TWENTANGLEMENT
user manual

J. T. Padding, P. Kindt, A. van den Noort, W. J. Briels

September, 2003

Method of operation

Without TWENTANGLEMENT

1. If not already compiled, pick a set of compiler options and compile (make). In the Makefile, the compiler options appropriate for your machine should be chosen. Compilation is normally only needed once (per machine type), because almost all array sizes are variable (see **??** for details on how to compile).

2. Enter the structure data in field.dat. In the examples, several field-generators are included for different types of structures.

3. Enter the force field parameters in parameter.dat.

4. Edit control.dat for the desired simulation run.

5. Run INITIATE. The program will generate Boltzmann distributed configurations using the intramolecular potential energy. The molecules are oriented and placed at random in the simulation box. Initial velocities are taken from a Maxwell distribution. The random placement of the molecules causes more than physical overlap. Equilibration with smaller time steps or softer potentials might be necessary.

6. If wanted, equilibrate the system: run KRAK. The default output is sent to the screen. Make sure that correlations are not calculated (0 for time write intervals).

7. Run SETZERO. The program will ask which output has to be generated and if previous correlation files have to be reset or not.

8. Run KRAK

With TWENTANGLEMENT

1. If not already compiled, pick a set of compiler options and compile (make). In the Makefile, the compiler options appropriate for your machine should be chosen. Compilation is normally only needed once (per machine type), because almost all array sizes are variable (see **??** for details on how to compile).

2. Enter the structure data in field.dat. In the examples, several field-generators are included for different types of structures.

3. Enter the force field parameters in parameter.dat.

4. Edit the control.dat for the desired simulation run. Make sure that the TWENTANGLEMENT algorithm is turned off.

5. Run INITIATE. The program will generate Boltzmann distributed configurations using the intramolecular potential energy. The molecules are oriented and placed at random in the simulation box. Initial velocities are taken from a Maxwell distribution. The random placement of the molecules causes more than physical overlap. Equilibration with smaller time steps or softer potentials might be necessary.

6. Equilibrate the system: run KRAK. The default output is sent to the screen. Make sure that correlations are not calculated (0 for time write intervals).

7. Turn on the TWENTANGLEMENT algorithm in control.dat

8. Run SETZERO and ask for TWENTANGLEMENT output.

9. Run KRAK to equilibrate with the TWENTANGLEMENT algorithm.

10. Edit the desired correlations in control.dat.

11. Run SETZERO.

12. Run KRAK.

Introduction

TWENTANGLEMENT is a simulation package based on various types of mesoscale simulation methods. The package contains several programs and the TWENTANGLEMENT uncrossability algorithm . The main executable is KRAK , a molecular dynamics program. KRAK is an extended version of JOMD [1] and is capable of handling different types of beads and structures. Both programs are inspired by Allen and Tildesley's book 'Computer Simulations of Liquids' [**?**].

KRAK includes the TWENTANGLEMENT uncrossability algorithm designed and implemented by J.T. Padding. The main reason for developing KRAK was the need for a program that can handle the TWENTANGLEMENT uncrossability algorithm in a more general way, because the original JOMD [2] program could only handle linear homopolymer melts.

The source code of KRAK (built upon JOMD) and TWENTANGLEMENT is written in a period of several years and although some effort is spent to make the code more clear for a new user, we apologize for remainders of old code, unclear variables etc. If you have any questions about the code, you are welcome to contact the authors.

The program source code is written in Fortran 90. It is compiled and tested on an SGI IRIX machine (Origin 2000 with R10000 processors), a Compaq Alpha station (DS20E with 21264 processors) and a PC (Pentium 4 2.4 GHz processor). For the moment is does not support parallel computing.

KRAK and TWENTANGLEMENT are made available under the following conditions:

1. The program shall be used for scientific purposes only, excluding industrial or commercial purposes.

2. Proper acknowledgements shall be made to the authors of the programs in publications resulting from the use of these programs.

3. The programs shall not be made available to users outside the recipient's laboratory, unless written consent is obtained.

The newest version of this manual can be downloaded from:

http://tnweb.tn.utwente.nl/cdr/Staff/Albert/

Johan Padding, Peter Kindt, Albert van den Noort and Wim Briels [1]

[1] contact:

- jtp26@cam.ac.uk
- p.kindt@utwente.nl
- a.vandennoort@utwente.nl
- w.j.briels@utwente.nl

Contents

1

Theory: Equation of motion and force fields

KRAK is capable of handling Langevin Dynamics with background friction (BF) or pairwise friction (PF). A short introduction into these simulation methods and the parameters needed to start simulating is given below.

1.1 BF

The equations of motion of BF are:

$$M_i \frac{d^2\vec{R}_i}{dt^2} = -\vec{\nabla}_i \psi - M_i \xi_i \frac{d\vec{R}_i}{dt} + \vec{F}_i^R \tag{1.1}$$

The potential energy, ψ , in the first term on the right hand side, consists of four parts:

$$\begin{aligned}
\psi &= \phi^{nb} + \phi^{rep} + \phi^{att} + \phi^{\vartheta} \\
\phi^{nb} &= \sum_{i<j} c_0 e^{-(R_{ij}/b_0)^2} \\
\phi^{rep} &= \sum_{bonded} c_1 e^{-(R_{i,i+1}/b_1)^2} + c_2 e^{-(R_{i,i+1}/b_2)^2} \\
\phi^{\vartheta} &= \sum_{triplets\, i} c_4 (1 - \cos\vartheta_i)^{\nu} \\
\phi^{att} &= c_3 [L_{i,i+1}(\vec{R}_n, \vec{R}_{ent})]^{\mu}
\end{aligned} \tag{1.2}$$

The non-bonded interaction, ϕ^{nb}, is not taken into account between directly bonded particles. The bonded interaction is split into a repulsive part, ϕ^{rep}, between two consecutive beads and an attractive part, ϕ^{att}, along the path between two consecutive beads and the entanglements along that path. The entanglement positions are determined in such a way that ϕ^{att} is always at a minimum with given positions of the beads, $\vec{R}_n$.

The angular term in Eq. (1.2), ϕ^{ϑ}, takes only particles into account, which are two bonds separated in the same chain. So, at branch-points of branched molecules, no angle-potential is present between two particles from a different chain (although they can physically be just two bonds separated).

The second term on the right hand side of Eq. (1.1), is the background friction term. For every particle type a specific value for ζ can be entered. The random force (third term) is related to the friction by the fluctuation dissipation theorem:

$$\left\langle F^R_{i,\alpha}(t) F^R_{j,\beta}(0) \right\rangle = 2 k_B T M_i \zeta_i \delta_{ij} \delta_{\alpha\beta} \delta(t) \tag{1.3}$$

The equations of motion are solved with pre-integration in the way described by Allen [3].

1.2 PF

The PF scheme is based on Langevin Dynamics and the friction and random forces of the Dissipative Particle Dynamics scheme of Koelman and Hoogerbrugge [4]. The leap-frog integration scheme is used:

$$\begin{aligned} \vec{R}(t + \Delta t) &= \vec{R}(t) + \vec{v}(t + \Delta t/2) dt \\ \vec{v}(t + \Delta t/2) &= \vec{v}(t - \Delta t/2) + \frac{\vec{F}(t) \Delta t}{M} \end{aligned} \tag{1.4}$$

The force in the PF scheme consists of three pair-wise parts, a conservative part, a dissipative part and a random part. The conservative force both non-bonded and bonded are handled in the same way as in BF.

$$\begin{aligned} \vec{F} &= \vec{F}^C + \vec{F}^D + \vec{F}^R \\ \vec{F}_{ij}^D &= -\gamma_{kl} w^D(r_{ij})(\hat{r}_{ij} \cdot \vec{v}_{ij})\hat{r}_{ij} \\ \vec{F}_{ij}^R &= -\sigma_{kl} w^R \theta_{ij} \hat{r}_{ij} \end{aligned} \tag{1.5}$$

The weight functions are correlated and θ_{ij} is a randomly fluctuating variable with Gaussian statistics, zero mean and unit variance.

$$\begin{aligned} w^D(r) &= \left[w^R(r)\right]^2 \\ w^R(r) &= \begin{cases} (1 - \frac{r}{r_{cut}}) & : \quad r < r_{cut} \\ 0 & : \quad r \geq r_{cut} \end{cases} \\ \sigma_{kl} &= \sqrt{2 k_B T \gamma_{kl}} \end{aligned} \tag{1.6}$$

Mark the fact that the noise strength σ_{kl} is allowed to depend on the particle types (kl), in contrast to the fixed values in most other simulations. If other conservative forces or weight functions than those implemented are needed, it will only require simple modifications in *force.f* in the subroutines *forcePF* and *sforcePF*.

2

TWENTANGLEMENT
Uncrossability algorithm

2.1 Introduction

An important feature of a melt of long polymers is that the bonds of the chains cannot cross each other. This seemingly simple fact has a great impact on the long time dynamics and rheology of the material. In this chapter we will describe an algorithm that explicitly detects and prevents bond crossings in mesoscopic simulations of polymers. The central idea is to consider the bonds as slippery elastic bands which can get entangled. At first, the method was applied to a simulation of a coarse-grained melt of $C_{120}H_{242}$, in which each chain is represented by six blobs [1]. The long time dynamics and zero-shear rate rheology were investigated and the relative importance of uncrossability and chain stiffness were established. As a result of the uncrossability of the chains we observed a subdiffusive exponent in the mean square displacement of the chains, a stretching of the exponential decay of the Rouse mode relaxations, an increase of relaxation times associated with large scales, and a slowing down of the relaxation of the dynamic structure factor. These results were in agreement with the results from microscopic molecular dynamics simulations. Finally, an increased viscosity as compared to the Rouse model is observed, which is attributed to slowly decaying interchain stress components.

An important consequence of averaging out the bath variables (coarse graining) is that the resulting bonded and non-bonded interactions become softer and broaden their range. It becomes likely that two bonds will cross. This is an unphysical process, which will make the model loose all its dynamic properties characteristic for polymer melts. For this reason we have developed

an entanglement algorithm called TWENTANGLEMENT (Twente University entanglement).

One can think of many ways to implement an uncrossability constraint. Our method is based on considering the bonds as elastic bands between the bonded particles. As soon as two of these elastic bands make contact, an 'entanglement' is created which prevents the elastic bands from crossing. This is depicted in Fig. 2.1.

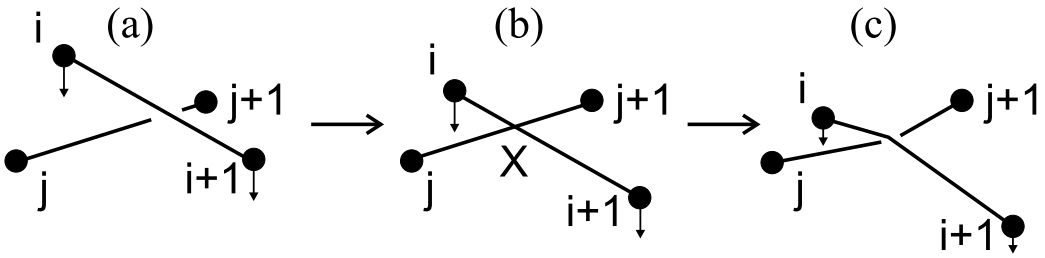

Figure 2.1: *The creation of an 'entanglement': (a) two pairs of bonded blobs are closing in on each other. (b) At a certain moment their bonds will touch. (c) An entanglement is created at the crossing point X, after which the bonds are viewed as slippery elastic bands. The elasticity will slow down the relative speed of the bonds.*

To take away any confusion: in the algorithm 'entanglements' are defined as the objects which prevent the crossing of chains. Only a few of these are expected to be entanglements in the usual sense of long-lasting obstacles, slowing down the chain movement. For instance, the $C_{120}H_{242}$ chain is generally considered not to be entangled, yet many 'entanglements' occur in the simulation. For the implementation of the algorithm, the following considerations have been taken into account: (i) the entanglement algorithm should be a simple generalization of the force routine, yet capture the essence of what an entanglement is, i.e. (ii) an entanglement must prevent the crossing of two bonds, (iii) an entanglement should be able to dynamically slide along the backbone of the chain (i.e. not be a permanent crosslink), and (iv) an entanglement should be able to disentangle if the topology of the chain demands so.

2.2 Detection of bond crossing

The detection of crossings between bonds is achieved by means of a simple geometric argument, details of which will be given in the next section. Once an imminent bond crossing is detected, an entanglement point $\vec{X}$ is defined at the crossing site, as in Fig. 2.1(b). As the blobs continue to move, the entanglement point $\vec{X}$ shifts, such that it will push both bonds back to their respective sides. This is accomplished by changing the attractive potential φ^{att} between bonded blobs by replacing the blob distance $R_{i,i+1}$ by the *path length* $L_{i,i+1}$ from one blob i to the next *via* the entanglement:

$$L_{i,i+1} = \left|\vec{R}_i - \vec{X}\right| + \left|\vec{X} - \vec{R}_{i+1}\right|, \tag{2.1}$$

$$\varphi^{\mathrm{att}}\left(L_{i,i+1}\right) = c_3 \left(L_{i,i+1}\right)^{\mu}. \tag{2.2}$$

The entanglement position $\vec{X}$ is fixed by requiring that the total attractive potential energy of the entangled bonds $\varphi^{\mathrm{att}}\left(L_{i,i+1}\right) + \varphi^{\mathrm{att}}\left(L_{j,j+1}\right)$ is at its minimum. This is equivalent to requiring equilibrium of forces at the entanglement. In a sense, the original bonds are replaced by slippery elastic bands which go via the entanglements. The finite extensibility of the bands prevents entangled chains from crossing each other. The expression for the path length that is given here is only valid in case of just one entanglement between two pairs of bonded blobs. The algorithm allows for any number of entanglements between pairs of bonded blobs. To this end the path length concept has been trivially modified.

The entanglement algorithm described below can be placed in the force routine of any standard molecular or stochastic dynamics program. A typical update in such a program consists of evaluating the forces that act on the particles, using these to calculate the accelerations of the particles and subsequently updating the particle velocities and positions. The entanglement algorithm in the force routine consists of the following three parts:

1. Given the new blob positions and the order of blobs and entanglements in the chains, move the entanglements to their new positions and calculate the forces that act on the blobs.

2. Detect new entanglements and disentanglements caused by the movements of the blobs and the entanglements.

3. If possible, let entanglements slip across blobs or each other (non-trivial moves).

The details of each of these parts will be described in the following subsections.

2.2.1 Moving entanglements

Suppose entanglements already exist. In the mesoscopic model, the entanglements have no volume and are fully characterized by their positions $\vec{X}_k$. As was already mentioned in the previous section, the attractive potential φ^{att} is redefined to be a function of the path length $L_{i,i+1}$. In case the bond runs from blob i, via p consecutive entanglements, to blob $i+1$ (see Fig. 2.2) the path length is defined as:

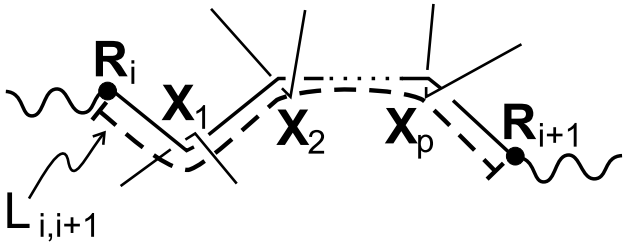

Figure 2.2: *Definition of the pathlength $L_{i,i+1}$ between bonded blobs at $\vec{R}_i$ and $\vec{R}_{i+1}$, Eq. (2.3).*

$$L_{i,i+1} = \left|\vec{R}_i - \vec{X}_1\right| + \left|\vec{X}_1 - \vec{X}_2\right| + \ldots + \left|\vec{X}_p - \vec{R}_{i+1}\right| . \tag{2.3}$$

Since each blob represents a large collection of monomer units, the heavy backbone of the mesoscopic chain will in general move very sluggishly. This is in contrast to an entanglement, which at the atomic level includes only a few monomer units. Consequently, the timescale with which the entanglement position adjusts itself is much shorter than the timescale with which the polymer backbone is moving. Effectively, on the coarse grained time scale, there will be an equilibrium of forces at the entanglements. Such an equilibrium of forces in a system with n blobs and p entanglements is achieved by the following minimization:

$$\Phi^{\text{att}}\left(\vec{R}^n\right) = \min_{\vec{X}^p} \sum_i \varphi^{\text{att}}\left(L_{i,i+1}\left(\vec{R}^n, \vec{X}^p\right)\right), \tag{2.4}$$

i.e. the entanglement positions are determined by the requirement that the total attractive energy be at its minimum (the blobs are kept at their respective positions). This minimum energy is the contribution $\Phi^{\rm att}$ of the attractive part to the total potential energy Φ of the system. The other contributions, ϕ^{rep}, ϕ^{nb} and ,ϕ^{ϑ} are calculated in the usual way from the blob positions, and together constitute the remainder Φ^r of the potential energy, i.e. $\Phi = \Phi^{\rm att} + \Phi^r$. At each time step the dynamics program will need the forces on the blobs in order to update their velocities and positions. The minimization in Eq. (2.4) does not complicate the evaluation of the force on a blob i since $\left(\partial\Phi^{\rm att}/\partial\vec{X}_k\right) = 0$ at the minimum:

$$\begin{aligned} \vec{F}_i &= -\frac{\partial\Phi^{\rm att}}{\partial\vec{R}_i} - \sum_k \left(\frac{\partial\Phi^{\rm att}}{\partial\vec{X}_k}\right)\cdot\left(\frac{\partial\vec{X}_k}{\partial\vec{R}_i}\right) - \frac{\partial\Phi^r}{\partial\vec{R}_i} \\ &= -\frac{\partial}{\partial\vec{R}_i}\left(\Phi^{\rm att} + \Phi^r\right). \end{aligned} \tag{2.5}$$

From the definition of the path length, Eq. (2.3), it is clear that the attractive force $\vec{f}_i^{\rm att}$ on blob i due to the elastic band between blobs i and $i+1$ is always directed along $\vec{R}_i - \vec{X}_1$:

$$\begin{aligned} \vec{f}_i^{\rm att} &= -\frac{\partial\varphi^{\rm att}\left(L_{i,i+1}\right)}{\partial\vec{R}_i} = -\frac{\partial\varphi^{\rm att}\left(L_{i,i+1}\right)}{\partial L_{i,i+1}}\frac{\partial L_{i,i+1}}{\partial\vec{R}_i} \\ &= -c_3\mu\left(L_{i,i+1}\right)^{\mu-1}\frac{\vec{R}_i - \vec{X}_1}{\left|\vec{R}_i - \vec{X}_1\right|}, \end{aligned} \tag{2.6}$$

and, correspondingly, the force due to this elastic band on blob $i+1$ is always directed along $\vec{R}_{i+1} - \vec{X}_p$, see Fig. 2.2.

It is important to stress that we have constructed a system in which the total energy is conserved. No work is done on the entanglements because the net force is always zero. Also, the entanglements bear no mass and hence have no kinetic energy. However, the Hamiltonian is history dependent. To calculate the forces at time t we need information about the number of entanglements and their positions along the backbones of the chains, which is a result of events at times $t' < t$. In this respect the ensemble is not canonical.

The reader may wonder why the repulsive term of the bonded blob interaction, was not redefined to be a function of the path length, as was realized for the attractive term. The main reason is that the equilibrium position of the entanglement would not be uniquely defined if lower repulsive energies

were associated with increased path lengths. Local minima would emerge in which the minimization procedure could be trapped. Moreover, direct repulsion between the centers of bonded blobs is needed to keep the angular forces from growing too large. A potential defined in terms of angle θ results in a torque between two bonds. If either one of the bond lengths tends to zero, the force on the blob will go infinite, which is highly undesirable.

We end this subsection by mentioning a subtlety involved in the minimization of the attractive energy. Most effective minimization procedures require evaluations of both functions and gradients, i.e. forces. The total force on an entanglement is the sum of the forces along its four arms. Each of these forces is minus the force on its neighbor along the relevant arm. Now, for example, while moving $\vec{X}$ in Fig. 2.3 at constant blob positions $\vec{R}$, the force on $\vec{X}$ has a singularity at $\vec{R}_i$, as elucidated in the figure.

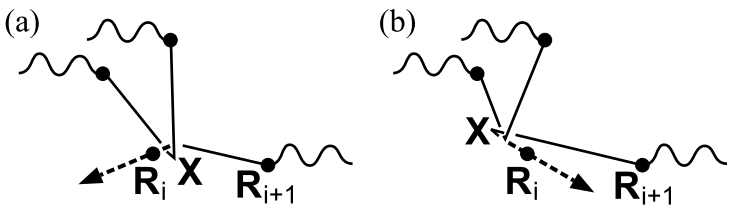

Figure 2.3: *The gradient of the attractive energy has a singularity at $\vec{R}_i$. Near this point the magnitude of the attractive force on the entanglement $\vec{X}$ stays approximately constant, while its direction is given by the unit vector from $\vec{X}$ to $\vec{R}_i$. This direction changes promptly near $\vec{R}_i$ (dashed arrow).*

Notice that the order of objects is not changed in this step of the entanglement algorithm; only in the last step will we check for a possible slip across the blob. Now the above singularity will cause the minimization to converge erroneously if at all. We remedy this problem by adding, at short distances between $\vec{X}$ and $\vec{R}_i$, an extra repulsive force

$$\vec{g}^{\text{add}} = -c_3 \mu \left(L_{i,i+1}\right)^{\mu-1} \left(1 - \frac{r}{\delta}\right) \frac{\vec{R}_i - \vec{X}}{\left|\vec{R}_i - \vec{X}\right|} \qquad (r < \delta) \tag{2.7}$$

$$r = \left|\vec{R}_i - \vec{X}\right|, \tag{2.8}$$

on the entanglement, and a corresponding force $\vec{f}_i^{\text{add}} = -\vec{g}^{\text{add}}$ on blob i. At short distances this extra repulsive force on the entanglement counteracts

the attractive force $\vec{g}^{\rm att} = -\vec{f}_i^{\rm att}$, and prevents the distance r to become too small. Of course in a next step it must be checked if it is profitable to interchange the order of $\vec{X}$ and $\vec{R}_i$. A similar procedure must be applied to all pairs of connected objects coming close to each other. The value of δ was chosen equal to 10^{-4} nm, small enough to have a negligible impact on the configurations of the entangled chains, but allowing for a successful convergence of the minimization procedure.

2.2.2 Detecting new (dis)entanglements

In contrast to crosslinks in rubbers, entanglements have a finite lifetime and are continuously appearing and disappearing. In the model that is developed here, a polymer chain is viewed as a succession of objects, be they blobs or entanglements, connected by line segments. During the simulation we keep track of all unattached pairs of line segments which are close together. They may, for instance, be extracted from the blob neighbor list. For each pair of line segments and at each instant of time the following triple product is calculated:

$$V_{ij} = (\vec{r}_i - \vec{r}_j) \cdot ((\vec{r}_{i+1} - \vec{r}_i) \times (\vec{r}_{j+1} - \vec{r}_j)) \,, \tag{2.9}$$

in which i and $i+1$ label two consecutive objects along a chain, i.e. define the first line segment. The second line segment is defined by j and $j+1$ [see Fig. 2.1(a)]. Note that the absolute value of the triple product is the volume of the parallelepiped defined by the vectors $\vec{r}_{i+1} - \vec{r}_i$, $\vec{r}_{j+1} - \vec{r}_j$, and $\vec{r}_i - \vec{r}_j$. There are two possibilities for the "volume" V_{ij} to become zero.[1] First, if the distance between the two line segments becomes zero. Second, if the two line segments are parallel. In a molecular or stochastic dynamics simulation the latter possibility can be neglected (the chance of two lines running *exactly* parallel at any time is extremely small, even considering the limited machine precision), so if V_{ij} changes sign from one time to the next, a possible bond crossing has occurred.[2] However, Eq. (2.9) does not distinguish between the physical crossing of two finite line segments and the crossing of two infinite lines. An additional check has to be made to be sure that the crossing is taking place along the physical part of the two finite line segments. In order to calculate the exact crossing point, first the exact time of crossing during

[1] A third possibility is that the length of either vector becomes zero. In this rare case impreciseness of the minimization step may lead to false (dis)entanglements. Therefore, if the distance between two connected objects is smaller than ε (introduced in Sec. 2.2.3) it is not allowed to (dis)entangle.

[2] Only in an extremely rare case will two line segments simultaneously cross and alter relative orientation during one time step. In that case V_{ij} will not alter sign. We are disregarding entanglements and disentanglements of this kind.

the last time step must be calculated. Although in principle this involves solving a third order equation, it is found empirically that solving a linear interpolation of V_{ij} in time gives nearly the same time of crossing. Having determined the positions of the objects at this time, the next task is to find the position $\vec{X}$ where the two line segments have crossed [see Fig. 2.1(b)]. To this end define the parameters λ_1 and λ_2 by

$$\begin{aligned} \vec{X} &= \vec{r}_i + \lambda_1 \left(\vec{r}_{i+1} - \vec{r}_i\right) \\ &= \vec{r}_j + \lambda_2 \left(\vec{r}_{j+1} - \vec{r}_j\right), \end{aligned} \tag{2.10}$$

i.e. λ_1 defines the crossing point on the line through objects i and $i+1$, with $\lambda_1 = 0$ at object i and $\lambda_1 = 1$ at object $i+1$, and correspondingly for λ_2. These are three equations with two unknowns, λ_1 and λ_2, so any one equation can be dropped and the remaining two be solved (this stems with the fact that the line segments are already confined to a plane in three dimensional space). A physical crossing between the two line segments has occurred if the crossing point lies between both i and $i+1$ and j and $j+1$, i.e.

$$(0 < \lambda_1 < 1) \wedge (0 < \lambda_2 < 1). \tag{2.11}$$

The algorithm now proceeds as follows: an entanglement is created, initially at the position $\vec{X}$ where the two line segments have crossed each other. In the next time step the minimization of the attractive energy will move the new entanglement to its equilibrium position, and the entanglement will contribute to the elastic forces between chains [Fig. 2.1(c)].

We would like to point out the fact that, in the developed model, two attached (consecutive) line segments can never entangle because the four objects which are connected by an entanglement will all have to be different. If either two objects are the same, the volume V_{ij} between the line segments is always zero and never changes sign.

After an entanglement is created, the associated volume V_{ij} will serve to detect future disentanglements. If the volume V_{ij} of the four objects surrounding an entanglement changes sign from one time to the next, a possible disentanglement has occurred. Additional checks are made in exactly the same way as described for the creation of entanglements, i.e. Fig. 2.1 may also be read backwards.

2.2.3 Non-trivial moves

While searching for its equilibrium position, an entanglement can move freely along the chain between two adjacent objects. At a certain moment, however, the attractive energy would be lower if the entanglement could slip past a blob one position further along the chain backbone, or to the other side of the next entanglement, or, in other words, if the order of objects within a chain would be altered. The order-altering moves are not trivial, but important for a realistic treatment of the entanglement constraints. The algorithm detects if an entanglement has a natural tendency to get close to either one of its adjacent objects by measuring the distance to these objects. If the distance becomes smaller than a prescribed value ε, i.e.

$$\left|\vec{r}_i - \vec{X}\right| < \varepsilon, \tag{2.12}$$

where $\vec{r}_i$ is the position of the adjacent object and ε is sufficiently small compared to the average bond length, a subalgorithm will check order-altering moves. The value of ε was chosen equal to 10^{-2} nm, much smaller than the average bond length of roughly 1.5 nm. The subalgorithm makes the following checks:

1. If the object which is approached is another entanglement go to 2; if it is a blob go to 3.

2. Check if it is physically possible for one entanglement to slip past the other. If it is possible, swap the order in which the entanglements appear in the chain backbone. Otherwise do not swap. Back to main program.

3. If the blob lies at the extremum of a chain, remove the entanglement and go back to the main program. Otherwise go to 4.

4. If a slip past the blob results in an entanglement of a line segment with itself, remove the entanglement (*vide infra*) and go back to the main program. Otherwise go to 5.

5. If a slip past the blob results in disentanglement, remove the entanglement. Otherwise slip the entanglement past the blob. Back to main program.

If the entanglement has survived the slip past the adjacent object, the algorithm will find a new equilibrium position in the next time step. Only one slip per time step is allowed for each entanglement. In the following we will clarify the various checks which are made in the subalgorithm.

Suppose two entanglements are very close to each other, as in Fig. 2.4.

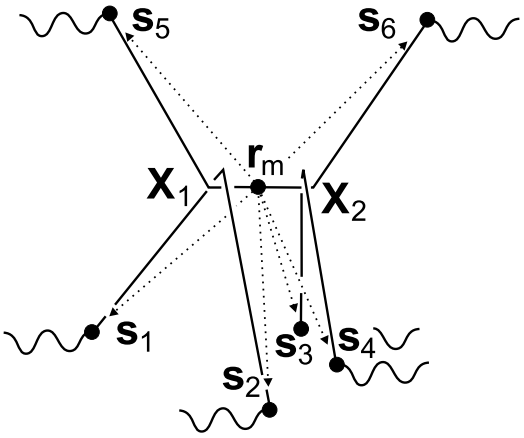

Figure 2.4: *Two entanglements at $\vec{X}_1$ and $\vec{X}_2$ are allowed to slip past each other if the two arms of the first entanglement ($\vec{s}_1$ and $\vec{s}_2$) can both pass over the two arms of the second entanglement ($\vec{s}_3$ and $\vec{s}_4$). The reverse case is also possible.*

The mean position of the two entanglements, $\vec{r}_m$, will then be very close to both entanglement positions, $\vec{X}_1$ and $\vec{X}_2$. Define $\vec{s}_1$ to $\vec{s}_6$ as the vectors going from $\vec{r}_m$ to the surrounding six objects with positions $\vec{r}_1$ to $\vec{r}_6$, in formula:

$$\vec{r}_m = 1/2\left(\vec{X}_1 + \vec{X}_2\right), \tag{2.13}$$

$$\vec{s}_i = \vec{r}_i - \vec{r}_m. \tag{2.14}$$

Now it is assumed that the two entanglements can swap if the two arms connected to the first entanglement, $\vec{s}_1$ and $\vec{s}_2$, *both* go over the two arms connected to the second entanglement, $\vec{s}_3$ and $\vec{s}_4$, or the other way around. In the first case we demand that the projections of $\vec{s}_3$ and $\vec{s}_4$ onto the plane defined by $\vec{s}_1$ and $\vec{s}_2$ both fall in between $\vec{s}_1$ and $\vec{s}_2$. To make this mathematically explicit: both $\vec{s}_3$ and $\vec{s}_4$ must lie in

$$\mathcal{R} = \left\{\vec{r} \in \mathbb{R}^3 \;\middle|\; \begin{array}{l} \vec{r} = l_1\vec{s}_1 + l_2\vec{s}_2 + l_3\left(\vec{s}_1 \times \vec{s}_2\right), \\ l_1 \in \mathbb{R}^+, l_2 \in \mathbb{R}^+, l_3 \in \mathbb{R} \end{array}\right\}, \tag{2.15}$$

i.e. any vector $\vec{r}$ in $\mathcal{R}$ must have positive components along $\vec{s}_1$ and $\vec{s}_2$ in the basis $\{\vec{s}_1, \vec{s}_2, \vec{s}_1 \times \vec{s}_2\}$. For a given vector $\vec{r}$ these components are

$$l_1(\vec{r}) = \frac{(\vec{r}\cdot\vec{s}_1)\, s_2^2 - (\vec{r}\cdot\vec{s}_2)\,(\vec{s}_1\cdot\vec{s}_2)}{s_1^2 s_2^2 - (\vec{s}_1\cdot\vec{s}_2)^2}, \tag{2.16}$$

$$l_2(\vec{r}) = \frac{(\vec{r}\cdot\vec{s}_2)\, s_1^2 - (\vec{r}\cdot\vec{s}_1)\,(\vec{s}_1\cdot\vec{s}_2)}{s_1^2 s_2^2 - (\vec{s}_1\cdot\vec{s}_2)^2}. \tag{2.17}$$

If all four numbers $l_1(\vec{s}_3)$, $l_2(\vec{s}_3)$, $l_1(\vec{s}_4)$, and $l_2(\vec{s}4)$ are positive, the entanglements can swap. Now $\vec{s}_1 \cdot \vec{s}_2 = s_1 s_2 \cos\varphi$, with φ the angle between the two arms $\vec{s}_1$ and $\vec{s}_2$. Since these are never exactly parallel, the denominators in Eqs. (2.16) and (2.17) are always positive, so only the nominators need to be calculated. As already mentioned, the inverse is also possible: the entanglements can swap if the projections of $\vec{s}_1$ and $\vec{s}_2$ onto the plane defined by $\vec{s}_3$ and $\vec{s}_4$ both fall in between $\vec{s}_3$ and $\vec{s}_4$.

Now suppose the entanglement is near a blob. The subalgorithm checks if this blob is the first or last blob of the chain, because then a slip past this blob means that it is slipping off the end of the chain [Fig. 2.5(a)].

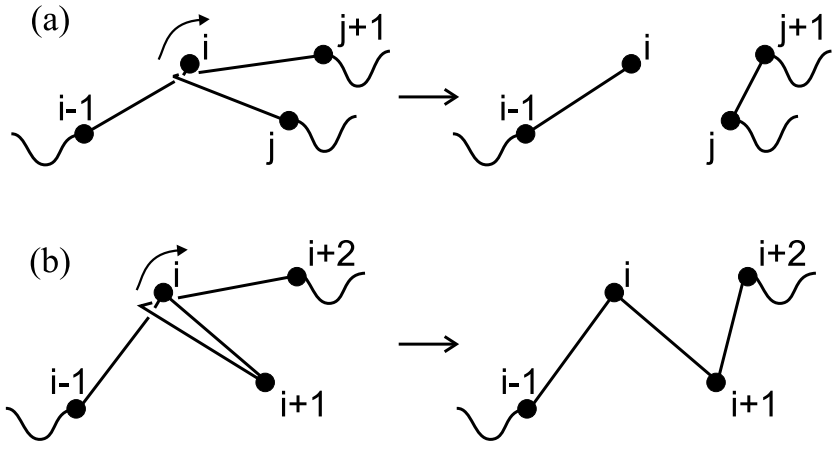

Figure 2.5: *(a) A slip past the last blob of a chain results in disentanglement. (b) If a chain is entangled with itself and the loop is shrinking as far as to pass through only one object apart from the entanglement, a self-disentanglement will occur.*

Usually this is not the case and the program checks if a slip past the blob results in an entanglement of two consecutive line segments, as indicated in Fig. 2.5(b). This happens if a chain is entangled with itself, and the loop

shrinks as far as to pass through only one object apart from the entanglement. Since all four arms of an entanglement have to end at different objects, the entanglement will simply be released in this case. This is called a self-disentanglement.

Finally it is checked if a slip past a blob results in a disentanglement. Suppose two (parts of) chains are entangled. One (part of the) chain is going from an object at $\vec{r}_1$, via the entanglement at $\vec{X}$, to an object at $\vec{r}_2$, and the other (part of the) chain is going from $\vec{r}_3$, via the entanglement, to $\vec{r}_4$, as in Fig. 2.6.

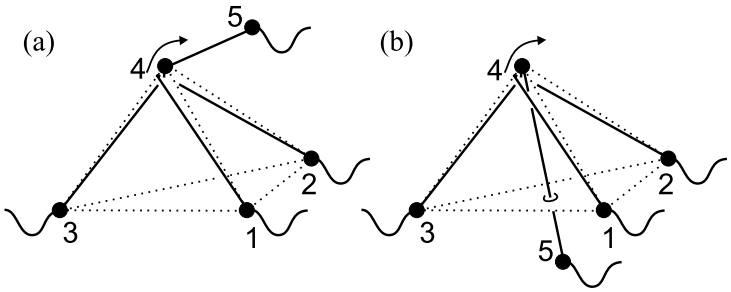

Figure 2.6: *(a) The entanglement X slips past blob 4 if the (extension of) the line from blob 4 to object 5 does not cross the 1-2-3 face of the tetrahedron (dotted lines). (b) The chain disentangles if this face is crossed.*

Due to the very definition of an entanglement it will always be positioned inside the tetrahedron formed by $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3$, and $\vec{r}_4$. Although in principle entanglements may occur in one of many complicated forms, the simplest situation shown in Fig. 2.7(a) is by far the most probable, and we will assume that we will always have to deal with this simple unwinded form.

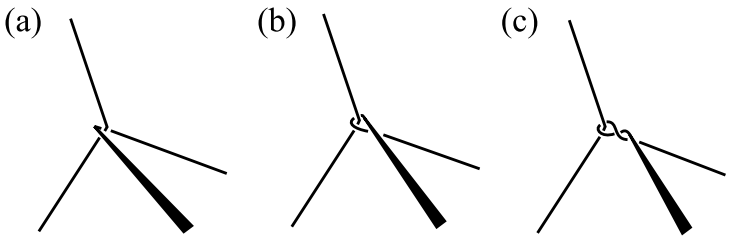

Figure 2.7: *The uncrossability constraint does not distinguish between different link windings. Windings (a), (b), and (c) are all equivalent to the algorithm. Obviously types (b) and (c) will be highly improbable.*

Now suppose the entanglement wants to slip past a blob at $\vec{r}_4$. It will depend on the orientation of the line segment between the blob at $\vec{r}_4$ and the next object at $\vec{r}_5$ whether the entanglement will continue to exists, or must be abolished. Given the unwinded form of the entanglement, it must be abolished if the line segment $\vec{r}_5 - \vec{r}_4$, or its continuation, passes through the $\vec{r}_1$-$\vec{r}_2$-$\vec{r}_3$ face of the tetrahedron, as in Fig. 2.6(b). In all other cases, as in Fig. 2.6(a), a new entanglement equilibrium position must be searched at the other side. A mathematical criterion is easily obtained by equating the expression for a point in the $\vec{r}_1$-$\vec{r}_2$-$\vec{r}_3$ face of the tetrahedron with the expression for a point along the $\vec{r}_5$-$\vec{r}_4$ line,

$$\vec{r}_3 + l_1 (\vec{r}_1 - \vec{r}_3) + l_2 (\vec{r}_2 - \vec{r}_3) = \vec{r}_4 + l_3 (\vec{r}_5 - \vec{r}_4) , \tag{2.18}$$

which can be solved for the parameters l_1, l_2 and l_3. A disentanglement will occur only if

$$(l_1 > 0) \wedge (l_2 > 0) \wedge (l_1 + l_2 < 1) \wedge (l_3 > 0) . \tag{2.19}$$

This completes the description of the non-trivial moves.

2.2.4 Increasing the speed of the algorithm

The introduction of the entanglement algorithm in a standard molecular or stochastic dynamics program will cause a lot of computational overhead. The calculations concerning non-trivial moves are quite complicated, but they do not occur very often and do not consume a considerable part of the cpu time. For the detection of new entanglements in principle all line segment pairs must be checked. Much time is saved by looping over the standard list of close neighbors and considering the line segments attached to these particles. The computational costs are then comparable to the costs of evaluating non-bonded forces in a standard program. The slowest part of the entanglement algorithm is the equilibration step where in principle the total attractive energy, Eq. (2.4), has to be minimized. However, the minimization can be split up in several independent parts by recognizing the fact that moving an entanglement on one side of a blob will not influence the attractive energy residing in the bonds which are connected to an entanglement on the other side of the blob (unless there is a path *around* the blob leading to the other entanglement without encountering any other blobs). For instance, in Fig. 2.8 the parts indicated by "1" and "2" can be equilibrated independently.

Despite this fact, the minimization can still consume between 70 and 95

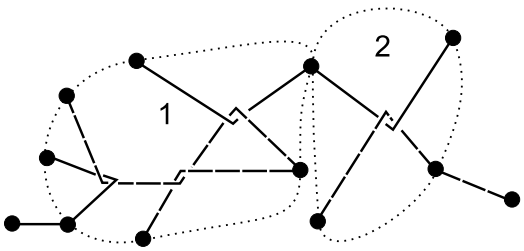

Figure 2.8: *The minimization to determine the positions of the entanglements between these four chains can be split up in two independent parts, indicated by the dotted lines.*

percent of the total time spent in the entanglement algorithm, depending on the efficiency of the minimization procedure and the desired accuracy. In this work we demanded a relative energy convergence of 10^{-8}. A stochastic dynamics program using the entanglement algorithm was roughly 10 times slower than the same program without entanglement constraints. The advantage of being able to coarse-grain, however, has more than compensated this unfavorable factor. More important, it is now possible to investigate the influence of the uncrossability constraint on all kinds of dynamic properties by comparing results with and without use of the entanglement algorithm.

2.3 Energy conservation

With the removal of an entanglement other than disentanglements, potential energy is removed from the system. The algorithm calculates this energy and returns the total energy dissipation (deltak) in one time step to the main program. The main program adds this energy to the system by scaling all velocities.

2.4 Required input for simulations

The TWENTANGLEMENT algorithm can almost be used as a black box, but some input is still required. In *control.dat* four parameters have to be entered and their meaning is described below:

2.4.1 Rcut2

Rcut2 is the cut-off distance used for the pairlist. Normally, this value should be larger than the cut-off for the non-bonded particle interaction. The pairlist

in the TWENTANGLEMENT algorithm should loop over all bonds that could possibly entangle, thus the cut-off should be equal or larger than the maximum bond length.

2.4.2 Excludedparticle

Excluded particle is the type of particle that can be excluded from the TWENTANGLEMENT algorithm (e.g. solvent beads)

2.4.3 Telmax

Telmax is the maximum number of entanglements per tree, see Fig. 2.8 for the definition of a tree. The minimazation procedure of the algorithm will take more and more CPU-time, if the branches grow. Although, the program can handle large trees, with today's computers it's recommended to take no more the 300-600 as a maximum value for Telmax.

2.4.4 Entbondmax

The algorithm can make more entanglements than physically possible. This is due to the fact that the path between two blobs has no volume and infinitely small knots can be formed. Those knots will slow the dynamics in an undesired way. That is way a maximum of entanglements per bond can be entered. The oldest entanglement is removed, if more entanglements are formed.

A good value for Entbondmax is the bondlength divided by the (microscopic, so not coarse grained) monomer size. This value is an approximation of the maximum of 'entanglements' that could occur in a microscopic simulation. For $(\text{-}CH_2\text{-})_{20}$ blobs at 450 K this would be 5. The maximum blob-blob bonded distance is about 2.0 nm and the CH_2-CH_2 (microscopic) non-bonded distance is 0.39 nm.

2.4.5 Entrange

Entrange is the number of successive segments (between beads) for which no internal entanglements are allowed (i.e. Entrange 1 excludes internal entanglements within the bond connecting two bonded blobs, Entrange 2 excludes entanglements in the two bonds between three consecutive blobs, etcetera. Setting it equal to at least the length of the longest chain-1 will prevent any entanglements within a chain.)

2.4.6 Minrange

Minrange is the minimum range (in number of segments) needed for another entanglement between the same pair of segments. Setting Minrange 0 allows entanglements in the direct neighborhood of an existing entanglement, Minrange 1 prevents the nearest bonded neighbors from entangling with the (neighborhood of the) segment of the entangee, etc. Setting it equal to at least the length of the longest chain-2 will allow only one entanglement per chain pair.

3
Overview of the package

3.1 Programs and input files

The simulation package consists of a number of Fortran 90 files and several input files. Using the Makefile, the program can be compiled and the following programs are constructed:

1. INITIATE - generates a configuration file
2. SETZERO - prepares or resets accumulator files needed to measure correlation functions 'on the fly'. Can also remove all formed entanglements and reset velocities to a new shear rate.
3. KRAK - performs the simulation and generates desired output

Four input files are needed for the simulation:

1. control.dat - simulation parameters, output write intervals, correlation lengths and TWENTANGLEMENT constraints
2. parameter.dat - contains force field parameters
3. field.dat - contains the structure and masses of the simulated system
4. config.dat - configuration file (unformatted) generated by initiate and overwritten at every save by KRAK

In Fig. 3.1 a simplified overview of the structure of *krak.f* is given. Detailed information about the structure of the input and output files, is given later on.

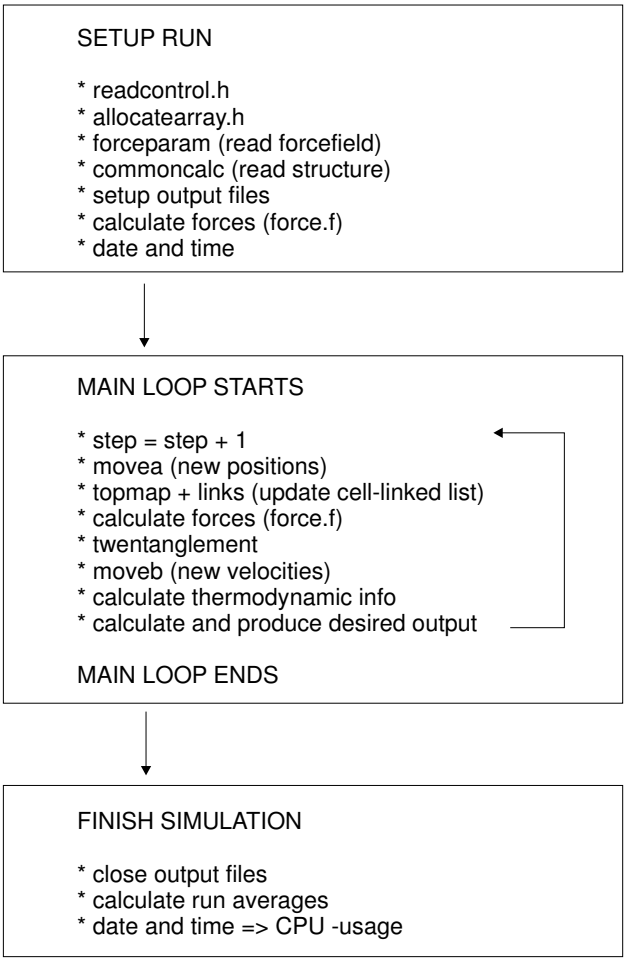

Figure 3.1: *Schematic overview of the main program*

3.2 TWENTANGLEMENT and branched structures

The original TWENTANGLEMENT code was written for a melt of linear chains. The extension to all kinds of other structures is handled in KRAK by composing every molecule as a combination of linear chains. If a molecule is branched, the cross point exists of two or more beads, fixed at the same position. All forces are only calculated once and one bead is moved to its new position. Thereafter its copies are placed at the same new location. The TWENTANGLEMENT code now handles the chains instead of the molecules.

Several examples are given below, in Fig. 3.2 including solvent, chains and branched structures. The bead numbers are given in Arabic style, the chain numbers in Latin style.

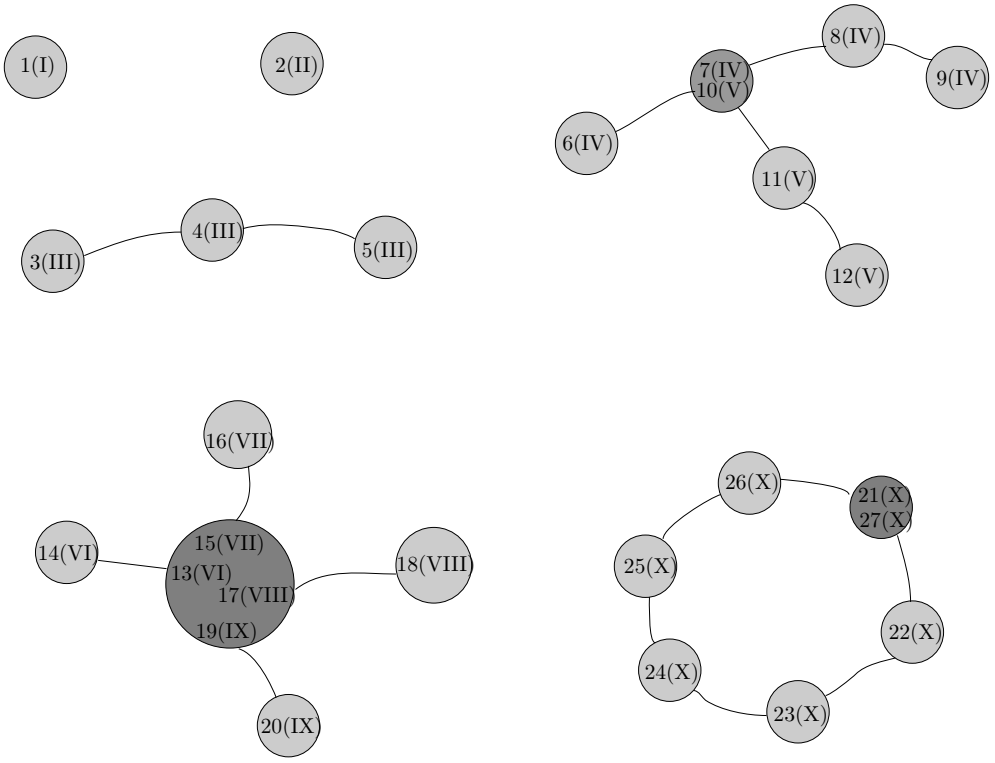

Figure 3.2: *Numbering of chains and beads*

The problem of entanglements, gliding towards a bead with copies is not handled correctly yet. The entanglement is removed at the moment a bead wants to slip over the end. This is not a real problem for star-like polymers, because most entanglements will not get so close to the central bead. However, in ordinary branched structures this problem needs further attention.

The program Initiate is capable of generating starting configurations for molecules, consisting of chains where only the first bead of side chains is allowed to have copies. Thus, close attention should be paid to the order of the chains in a molecule, as well as the order of the beads in the chains. In other cases (like ring-like structures, branched networks etc) the user should generate a config.dat file. The specific format of config.dat can be found in tape.f, subroutine *commoncalc*, and in the paragraph dealing with the input files.

3.3 Units

The following units are used in the program (derived units below the line):

Quantity	Unit	SI
Length	Nanometer	10^{-9} m
Time	Picoseconds	10^{-12} s
Mass	10^{-24} kg	10^{-24} kg
Temperature	Kelvin	K
Force	Nanonewton	10^{-9} N
Pressure	Gigapascal	10^{9} Pa
Energy	Attojoule	10^{-18} J

EXCEPTION: In control.dat the particle mass has to be entered in atomic mass units (Dalton), so a multiplication with 1.66 10^{-3} (10^{-24} kg) is performed internally.

The unit of energy corresponds to 602.2 kJ/mol, the Boltzmann constant is 1.38 10^{-5} (attojoule/K).

4

Input files

4.1 Control.dat

The file contains all the simulation preferences. It is free formatted and consists of keywords. Two examples are given below, all keywords are explained afterwards.

The first is for a melt of linear chains, all consisting of six equal beads. Shear-flow is applied, a movie is generated and the end-to-end vector is calculated, just as the mean square displacement.

```
************** RUN INFO ************************
title          Melt of linear chains
algorithm      BF
nparticles     600
nchain         100
temperature    300.0
timestep       0.05
nstep          1000000
lbox           12.4
cutoff         2.0
shear          0.001
************** ANALYSE INFO *********************
save           200
write          100
mdg            1000
endtoend       100
msd            100 1000
************** ENTANGLEMENTS *******************
entangs        4.0 0
entbondmax     5
```

The second example is for a diblock copolymer system A_3-B_3 with PF in a rectangular box. The order tensor and Rouse mode autocorrelation are measured.

```
************** RUN INFO ************************
title          Melt of linear chains
algorithm      PF
nparticles     600
nchain         100
ntypes         2
temperature    450.0
timestep       0.01
nstep          1000000
varlbox        12.4 10.0 6.0
cutoff         2.0
cputime        3600
************** ANALYSE INFO *********************
save           200
write          100
mdg            1000
order          100
rac            100 1000 10
************** ENTANGLEMENTS *******************
entangs        4.0 0
iclean         20
```

Control.dat has a fixed format. Every line starts with a keyword on the first 12 positions, the needed parameters should be given from position 13 on. Readcontrol.h reads in the first 5 letters from a keyword. If this string is not recognised, the line will be skipped. Available keywords for control.dat are:

keyword	parameters	default value	comment
title	string	'no-title'	title of the run
algorithm	pf or bf	both .false.	algorithm must be defined
nparticles	N	0	number of particles
nchain	maxnchain	0	number of chains
ntypes	maxitype	1	number of different bead types
ncopies	maxcopies	0	number of copies for branched structures
nmoltypes	maxmoltypes	1	number of different molecular types
temerature	temper	0.0	temperature
timestep	dt	0.0	time step (ps)
nstep	nstep	0	number of simulation steps
lbox	lboxxsi	1.e20	length of cubic simulation box (nm)
varlbox	lboxxsi lboxysi lboxzsi	1.e20 1.e20 1.e20	lengths of rectangular box(nm)
cutoff	rcut1	0.	cut-off for the non-bonded interaction
shear	gamma	0.	shearrate (ps^{-1})
entangs	rcut2 excludedparticle	0. 0	default: entanglements off cut-off for entanglements and particletype not included
cputime	maxcputime	604000	maximum cpu-time (s, default: one week) program shuts down properly after maxcputime
barostat	baro tp	0 0.	boxsize adjusting subroutines baro=1: Berendsen NPT, baro=2: NVT x=z, x/y=var tp = barostat coefficient
pressure	pdes	0.	pressure in MPa
save	isave	10000	save interval of the configuration
write	iprint	1000	write interval of thermodynamic data
mdg	iquanta	0	mdg output interval
pos	itapepos	0	write interval of positions and velocities
stress	istress	0	stress tensor write interval
sac	isac scorr	0 0	stress auto corr write interval and correlation length
rac	irac rcorr kmax	0 0 0	rouse mode autocorrelation write interval, correlation length, maximum mode
msd	irms ncorr	0 0	mean square displacements write interval, correlation length; only the msd of the com of moltype 1
endtoend	iendtoend	0	average endtoend-length of all chains(!)
order	iot	0	write interval of the order tensors
gyration	irg	0	write interval of the radius of gyration
density	idisrt maxbins	0 0	local density around bead 1 write interval and nr. of bins
telmax	telmax	500	maximum nr. of entangs per tree
entbondmax	entbondmax	5	maximum nr. of entangs for one bond
entrange	entrange	1	entrange
minrange	minrange	0	minrange
iclean	iclean	50	interval for entangs cleanups
maxentbin	maxentbin	100	maximum nr. of entangs for histogram
maxagebin	maxagebin	100	maximum nr. of bins for entangs age

The minimum control file should contain: N, dt, maxnchain, nstep, rcut1, an algorithm and boxlengths. The boxlengths are needed for the initial configuration made by initiate.exe. Krak.exe uses the values for the box dimensions stored in config.dat, i.e. it does NOT necessarily use the values given in control.dat!

4.2 Config.dat

Initiate can make the configuration file, but that program is not able to generate suitable configurations for all structures (see the paragraph about TWENTANGLEMENT and branched structures). In that case a config.dat file has to be generated by the user and it should be built up as follows (unformatted):

TYPE	VARIABLE	DIMENSION
real*8	lboxxsi, lboxysi, lboxzsi	(1)
integer	N, Nentmax	(1)
real*8	rx, ry, rz	(N)
real*8	vx, vy, vz	(N)
real*8	urx, ury, urz, uvx	(N)
integer, real*8	step, strain	(1)
real*8	acv, ack, ace, acp, act	(1)
real*8	acvsq, acksq, acesq, acpsq, actsq	(1)
integer	nent	(1)
real*8	ex, lex	(3*Nentmax)
real*8	volent	(Nentmax)
integer	next, prev	(Nentmax)
logical	entang	(Nentmax)

Nentmax is the maximum number of entanglements. The program adjusts this value of the maximum is reached, so any realistic value will do (only zero should not be used).

4.3 Parameter.dat

parameter.dat contains different data for BF and PF, but the first part is equal for both algorithms.

The file is free formatted, each parameter on a new line in the following order. The first line starts with a zero, the rest of the line is comment. From there on, $\frac{1}{2}(maxitype) * (maxitype + 1)$ blocks of interaction parameters are needed, divided by a comment line, starting with a 0. The order of these blocks is: A-A, A-B, A-Cc A-MaxITYPE, B-B, B-C etc. For LD and BD

these blocks have the following order:

```
0 A-A interaction
3.d0      ; non-bonded distance (nm) (b0)
1.d0      ; non-bonded gaussian pre-factor (attojoule) (c0)
0.0d0     ; NON-USED: was:  bonded, first gaussian width (nm) (b1)
0.0d0     ; NON-USED: was:  bonded, first gaussian pre-factor (attojoule) (c1)
0.0d0     ; NON-USED: was:  bonded, second gaussian width (nm) (b2)
0.0d0     ; NON-USED: was:  bonded, second gaussian pre-factor (attojoule) (c2)
0.001d0   ; bonded, powerlaw prefactor (c3)
10.d0     ; bonded, powerlaw exponent (μ)
1.d0      ; angle, powerlaw prefactor (attojoule) (c4)
1.2d0     ; angle, powerlaw exponent (ν)
```

This block is followed by a line starting with a zero, furthermore containing comment, after which in case of BF, MaxITYPE background friction coefficients have to be given.

```
0 friction coefficients
8.d0   ;A-friction
12.d0  ;B-friction
```

For PF $\frac{1}{2}(maxitype) * (maxitype + 1)$ pairwise friction coefficients have to be given.

```
0 friction coefficients
0.3  ;A-A friction
0.1  ;A-B friction
0.1  ;B-B friction
```

4.4 Field.dat

The file is equally built up for all four dynamic schemes, free formatted and consists of three blocks of data. The first block of N-lines, contains respectively the chain-number, the particle type and MaxCOPIES columns with the copies of the i-th particle. If a particle has no copies, zero's have to be filled in, and at least two columns with copies have to be given. All values are integers.

```
chain-nr  particle-type  copy1  copy2  ...
```

For a melt of two diblock copolymers A_2-B_2 chains, the first eight lines are:

```
1   1   0   0
1   1   0   0
1   2   0   0
1   2   0   0
2   1   0   0
2   1   0   0
2   2   0   0
2   2   0   0
```

More examples on the numbering for *field.dat* are given in 3.2. The N+1th line should start with a zero. The following MaxITYPE lines should contain the masses of the particle types (real*8), followed by a zero in the line below those masses.

The last block of the field-file (nummol-lines) contains the molecular types of all the molecules (so not per definition the chains) of the system, followed by a zero again. For the example of two diblock copolymers A_2-B_2 chains, this could be:

```
0 masses (amu)
320
250
0 moltypes
1
1
0 end field.dat
```

The molecular types are allowed to run from 1 to MaxMOLTYPE. If different molecules are present (for example polymer and solvent), every molecule can be given its own type-number, so that output data are generated per molecule-type. The program does not perform a check if all molecules of type i are really equivalent, neither is checked if type i and type j are really different molecular structures. Thus, in the case of two diblock copolymers A_2-B_2 chains, the first chain can be given moltype 1, and the second moltype 2. In that case the output (e.g. mean square displacements) will be generated per chain in stead of averaging over both chains.

5

Output files

The standard output is written to the screen, but can easily be printed to a file (e.g. KRAK >! out.dat &). The relevant data from *control.dat* is printed and during the simulation every k-steps (thermodynamic write interval in *control.dat*) energy, pressure and temperature are printed. At the end of the simulation the averages are printed, just as the total CPU-time consumed.

In *control.dat* other write-intervals can also be given a non-zero value. It will result in generating output files. The contents of those output files are described below. If entanglements, stress autocorrelation, rouse modes or mean-square displacements have to be calculated, you need to run SETZERO before you perform a run. In all other cases, this is not necessary.

5.1 Config.dat

config.dat contains the configuraton data and is overwritten every isave timesteps. It is used for restarts of the system and is built up exactly as described in 4.2. The file is unformatted.

TYPE	VARIABLE	DIMENSION
real*8	lboxxsi, lboxysi, lboxzsi	(1)
integer	N, Nentmax	(1)
real*8	rx, ry, rz	(N)
real*8	vx, vy, vz	(N)
real*8	urx, ury, urz, uvx	(N)
integer, real*8	step, strain	(1)
real*8	acv, ack, ace, acp, act	(1)
real*8	acvsq, acksq, acesq, acpsq, actsq	(1)
integer	nent	(1)
real*8	ex, lex	(3*Nentmax)
real*8	volent	*(Nentmax)
integer	next, prev	(Nentmax)
logical	entang	(Nentmax)

5.2 Config.mdg

This file is a MD-Graph [5] output file. MD-Graph uses snapshots of the system to make a movie. All particles, entanglements and bonds are shown, just as the simulation box and a moving plane to indicate the shear-flow. MD-Graph is freeware, but not available anymore on the internet. Contact the authors for an installation version (working under windows / linux (with wine)).

5.3 Pos.dat

All positions of the particles are unformatted and sequentially written to this file, folded and the peculiar displacements. This file can be generated for user-made not on-the-fly measurements. Step is an integer, the other variables are real*8 arrays of length N.

step	rx	ry	rz	urx	ury	urz

5.4 Velo.dat

All velocities of the particles are unformatted and sequentially written to this file, folded and the peculiar velocity in the direction of the shear. This file can be generated for user-made not on-the-fly measurements. Step is an

integer, the other variables are real*8 arrays of length N.

```
step  vx  vy  vz  uvx
```

5.5 Stress.dat

The stress-tensor is written in *stress.dat*, divided in the kinetic and virial terms. The file is formatted ('(i6,3(3(f10.6,1x),3x))') and sequentially.

```
step  sxxvv  sxyvv  sxzvv  sxxfr  sxyfr  sxzfr  sxxfr2  sxyfr2  sxzfr2
step  sxyvv  syyvv  syzvv  sxyfr  syyfr  syzfr  sxyfr2  syyfr2  syzfr2
step  sxzvv  szyvv  szzvv  sxzfr  szyfr  szzfr  sxzfr2  szyfr2  szzfr2
```

The stress tensor is given by:

$$\sigma_{\alpha\beta} = \sum_i m_i v_{\alpha,i} v_{\beta,i} + \sum_{i>j} (r_{\alpha,i} - r_{\alpha,j}) F_{\beta,ij} \tag{5.1}$$

VV is the kinetic term in 5.1, FR is the virial term due to the excluded volume interactions, and FR2 is the virial term due to attractive bonding forces and angular forces.

5.6 Msd.dat

This file contains the mean-square displacements of the center of masses all molecular types. The file is free formatted and contains only real*8's.

```
time  msd(moltype=1)  msd(moltype=2)  ...  msd(moltype=maxmoltype)
```

If many molecular types are defined, it is advisable to alter the source code in *correlate.f* to calculate only the MSD of some specific user defined moltypes. Otherwise, the file may not be readable by XMGR.

5.7 Entangevent.dat

In this file a summary of all entanglement events is given. The number of entanglements and other events (unentanglements, slipping of the end, unknotted, entrange reached and maxentbond reached) are printed.

5.8 Entanghist.dat

In this file a histogram of the number of entanglements is written in free formatted form:

```
#entanglements  #encounters  normalized #encounters
```

The number of entanglements runs from zero to MaxEntbin as set in *control.dat*. The last line of the file prints the average number of entanglements.

5.9 Entangage.dat

entangage.dat contains the age distribution of the entanglements in free formatted form.

```
age  #encounters with that age  normalized #encounters
```

5.10 Endtoend.dat

endtoend.dat contains the average length of the end-to-end vector of all chains, so watch out with branched molecules or several molecule types! The file is free formatted.

```
time  average endtoend distance
```

5.11 Rg.dat

The radius of gyration of molecule 1 is defined as:

$$R_g = \left(\frac{1}{N_1 + 1} \sum_{i=1}^{N_1} \left\langle (\vec{R}_i - \vec{R}_{com})^2 \right\rangle \right)^{0.5} \tag{5.2}$$

where N_1 is the number of beads in molecule 1. It is printed every k-timesteps.

```
radius of gyration
```

5.12 Order.dat

The orientation and gyration tensors measure the internal structure of the molecules. The orientation tensor is defined as:

$$\underline{\underline{S}} = \frac{3}{2} \left(< \hat{u}\hat{u} > - \frac{1}{3} \mathbb{I} \right) \tag{5.3}$$

with:

$$< \hat{u}\hat{u} >= \frac{1}{L} \sum_{i=1}^{L} \frac{\vec{r}_{i+1} - \vec{r}_i}{|\vec{r}_{i+1} - \vec{r}_i|} \frac{\vec{r}_{i+1} - \vec{r}_i}{|\vec{r}_{i+1} - \vec{r}_i|} \tag{5.4}$$

where L represents the number of bonds of molecules of type 1. The orientation and gyration tensor of molecules of type 1 are formatted written every k-timesteps. All values are real*8. The last line in every block is the radius of gyration.

```
otxy   otxx-otyy   otyy-otzz
axx    axy         axz
ayx    ayy         ayz
azx    azy         azz
sqrt(axx+ayy+azz)
```

5.13 Rac.dat

The rouse mode autocorrelation is defined as the normalized autocorrelation of:

$$\vec{X}_k(t) = \frac{1}{N}\sum_{i=1}^{N} \vec{R}_i(t) cos\left(\frac{k\pi}{N}\left(1-\frac{1}{2}\right)\right) \qquad k = 0, \cdots, N-1 \qquad (5.5)$$

The rouse mode autocorrelation is printed every k-timesteps in two separate files. Rac.dat contains k=1, kmax, while rac2.dat contains k=kmax+1, 2*kmax.

```
time  rac(k=1)  rac(k=2)  ···  # encounters
```

5.14 Sac.dat

In this file the non-diagonal elements of the stress-tensor autocorrelation function multiplied with k_BTV (leading to the shear relaxation modulus G(t) in GPa) is written in formatted form:

```
time  total G(t)   G(t) excl.kin.terms  G(t) non-bonded
      G(t) bonded  # encounters
```

5.15 Densdistr.dat

In this file, the density distribution around the C.O.M. of molecule one and around the central particle of molecule one are given per bin.

```
r        density1  (1-maxbins)
comment
r        density2  (1-maxbins)
```

6

Details

6.1 Source code

If more analysis of the simulation is needed or a different force field is to be included, the source code has to be changed. In this paragraph we will shortly mention the structure of the files and the arrays already declared.

The package consists of several .f files, containing programs and subroutines and some .h files, containing commonblocks (with global variables) or source lines, used several times in different subroutines. Those files are listed below:

```
krak.f             main program
initiate.f         program initiate for generating initial configuration
setzero.f          program to build the accumulator files
force.f            includes all force-related subroutines
dynamics.f         includes all integration schemes
correlate.f        included all correlation subroutines
mdgraph.f          generates md-graph output files (movies)
random.f           random number generator
tape.f             reads and writes configuration file
forcetabpar.f      reads parameter.dat
positions.f        some blob position operations
neighbourlist.f    includes the cell-linked list subroutine
nag.f              nag-library
TWENTANGLEMENT.f   includes subroutines concerning the uncrossability-constraint

commonblocks.h     includes global variables
commontwent1.h     includes global variables
commontwent2.h     includes some global variables used in jofunc
readcontrol.h      reads in the control file and fills the global variables
runinfo.h          writes the run information to the default output
allocatearrays.h   allocates the size of all dynamic arrays
definitions.h      contains some definitions and constants
filenames.h        contains the names of the output files
```

Several constants, variables and arrays can be useful if for example extra correlation functions have to be added. Take care that the memory allocation is dynamic and that some of the arrays listed below may not have the right dimensions in the first part of the code. The following variables are one dimensional.

```
N              number of particles
NENT           N+2*(number of entanglements)
nummol         actual number of molecules
numchain       actual number of chains
maxitype       (maximum) number of different blob types
maxmoltypes    number of molecule types
maxnchain      maximum number of chains
temper         actual temperature
dt             timestep
step           time divided by timestep
icross         zero without entang algorithm
deltak         energy loss due to unentanglements
```

The following variables are more dimensional:

```
RX, RY, RZ (1..N)               folded blob positions
URX, URY, URZ (1..N)            new peculiar blob positions
VX, VY, VZ (1..N)               new blob velocities
DRX, DRY, DRZ (1..N)            blob displacements in last timestep
NEXT (1..NENT)                  index of next item in chain (0=end of chain)
PREV (1..NENT)                  index of previous item in chain(0=start of chain)
beadinfo(N,1..maxitype+2)       chainnr, type, copies of particle i
friction(maxitype)              the friction constant of itype-blobs
moltype(maxmoltypes)            type of molecule i
mol(N+1)                        shows to which molecule particle i belongs
urcomx(nummol), urcomy, urcomz  mass midpoints of the molecules
```

6.2 Shear

In *control.dat* shearflow can be turned on in the x-direction. The gradient of the flow is in the y-direction. Lees-Edwards boundary conditions [6] are applied and thus the periodic image convention is modified.

6.2.1 BF

In the case of BF the friction of the beads in shearflow is relative to the background. Every bead is given an extra force, depending on the y-coordinate of this bead. While the boxes, surrounding the central box, are not straight above the central box anymore, the x-part of the particle distance in different boxes has to be adjusted. The periodic image convention now looks like:

$$\begin{aligned} cory &= ANINT(ryij/LBOXYSI) \\ rxij &= rxij - cory * DELRX \\ rxij &= rxij - ANINT(rxij/LBOXXSI) * LBOXXSI \\ ryij &= ryij - CORY * LBOXYSI \\ rzij &= rzij - ANINT(rzij/LBOXZSI) * LBOXZSI \end{aligned} \tag{6.1}$$

where DELRX stores the displacement δr_x. If a particle crosses a YZ-plane, its velocity has to be adjusted. Periodic boundary crossing is now handled as follows:

$$
\begin{aligned}
cory &= ANINT(ry(i)/LBOXYSI) \\
rx(i) &= rx(i) - cory * DELRX \\
rx(i) &= rx(i) - ANINT(rx(i)/LBOXXSI) * LBOXXSI \\
ry(i) &= ry(i) - CORY * LBOXYSI \\
rz(i) &= rz(i) - ANINT(rz(i)/LBOXZSI) * LBOXZSI \\
vx(i) &= vx(i) - cory * DELVX
\end{aligned} \tag{6.2}
$$

where DELVX is the difference in velocity between adjescent layers.

6.2.2 PF

In the case of PF, the correction for the pairdistances and the wall crossing are equal to the case of BF, Eq. 6.1 and Eq. 6.2. The friction is pairwise and so no extra force is needed to introduce the shear-flow. However, the relative velocities of the particles in different boxes have to be adjusted, to correct for the differences in velocities of those boxes.

$$
\begin{aligned}
cory &= ANINT(ryij/LBOXYSI) \\
vxij &= vxij - cory * DELVX
\end{aligned} \tag{6.3}
$$

7

TWENTANGLEMENT as plugin

It is also possible to use the TWENTANGLEMENT algorithm within your own program. In that case you need *twentanglement.f*, *commontwent1.h*, *commontwent2.h* and a minimization subroutine as can be found in libraries as for example NAG [7] (F04ARF [8]). So, if you like, you can plug these routines into your favourite molecular or mesoscopic dynamics code, just like any other force routine. Of course, there are some demands on the input parameters, arrays, and on the types of systems that can be simulated. This will be described in this section.

7.1 Blob variables and topology

The number of blobs, chains, and chain lengths must be specified through the following variables:

```
N                          Number of blobs
numchain                   Number of chains
bead1(1 ··· numchain+1)    Index of first blob of each chain.
                           The last blob of ichain is bead1(ichain+1)-1
chainnum(1 ··· N)          Chain index for each blob
```

The new blob positions (where entanglement forces must be evaluated) must be stored in variables RX, RY, and RZ, with indices running from 1 to N. For each chain, these positions must be the unfolded positions (not folded back into a central box). The blob displacements, which have been made from the previous to the new time step, must be stored in variables DRX, DRY, and DRZ, also running from 1 to N. These values are not altered by

the uncrossability constraint.

The order in which objects appear along a chain, be they blobs or entanglements, must be specified by two arrays: NEXT and PREV. These are linked lists, one running from the first to the last blob of a chain, the other one backwards. For example, NEXT(1) is the entanglement or blob which follows after blob 1, and NEXT(NEXT(1)) is the entanglement or blob which follows after that. The NEXT of the last blob of a chain is defined as 0, which is how the end of a chain may be recognized. Conversely, the PREV of the first blob of a chain is also defined as 0, which is how the beginning of a chain may be recognized.

Summary of variables:

```
RX,RY,RZ (1··· N)      New blob positions (unfolded for each chain)
DRX,DRY,DRZ (1··· N)   Blob displacements from previous to new time step
NENT                   N + (2 times the number of entanglements)
NEXT (1··· NENT)       Index of next item along the chain; 0=end of chain
PREV (1··· NENT)       Index of previous item along the chain; 0=start of chain
```

7.2 Entanglement variables

Upon input, the previous blob and previous entanglement positions must be entered through the array EX, with indices between 1 and 3N for the blobs (in the order x1,y1,z1,x2, $\cdots$,zN), and between 3N+1 and 3*NENT for the entanglements, where NENT equals N+2*(# entanglements). Each entanglement occupies 6 (consecutive) positions in EX, 3 for the position of the entanglement in the one chain and 3 for the position of the same entanglement in the other chain (the entangee). Notice that these two positions can differ by an integer number of box lengths in each direction (excluding the effect of strain; see further on) because a chain can be entangled with a periodic image of another chain (or even a periodic image of itself!). In various routines it is convenient to have available the distance that has to be travelled to reach the entangee. This is stored in the array LEX (with the same indexing as in EX).

Given the index i of an entanglement, the index of its entangee (the same entanglement point, but part of the other chain(part)) equals i-1 if i is even, and i+1 if i is odd.

The 'volume' that is associated with a certain entanglement (needed to de-

tect possible disentanglements; see Eq. (20) in [2] must be stored in the array VOLENT, at indices N+1 to NENT. Notice that the same volume must be stored at the indices of entanglement and entangee.

During a simulation, new entanglements are added at the end of the arrays (at indices NENT+1 and NENT+2, then NENT+2 is updated to NENT). Old entanglements may also disappear, leaving 'holes' in the arrays. To determine whether an entanglement is still active or not, a logical array ENTANG is used, which is true by default, but false if the entanglement has been annihilated. To prevent ever increasing memory usage, the holes are removed from all relevant arrays once every ICLEAN time steps (defined in definitions.h) and NENT is reset to its new value. To this end the uncrossability constraints also needs to know the current time step STEP. Notice that all arrays involving NENT have been declared with size NENTMAX (also in definitions.h), which must not be exceeded.

Upon output, EX has been reset to the new blob and entanglement positions, VOLENT has been reset to the new entanglement volumes, and ENTANG updated to the new entanglement situation.

Summary of variables:

```
EX (1···3*NENT)       Previous blob and entanglement positions (unfolded)
LEX (N+1···3*NENT)    Distance to entangee (excluding effect of strain)
VOLENT (N+1···NENT)   Volume associated with entanglement
ENTANG (N+1···NENT)   Logical; states whether this entanglement is still active
NENTMAX               Maximum of NENT
ICLEAN                Array cleaning interval
STEP                  Current time step
```

7.3 Box dimension and shear

The uncrossability constraint must know the size of the (cubic) periodic box and the current strain value at various points. This must be provided through the following variables:

```
LBOXXSI   Box length in x-direction
LBOXYSI   Box length in y-direction
LBOXZSI   Box length in z-direction
LBOXXINV  Inverse box length in x-direction
LBOXYINV  Inverse box length in y-direction
LBOXZINV  Inverse box length in z-direction
STRAIN    Current strain value
LBOXXSI   Box length in x-direction
LBOXYSI   Box length in y-direction
LBOXZSI   Box length in z-direction
LBOXXINV  Inverse box length in x-direction
LBOXYINV  Inverse box length in y-direction
LBOXZINV  Inverse box length in z-direction
STRAIN    Current strain value
```

Notice that STRAIN is not dimensionless, but given as a length: the distance a sliding box has moved relative to the underlying box (which equals the real dimensionless strain times the box length). Because of the periodicity of the system the value of STRAIN may be chosen to range between -LBOXXSI/2 and +LBOXXSI/2.

If simulations are performed under shear, the uncrossability constraint still assumes that the blob and entanglement coordinates are unfolded. This means that the usual Lees-Edwards boundary conditions must be applied with care: only entire chains are allowed to move over one or more box lengths, not individual blobs. In the current main program (krak) this resetting of coordinates is done when the strain has reached a value of L/2 (and must be reset to -L/2). At this point the position of the middle blob of each chain is inspected and transferred to the central box along with all the other objects of that chain. Such displacements will alter the values of EX and LEX (two different chains will generally not be translated in the same direction), which must therefore be updated in the main program. Any displacement in the shear gradient direction is of course accompanied by a change of blob velocities in the shear direction.

Important: The value of LEX, stating the distance to the entangee, is an integer number of box lengths, even under shear because the effect of strain is excluded. This means that the distance to the entangee of entanglement i in the shear direction (x) is calculated by using the distance to the entangee in the shear gradient direction (y), i.e., the distance is

$$LEX(3*i-2)+STRAIN*LBOXINV*LEX(3i-1) \qquad (7.1)$$

7.4 Self linked lists

To save some computational time, in the main program, non-bonded interactions are determined using a cell linked list method, as described in the book of Allen and Tildesley [9]. In *twentanglement.f* it is assumed that exactly such a list is present to scan for possible new entanglements between bonds. For a certain pair of blobs, i and j, a scan is made for possible entanglements between all segments from i to the next connected blob and from j to its next connected blob. This means that the cell size must be larger than usual in order not to miss new entanglements (about two times the maximum extension). This poses a limit to the minimum box size because a minimum of 3x3x3 cells is required for this method to work. The number of cells in each direction is determined by MCELL, and the maximum number of entries in the linked list by N1MAX. Both are defined in *definitions.h*. In the main program it is checked whether the minimum required cell size (as input in control.dat) and the number of cells are compatible with the box size.

(Notice that the above requirements for minimum system size can be relaxed a little if MCELL is indeed chosen equal to 3, because in that case all blob pairs will be checked. Still, the minimum box length L must be at least two times MAX(cut-off length for non-bonded interactions, maximum extension of bonded blobs).

In the main program a copy of the Allen and Tildesley code is included. The 'head-of-chain' array HEAD contains one number for each cell and contains the identification number of one of the blobs sorted onto that cell. This number is used to address the element of the linked list array LIST, which contains the number of the next blob in that cell. In turn, the LIST array element for that blob is the index of the next blob in that cell, and so on. Such a 'chain' of blobs in a cell is terminated by a zero. When non-bonded forces are evaluated, every pair inside the current cell and every pair between the current and (half) the neighbouring cells are considered. In the array MAP is indicated which neighbouring cells must be considered. To allow for sheared conditions, 'interactions' between particles in the cell linked list are evaluated using the 'shearing top layer' approach.

Summary of variables:

MCELLX	Number of cells in box (in x-direction); must be $\geq$3
MCELLY	Number of cells in box (in y-direction); must be $\geq$3
MCELLZ	Number of cells in box (in z-direction); must be $\geq$3
HEAD (1···**3)	First blob in each cell
LIST (1···)	Linked list of blobs
N1MAX	Maximum array length of LIST
MAP (1···*MCELL**3)	Mapping of neighbouring cells

7.5 Force field parameters

In its current implementation, the uncrossability constraint needs two force field parameters in order to function correctly. These are the power law exponent μ and the power law pre-factor c_3. The contribution to the 'entanglement' potential energy VENT due to the stretching of a bond (with path length u=r1+r2+···+rp) is cast in the following form:

$$vent = vent + c_3 u^{\mu} \tag{7.2}$$

The magnitude of the attractive force, which acts through this bond, is the derivative and also defines the contribution to the virial WENT:

$$f_1 = \mu c_3 u^{mu-1.d-0} \tag{7.3}$$

$$went = went - f1u/3 \tag{7.4}$$

$$\tag{7.5}$$

The direction of the force is always along the line segments, so the force on a blob is always directed between the blob and the first entanglement encountered along the chain (see Eq. (17) in [2] . To avoid singularities in the force an extra repulsive force is added for very small line segments. If the length of a line segment, r, is smaller than rsmooth (called δ in the article in JCP and defined as 10-4 nm in *twentanglement.f*), a force of magnitude

$$f_1 \left(1 - \frac{r}{rsmooth}\right) \tag{7.6}$$

is added to the objects at both sides of the small line segment. This causes the total force to be exactly zero when r=0. If it is assumed that the path length u is large compared to rsmooth (which in practice is correct to high accuracy), the extra potential energy term can be found by simply integrating the repulsive force from rsmooth to the value of r. Similarly an extra virial contribution is introduced:

$$vent = vent + \frac{f_1}{2}\left(rsmooth - 2r + \frac{r^2}{rsmooth}\right) \tag{7.7}$$

$$went = went + f_1\frac{r}{3}\left(1 - \frac{r}{rsmooth}\right) \tag{7.8}$$

$$\tag{7.9}$$

Upon output, the forces FXENT, FYENT, and FZENT contain the forces due to entanglements. Also, the total forces FX, FY, and FZ have been updated (i.e. added to previous force values) with these bond-stretching forces.

If the user wants to alter the form of the bond-stretching potential, he must bear in mind that a monotonically increasing function is preferred. Otherwise there may not be a unique minimum when minimizing the entanglement positions. The inclusion of angular potentials may be dangerous, because these will define a torque on each line segment, leading to infinite forces when line segment lengths become zero. Also, defining angular potentials across blobs will introduce unwanted dependencies of one 'tree' on another.

c_3, μ	Force field parameters (bond stretching part)
vent, went	Potential energy and virial due to entanglements
fx, fy, fz (1···N)	Forces on blobs; these are updated with:
fxent (1···N)	Forces in x-dir. on blobs due to entanglements
fyent (1···N)	Forces in y-dir. on blobs due to entanglements
fzent (1···N)	Forces in z-dir. on blobs due to entanglements

Bibliography

[1] J. Padding, *JOMD simulation program*, Twente University.

[2] J. Padding and W. Briels, *Uncrossability constraints in mesoscopic polymer melt simulations: Non-Rouse behavior of* $C_{120}H_{242}$, J. Chem. Phys. **115**, 2846 (2001).

[3] M. Allen, *Brownian dynamics simulation of a chemical reaction in solution*, Mol. Phys. **40**, 1073 (1980).

[4] P. Hoogerbrugge and J. Koelman, *Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics*, Europhys. Lett. **19**, 155 (1992).

[5] http://www.sintef.no/units/chem/chemengr/staff/paals/mdg, 2002.

[6] A. Lees and S. Edwards, *The computer study of transport processes under extreme conditions*, J. Phys. **C5**, 1921 (1972).

[7] http://www.nag.co.uk, 2003.

[8] http://www.nag.co.uk/numeric/fffg/ff01/f/f04arf.html, 2003.

[9] M. Allen and D. Tildesley, *Computer simulation of liquids*, 7 ed. (Oxford science publications, New York, 1993).