane	107	44	24	14
2,4-Dimethyloctane	248	91	49	26
2,4,6-Trimethylheptane	338	117	67	34

tion of a linear and a branched alkane has decreased significantly. Using the new method, the difference in CPU time required to sample branched versus linear isomers still increases as the temperature is reduced, but it does so at a lower rate than the old method. In both cases the increase in CPU time is due to the probability distribution $P_{li}^{\text{generating}}$ narrowing as the temperature decreases.

However, the generation of trial orientations is only one part of the cost of the simulation, while the main cost of simulating non-ideal fluids is due to the loop over all other molecules needed in order to compute the Rosenbluth weights. Table 2 shows that the speedup gained by generating bond angles from a Gaussian distribution is only 1.2 for the simulation of linear octane, while for 3,4-dimethylhexane a factor of 1.8 is achieved (cf. simulations 1 and 2). Note that even with the new algorithm, heavily branched alkanes remain almost twice as expensive to simulate as their linear counterparts, and the CBMC moves have lower acceptance rates.

3. Dual cut-off configurational-bias Monte Carlo

Consider two molecules, A and B, each of which has a maximum distance d_a and d_b from its centre of mass to any of its interaction sites. Given the cutoff distance for the potential is r_{cut}, then when the centres of mass (COM) of the two molecules are separated by more than $d_a + d_b + r_{cut}$, the molecules cannot have any direct interaction. Thus, the distances between all interaction sites do not have to be computed [23]. The maximum distance from the centre of mass of a molecule to any of its sites has to be updated only after a successful change of its conformation. In table 2 it is shown that for systems of 144 octanes, implementation of this COM cutoff decreases the CPU time by about 20% (cf. simulations 1 and 3). Note that this speedup increases greatly for large system sizes. The speedup for a system of 1152 n-octanes is a factor of 5.

In a CBMC move, the COM cutoff can be used also when the Boltzmann factors of the k trial orientations for segment i have to be calculated. A list is made of all molecules j which have a distance to segment $i-1$ that is less than $l + r_{cut} + d_j$, where l is the bond length between segments i and $i-1$. The external energy of each bead of each molecule on this list with the k trial sites must be computed in order to calculate the Boltzmann factor and Rosenbluth weight.

The disadvantage of the use of this list is that the number of interactions that need to be calculated in such a list can be large. To reduce the size of the list, we split the external potential into two parts

Table 2. CPU time divided by the number of accepted CBMC moves (η) and the percentage accepted CBMC moves for the NpT simulation of isomers of n-octane with different algorithms (T = 450 K). The forcefield and details of the simulation are described in the appendixes.

	Simulation									
	1		2		3		4		5	
Improved orientation algorithm	no		yes		no		yes		yes	
COM	no		no		yes		yes		yes	
r_{cut}*	—		—		r_{cut}		r_{cut}		optimal	
	η	%	η	%	η	%	η	%	η	%
n-Octane	0.58	6.5	0.46	7.1	0.48	6.7	0.36	7.0	0.18	6.7
3-Methylheptane	0.89	5.3	0.60	5.8	0.71	5.6	0.39	7.0	0.22	6.5
3,4-Dimethylhexane	1.46	4.7	0.81	5.0	1.25	4.9	0.63	5.1	0.40	4.8

Table 3. CPU time divided by the number of accepted DC-CBMC moves (η) and the percentage accepted DC-CBMC moves as a function of r_{cut}* for the NpT simulation of isomers of n-octane (T = 450 K). The improved algorithm for the generation of trial orientatons was used. The forcefield and details of the simulation are described in the appendixes.

	n-Octane		3-Methylheptane		3,4-Dimethylhexane	
r_{cut}* /Å	η	%	η	%	η	%
2.5	2.89	0.3	1.80	0.6	1.93	0.8
3.0	0.87	1.3	1.06	1.2	1.19	1.4
4.0	0.18	6.7	0.22	6.5	0.40	4.8
5.0	0.18	7.0	0.24	6.2	0.41	4.8
6.0	0.22	6.2	0.29	5.4	0.40	5.1
10.0	0.29	6.6	0.37	5.6	0.50	5.2
14.0	0.36	7.0	0.39	7.0	0.63	5.1

$$u^{ext} = \bar{u}^{ext} + \delta u^{ext}, \quad (12)$$

where $\bar{u}^{ext}$ is a potential that is less expensive to calculate than u^{ext}, and δu^{ext} the difference between the two potentials. A useful choice for $\bar{u}^{ext}$ is the potential described by a shorter cutoff radius (r_{cut}*)

$$u^{ext}(r) = \bar{u}^{ext}(r < r_{cut}*) + \delta u^{ext}(r_{cut}* < r < r_{cut}), \quad (13)$$

where $\bar{u}^{ext}(r < r_{cut}*)$ consists only of interactions within a distance r_{cut}*. We can use this additional cutoff radius to generate a list that is much shorter than the one used for r_{cut}. Dual cutoff configurational-bias Monte Carlo (DC-CBMC) uses the potential $\bar{u}^{ext}(r < r_{cut}*)$ for generation of the chain, and thus calculates the Rosenbluth weight faster than CBMC. However, this would lead to an incorrect distribution if we used the conventional acceptance rule (equation (10)). The correct distribution is recovered by using

$$acc(o \rightarrow n) = \min\left(1, \frac{\bar{W}(n)}{\bar{W}(o)} \exp\left(-\beta\left[\delta u^{ext}(n) - \delta u^{ext}(o)\right]\right)\right) \quad (14)$$

as the acceptance rule, where $\bar{W}(n)$ and $\bar{W}(o)$ are the Rosenbluth weights calculated using $\bar{u}^{ext}$.† The proof is given in appendix D. Thus, a DC-CBMC regrowth requires only the calculation of the full potential for the final configuration, and not for all of the trial orientations. (The potential of the selected configuration can be calculated also using a pair list. This pair list can be made readily when the pair list for the calculation of the Rosenbluth weight is made.)

In the appendixes we prove that the distribution generated by DC-CBMC is equal to the distribution generated by ordinary CBMC. However, DC-CBMC does generate a different Markov chain. Therefore it is not obvious that DC-CBMC is more efficient since, for example, the percentage of accepted moves may drop significantly if the additional cutoff radius is decreased. It is reasonable to assume that a new configuration generated by DC-CBMC is equally as independent as a new configuration in the old scheme. Hence, a good comparison of the efficiency is the CPU time divided by the number of accepted moves. Our test simulations show that as long as r_{cut}* is larger than the Lennard-Jones size parameter σ, the percentage of accepted moves does not decrease significantly (see table 3). We define the computational cost of the simulation (η) as the CPU time divided by the number of accepted DC-CBMC moves.‡ The efficiency of the simulation is the inverse of the computational cost. In figure 3 and table 3 the computational cost is shown as a function of the additional cutoff radius. For r_{cut}* = r_{cut} we recover the original CBMC algorithm, and r_{cut}* = 0 corresponds to a random insertion of the chain. As r_{cut}* is decreased the efficiency increases almost twofold until, for very low r_{cut}*, the efficiency decreases dramatically. The optimal r_{cut}* is that for which the efficiency is highest. We found that for all octanes this optimal radius is approximately 4.0 Å, which is roughly the size of a pseudo-atom. This suggests that the excluded volume is the dominant term for the CBMC generation of a chain.

To investigate the effect of temperature and density on the efficiency of DC-CBMC, we performed NVT simulations at several temperatures and densities. In figure 4, the computational cost versus r_{cut}* is shown for the NVT simulation of n-octane at three tempera-

† Note that this division of the external energy can be done in any consistent fashion, and is typically used in conventional CBMC to account for Lennard-Jones tail corrections. This may also be used for Ewald corrections in charged systems.

‡ In order to obtain information on the computational cost of a typical simulation, in addition to CBMC moves other types of moves are also performed. See the Appendix for details.

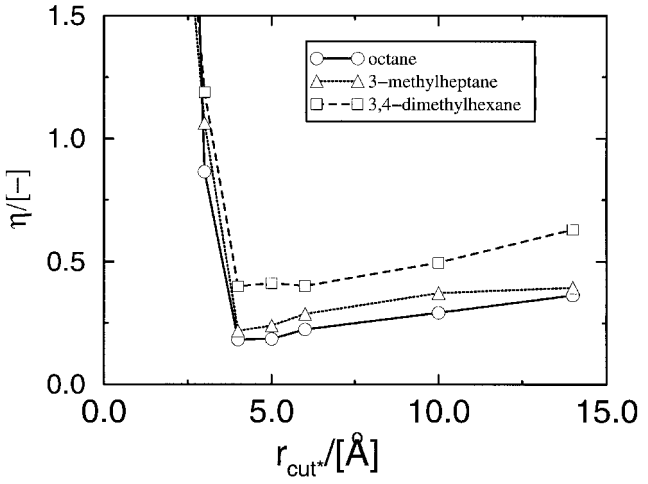

Figure 3. CPU time divided by the number of accepted DC-CBMC moves (η) as a function of r_{cut}* for the NpT simulation of isomers of n-octane (T = 450 K). The improved algorithm for the generation of trial orientations was used. The forcefield and details of the simulation are described in the appendixes.

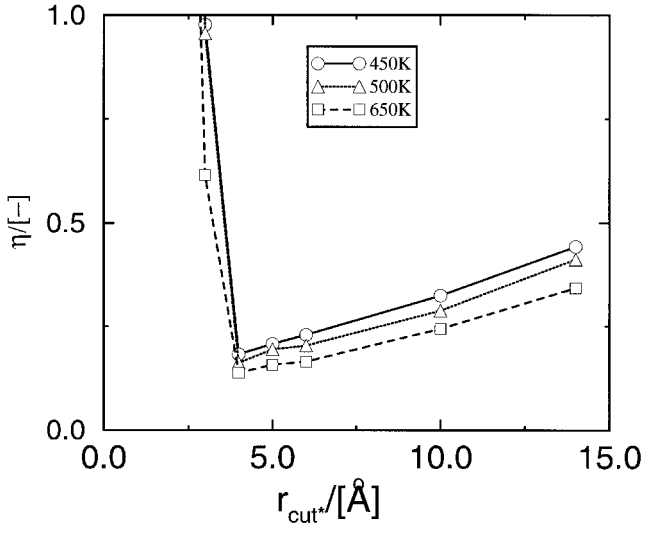

Figure 4. CPU time divided by the number of accepted DC-CBMC moves (η) as a function of r_{cut}* for the NVT simulation of n-octane at different temperatures (ρ = 0.61 g ml^{-1}). The improved algorithm for the generation of trial orientations was used. The forcefield and details of the simulation are described in the appendixes.

tures. The optimal r_{cut}* and the gain in efficiency do not change significantly with temperature.

The influence of the system density on r_{cut}* is plotted in figure 5. Again, the optimal r_{cut}* is constant. Both DC-CBMC and CBMC are computationally more expensive at higher densities because the insertion of a trial segment becomes more difficult. Figure 5 suggests

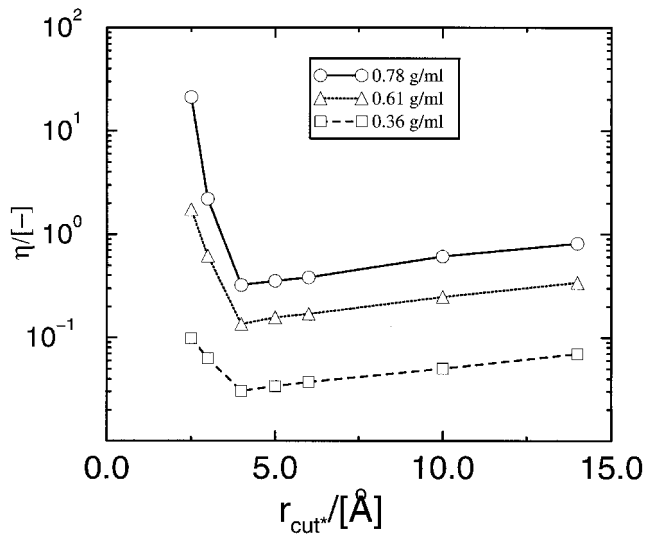

Figure 5. CPU time divided by the number of accepted DC-CBMC moves (η) as a function of r_{cut}* for the NVT simulation of n-octane at T = 650 K and at different densities. The improved algorithm for the generation of trial orientations was used. The forcefield and details of the simulation are described in the appendixes.

that the plot of the logarithm of the efficiency versus r_{cut}* consists of two linear parts, so the optimal r_{cut}* can be calculated easily.

4. Conclusion

It is demonstrated that for a system of 144 octanes the use of DC-CBMC and generating angles from a Gaussian type distribution is three to four times faster than conventional CBMC (see table 2). The speedup originating from the use of a centre-of-mass based cutoff increases with increasing system size. The simulation of branched alkanes remains computationally more expensive than the simulation of linear alkanes, but the difference in CPU time per accepted move is reduced. The optimal DC-CBMC additional cutoff radius r_{cut}* suggests that the excluded volume is the most important term for biasing the selection of CBMC growth.

Financial supported provided to T.J.H.V. from SON (Stichting Scheikundig Onderzoek Nederland), to B.S. via the Graduate School on Process Technology (OSPT), to M.G.M. via a Department of Energy Computational Science Graduate Fellowship and to J.I.S. from the Petroleum Research Fund, administered by the American Chemical Society, and a Camille and Henry Dreyfus New Faculty Award are gratefully acknowledged. A large part of the computer resources was generously provided by the SARA (Stichting Aca-

demisch Rekencentrum Amsterdam) and the Minnesota Supercomputer Institute.

Appendix A
Force field

At the time of this work the new transferable potentials for phase equilibria (TraPPE) forcefield was not yet complete for the alkanes. Thus, we have used a hybrid forcefield which combines the TraPPE methyl group [23], the Siepmann–Karaborni–Smit methylene group [6, 7], and a preliminary version of the TraPPE methine group. These Lennard-Jones parameters were fitted to give the vapour–liquid equilibria of a range of alkanes.

1. The united-atom representation is used, i.e., CH_3, CH_2 and CH are each considered to be one pseudo-atom.
2. The Lennard-Jones parameters are:

$$CH_3 : \varepsilon/k_B = 98\,K, \sigma = 3.75\,\text{Å},$$

$$CH_2 : \varepsilon/k_B = 47\,K, \sigma = 3.93\,\text{Å},$$

$$CH : \varepsilon/k_B = 10\,K, \sigma = 4.6\,\text{Å}.$$

Interactions between different united atoms are calculated with

$$\varepsilon_{ij} = (\varepsilon_{ii}\varepsilon_{jj})^{1/2}, \qquad (A\,1)$$

$$\sigma_{ij} = \frac{\sigma_{ii} + \sigma_{jj}}{2}. \qquad (A\,2)$$

3. Intermolecular interactions are truncated at 14 Å and the usual tail corrections are added [4, 20]. Intramolecular Lennard-Jones interactions are considered for sites separated by more than 3 bonds.
4. Bond lengths are fixed at 1.54 Å
5. Bond angles are 114° for CH_2 centred angles and 112° for CH centred angles. The bond-bending energy is described by equation (2) with $k_\theta/k_B = 62\,500\,K\ \text{rad}^{-2}$ [17].
6. The torsion energy is described by equation (3). The constants are taken from [18, 24].

Appendix B
NpT simulations

Systems of $N = 144$ molecules were simulated at a temperature of 450 K and a pressure of 100 kPa for 2500 cycles (one cycle is equal to N Monte Carlo moves). Trial moves were selected randomly with a fixed probability P from four possible moves: CBMC (or DC-CBMC) regrowth of the chain starting with the second segment† ($P = 0.79, k = 6$), isotropic volume change ($P = 0.01$), translation of a randomly chosen molecule ($P = 0.1$) and rotation of a randomly chosen molecule about its centre of mass ($P = 0.1$). All maximum displacements were adjusted to give a 50% acceptance rate. The initial configurations were equilibrated for 10^4 cycles.

† In order to get good statistics, only one type of CBMC move was performed.

Appendix C
NVT simulations

Systems of $N = 144$ octane molecules were simulated at a constant temperature and volume for 2500 cycles. Trial moves were selected randomly with a fixed probability P from three possible moves: CBMC (or DC-CBMC) regrowth of the chain starting with the second segment ($P = 0.80, k = 6$), translation of a randomly chosen molecule ($P = 0.1$) and rotation of a randomly chosen molecule about its centre of mass ($P = 0.1$). All maximum displacements were adjusted to give a 50% acceptance rate. The initial configurations were equilibrated for 10^4 cycles.

Appendix D
Proof of equation (14)

The flow of configurations from state o to state n can be written as

$$K(o \to n) = N(o)p(o \to n)\,\text{acc}(o \to n) \qquad (D\,1)$$

where $N(o)$ is the probability of the system residing in state o, $p(o \to n)$ is the probability that we attempt to move the system from state o to state n, and $\text{acc}(o \to n)$ is the probability of accepting this move. Detailed balance requires that the flow of configurations from o to n is equal to the flow of configurations from n to o. From this we can derive the following expression [7]

$$\frac{\text{acc}(o \to n)}{\text{acc}(n \to o)} = \frac{\exp\left[-\beta u^{\text{ext}}(n)\right]}{\exp\left[-\beta u^{\text{ext}}(o)\right]} \frac{\exp\left[-\beta \bar{u}^{\text{ext}}(o)\right]}{\bar{W}(o)} \times \frac{\bar{W}(n)}{\exp\left[-\beta \bar{u}^{\text{ext}}(n)\right]} = \frac{\bar{W}(n)}{\bar{W}(o)} \exp\left(-\beta\left[\delta u^{\text{ext}}(n) - \delta u^{\text{ext}}(o)\right]\right). \qquad (D\,2)$$

It is straightforward to show that equation (14) obeys this equation.

References

[1] Siepmann, J. I., and Frenkel, D., 1992, *Molec. Phys.*, **75**, 59.

[2] Frenkel, D., Mooij, G. C. A. M., and Smit, B., 1992, *J. Phys. condensed Matter*, **4**, 3053.

[3] de Pablo, J. J., Laso, M., and Suter, U. W., 1992, *J. chem. Phys.*, **96**, 6157.

[4] Frenkel, D., and Smit, B., 1996, *Understanding Molecular Simulations: From Algorithms to Applications* (San Diego: Academic Press).

[5] Siepmann, J. I., 1993, *Computer Simulation of Biomolecular Systems: Theoretical and Experimental Applications*, edited by W. F. van Gunsteren, P. K. Weiner and A. J. Wilkinson (Leiden: Escom), p. 249.
[6] Siepmann, J. I., Karaborni, S., and Smit, B., 1993, *Nature*, **365,** 330.
[7] Smit, B., Karaborni, S., and Siepmann, J. I., 1995, *J. chem. Phys.*, **102,** 2126.
[8] Martin, M. G., and Siepmann, J. I., 1997, *J. Amer. chem. Soc.*, **119,** 8921.
[9] Siepmann, J. I., Martin, M. G., Mundy, C. J., and Klein, M. L., 1997, *Molec. Phys.*, **90,** 687.
[10] Zhuravlev, N. D., and Siepmann, I. J., 1997, *Fluid Phase Equilibria*, **134,** 55.
[11] Cui, S. T., Cummings, P. T., and Cochran, H. D., 1998, *Fluid Phase Equilibria*, **141,** 45.
[12] Smit, B., and Maesen, T. L. M., 1995, *Nature*, **374,** 42.
[13] Smit, B., 1995, *J. phys. Chem.*, **99,** 5597.
[14] Bates, S. P., van Well, W. J. M., van Santen, R. A., and Smit, B., 1996, *J. phys. Chem.*, **100,** 17 573.
[15] Maginn, E. J., Bell, A. T., and Theodorou, D. N., 1995, *J. phys. Chem.*, **99,** 2057.
[16] Bandyopadhyay, S., and Yashonath, S., 1997, *J. phys. Chem.* B, **101,** 5675.
[17] Van der Ploeg, P., and Berendsen, H. J. C., 1982, *J. chem. Phys.*, **76,** 3271.
[18] Jorgensen, W. L., Madura, J. D., and Swenson, C. J., 1984, *J. Amer. chem. Soc.*, **106,** 6638.
[19] Esselink, K., Loyens, L. D. J. C., and Smit, B., 1995, *Phys. Rev.* E, **51,** 1560.
[20] Allen, M. P., and Tildesley, D. J., 1987, *Computer Simulation of Liquids* (Oxford: Clarendon Press).
[21] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., 1986, *Numerical Recipes: The Art of Scientific Computing* (Cambridge University Press).
[22] Spyriouni, T., Economou, I. G., and Theodorou, D. N., 1997, *Macromolecules*, **30,** 4744.
[23] Martin, M. G., and Siepmann, J. I., 1998, *J. phys. Chem.* B, **102,** 2569.
[24] Wang, Y., Hill, K., and Harris, J. G., 1994, *J. phys. Chem.*, **100,** 3276.