Brownmove-Setup-Documentation

Tihamér Geyer

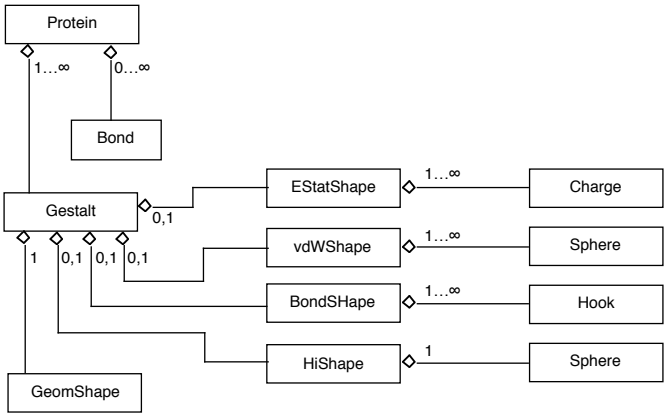
Zentrum für Bioinformatik, Universität des Saarlandes, D–66041 Saarbrücken, Germany
tihamer.geyer@bioinformatik.uni-saarland.de

V1.7 — Sep. 2012 for Brownmove 1.4

The Structure of a Simulation

Each simulation is performed inside a so-called "cup" (remember the "cup of hot tea" used to drive the "Heart of Gold"?), a container for the walls that encloses the actual simulation volume and a number of "proteins", which are the moving objects.

A "protein" is a hierarchical structure defining a bead-spring model of a polymer or a biological protein. The individual beads move and interact with other beads independently. A protein thus contains at least one rigid gestalt object, resembling the "beads". Each gestalt again contains a number of shapes, which model the vdW surface of this rigid unit, the effective charges, the hooks to which the springs are attached, or a shape for the hydrodynamic interactions. This structure is shown in the following figure. The GeomShape object is special as it is not involved in collecting forces from the interactions, but converts the total force and torque into the respective displacements. Consequently, this shape must always be present, whereas the other shapes only have to be present if the corresponding interactions are used in the model.



Interactions

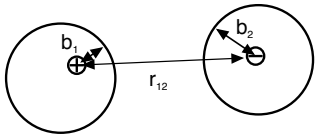
The "natural" units of brownmove are one nanometer for distances, one picosecond for time intervals, one kDa for masses, and one kJ/Mol = 9.9278 x 10^{-4} kDa nm^2 ps^{-1} for energies. Charges are given in elementary charges. Thus, the vacuum dielectric constant ε_0 = 5.728 x 10^{-4} e^2 $(kJ/mol)^{-1}$ nm^{-1}. Forces are given in 1 kJ/mol nm^{-1} = 1.67 pN. Densities are in particles per nm^3, which means that 1M = 0.6022 nm^{-3}.

Some more constants in the brownmove units: Boltzmann constant: k_B = 8.3145 x 10^{-3} kJ/mol K^{-1}; viscosity of water at 20°C (293 K): η_0 = 0.532 kDa nm^{-1} ps^{-1}. Consequently, translational diffusion ceofficients are given in nm^2 ps^{-1}.

Electrostatic Interactions

Electrostatic interactions are modelled as shielded point charges a la Debye-Hückel. The interaction energy between two point charges q_i and q_k (on different gestalt objects) is

$$E_{ik} = \frac{1}{4\pi\epsilon\epsilon_0}\, q_i q_k\, \frac{\exp[-\kappa(r_{ik} - B_{ik})]}{(1 + \kappa B_{ik}/2)^2 r_{ik}}$$



Here, ε is relative dielectric constant of water and ε_0 the vacuum dielectric constant. The shielding from the ions is captured in the inverse Debye length $\kappa = 1 / l_D$. Mostly, the charges are embedded inside the protein, where no counter ions shield the interactions. $B_{ik} = b_i + b_k$ is the sum of the the effective burial depths of the two charges, b_i and b_k, respectively.

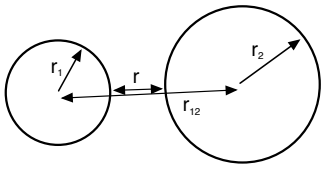
If no value is specified, brownmove uses a typical physiological ion strength of 0.9 M as a default, i.e., l_D = 1.01 nm.

van-der-Waals Interactions

In brownmove, the van-der-Waals interactions denote the effective short ranged repulsions and attractions between the protein surfaces. For this a number of different force types are implemented: the usual Lennard-Jones potential, a constant force, and a potential that models the interaction between metallic beads with a polymer coating.

The first type is the Lennard-Jones potential. This can also be used to model hydrophobic interactions. For this, an r^{-12}, an r^{-6} , and an r^{-3} term are used for these phenomenological interactions between two of the spheres with a distance $r = r_{12} – (r_1 + r_2)$ between their closest points:

$$E = C_{12}\left(\frac{r_0}{r + dr}\right)^{12} + C_6\left(\frac{r_0}{r + dr}\right)^{6} + C_3\left(\frac{r_0}{r + dr}\right)^{3}$$



Usually, C_{12} would be positive and C_6 negative. As usual, r_0 determines the width of the potential (well). The potential is defined such that when the two vdW spheres touch, i.e., at $r = 0$, the repulsive potential has a positive value of $V_0 = 1\ k_BT$. For the usual repulsive Lennard-Jones-6-12 interaction, i.e., with, $C_{12} > 0$ and $C_3 = 0$, this leads to

$$dr = r_0\left[\frac{2C_{12}}{\sqrt{C_6^2 + 4C_{12}V_0} - C_6}\right]^{1/6}$$

The same formula (with a different V_0) can be used to determine a minimal distance R_{min} (which is actually negative) from which on the interaction is linearized for numerical stability. If the potential should be linearized for, e.g., V ≥ 5 kT, then $R_{min} = dr(V_0 = 5kT) – dr(V_0 = 1kT)$, The last parameter is a maximal cutoff radius R_{max}. For distances $r > R_{max}$ the evaluation of the interparticle force is skipped and a zero returned. To avoid numerical artefacts, R_{max} should be larger than the extent of the potential.

To allow for different interaction parameters between different pairs of particles, each vdW sphere has an index, named the "vdW color". At setup, for any occuring pair of colors a set of interaction parameters {r_0, dr, C_{12}, C_6, C_3, R_{min}, R_{max}} has to be specified. These parameters are globally visible in the simulation.

For vdW interactions between particles and walls, the same form of the interaction potential is used with the same global vdW color indices.

The next type of interaction was implemented to describe the interaction of alcyl thiol coated gold nanobeads [Geyer, Born, Kraus, PRL (2012)]. It consists of a dispersive Hamaker type attraction between the gold cores plus a repulsion from the overlap of the ligand shells. The Hamaker potential between two spheres of equal radii R at a center-to-center distance of r has the form

$$V_{Ham}(r) = -\frac{A_{SS}}{6}\left[\frac{2R^2}{(r^2-4R^2)} + \frac{2R^2}{r^2} + \ln\left(\frac{r^2-4R^2}{r^2}\right)\right].$$

Typical values of the Hamaker constant A_{SS} for gold are about 125 kJ/mol = 1.3 eV. Note that this attractive potential diverges for $r < 2R$. In the nanobead project we therefore needed a repulsion that diverges faster than the Hamaker potential for stable simulations. This was modeled from the mutual overlap volume between the ligand shells times a distant dependent potential for jamming the alkyl thiol ligand chains of length L into each other:

$$V_{alk}(r) = \frac{A_{LL}\left(\frac{R-\Delta}{R+L}\right)^2}{1-\left(\frac{r-\Delta}{r-(R+L)}\right)^2}\left[\frac{2(R+L)^3}{3} - \frac{r(r+L)^2}{2} + \frac{r^3}{24}\right] \quad \text{for } r \le 2(R+L)$$

Here, A_{LL} describes the overall strength of this term, while Δ describes a bending of the ligands so that the two sphere surfaces may come closer than L.

A third variant for these effective short ranged interactions is a linear potential, i.e., a constant force that acts when the surfaces of the two spheres (or of a sphere and a wall) come closer than a specified distance.

Sticky Interactions

In coarse-grained simulations the particle surfaces are often much smoother than for example those of real proteins or of the above mentioned alkyl thiol coated gold nanobeads. To allow for some "stiction" (static friction) between the otherwise smooth shapes, so-called sticky interactions were implemented. In these, a layer is defined around a sphere, and when the layers from sticky spheres of two particles overlap, a temporary harmonic bond of length zero is formed halfway between the centers of the two spheres, i.e., the two shells are attached to each other in the overlap region. This bond may break again when the displacement during a timestep exceeds a specified threshold, i.e., when the force on the bond becomes too strong. However, the bond may be re-established when the two spheres are still overlapping.

Though these sticky interactions were introduced to describe the "sticky" overlap of ligand shells, they can also be used to model for example unspecific, transient binding between proteins due to their surface roughness and small hydrophobic patches.

Hydrogen Bonds

A hydrogen bond with its directionality is implemented in the following way: for both donor and acceptor, the positions of the heavy atom and of the hydrogen are defined. Then the two "hydrogens" attract each other with an inverted Gaussian potential, while the heaavy atoms repell each other with a harmonic force when they are closer than the combined distances between the heavy atoms and the respective hydrogen.

Bonds

Bonds are harmonic and quartic interaction potentials between the "hook-up points" in the bondShape objects with (here for only a harmonic interaction)

$$E_{ik} = \frac{k_{ik}^2}{2} \left(r_{ik} - L_{ik}\right)^2$$

For any pair of hooks, a spring constant k_{ik} and a rest length L_{ik} have to specified within the scope of the protein. A bond can not be broken.

To be able to discern between local and global interactions, also bonds of vdW- and Coulomb-type can be defined. They use the same interactions as between the respective shapes. When such a bond is defined between two gestalten, the interactions between the corresponding shapes are ignored.

Hydrodynamic Interactions

The hydrodynamic interactions are modelled with the truncated expansion algorithm as described in [T. Geyer and U. Winter, "An $O(N^2)$ approximation for hydrodynamic interactions in Brownian dynamics simulations", *J. Chem. Phys.*, **130** (2009) 114905]. For the translational interactions, the modified Rotne-Prager-Yamakawa tensor of Garcia de la Torre and Bloomfield [*Biopolymers* **16** (1977) 1747] is used which allows for particles of different sizes. Note that this form of the RPY tensor diverges with increasing particle overlap. If the radii of the HIShape spheres are set to zero, the implemented version of the RPY tensor collapses to the Oseen form. To test how important HI is for the system behavior, the strength of the HI can be changed by a global scaling factor (which is there for only this purpose and has no physical meaning).

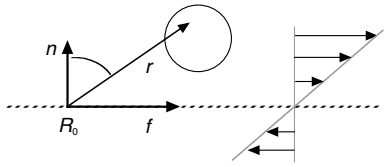
By default, only translational HI are used. To enable (the yet not so thoroughly tested) rotational HI, set the flag -DHI_WITH_ROTATION in the file config.mk during compilation.

External Forces

External position dependent forces can be attached via hook-up points in an ExternalShape object. Currently, a constant force, a harmonic potential, and a shear force are implemented. The actual force is the product of the (position dependent) external force $F(r)$ and a "weight" α, such that the same external potential can excert different forces on different hook-up points. For a gravitational force, α would be the mass, while an α proportional to the particle radius leads to the same drift velocities of different sized particles.

The shear force is calculated with the formula given below. Note that the force f and the normal n do not have to be normalized or even orthogonal to each other (though this would be the usual case).

$$\vec{F}(r) = \alpha \left(\vec{n} \cdot (\vec{r} - \vec{R}_0)\right) \vec{f}$$



Propagation algorithms

Brownmove used a Langevin Dynamics (LD) propagation algorithm formulated in terms of forces as described in [U. Winter and T. Geyer, "Coarse Grained Simulations of a Small Peptide: Effects of Finite Damping and Hydrodynamic Interactions", *J. Chem. Phys.* **131**, (2009) 104102]. This formulation allows to continuously blend from the LD to the standard Ermak-McCammon BD propagation by setting the particle

mass to zero. It even allows for mixed setups where small particles are "massless" and overdamped while for the large particles accelleration is considered explicitly.

Setup Configuration File

The simulation package "brownmove" is essentially a many-particle BD and LD library, but can also be used as a black-box simulation tool. For this, many interesting simulation scenarios can be defined through configuration files as described below. At least two files are required: one global configuration and one or more files that define the proteins. Often, there is a another file describing the simulation box (environment).

General Stuff

This central configuration setup file contains global information about the simulation like length, timestep, output interval, general interaction parameters, or output scripts as well as definitions of the simulation box and the proteins.

Configuration lines start with a case-insensitive keyword followed by the required data. Any text beyond the last required data is ignored and can be used for comments (even without a leading '#'). Keywords and data are separated by whitespace, i.e., any number of spaces or tabs. Leading whitespace is ignored, indentation can be used freely to increase readability. Empty lines or lines starting with a '#' are considered comments and are skipped.

To make the setup files more general and flexible, a simple text replacement feature allows to define arbitrary constants. A definition is indicated by the keyword "constant" followed by the (case-sensitive) label. The rest of the line (up to the actual line end or a comment character) is stored as replacement string. Note that (currently) trailing whitespace is copied, too, whereas the whitespace between the label and the first non-white character of the replacement can have any length. An occurance of a constant is then indicated by a preceeding "$" sign. Constant names should be unique, i.e., one label should not be a substring of another constant's label. Constants can occur anywhere in the setup files and are effective after they have been declared. A subsequent re-definition of a constant with a different replacement string overwrites an earlier definition. In the following example a constant MASS is defined, which is then used to to declare a constant to be used as the name of an output file.

```
constant   MASS      134.5
constant   OUTFILE   attraction_m$MASS.snapshot
```

In this example the outputfile name evaluates to "attraction_m134.5.snapshot" and can be used via $OUTFILE.

Since version 1.2 brownmove also accepts constant declaration on the commandline allowing, e.g., to scan a parameter range from a script without the need to change the input files for every parameter value. In the following example, a parameter VDW_RADIUS is defined upon start of brownmove. Note that these definitions follow the input file (here „mysetup.bsc“) and are effective before any input file is parsed and that they can potentially be overridden by a later definition.

```
# call brownmove with a constant
./brownmove mysetup.bsc -d VDW_RADIUS 5.24
```

Another addition is the "evaluate" keyword which defines a constant by evaluating the remainder of the line. This allows to perform simple calculations during the setup process. The expression is evaluated by a systems call to /bin/perl after any constants have been replaced by their actual values.

```
# define the radius and the volume of a sphere
constant  radius  2.5
evaluate  volume  4.0 * 3.1415 / 3.0 * $radius * $radius * $radius
```

The "echo" keyword allows some kind of logging output. Anything following this keyword is simply echoed to the console after variable substitution.

Parts of a setup file (both in the global setup and in a protein or box definition) can be repeated with the "repeat - endrepeat" keywords. "Repeat takes a counter variable (constant), a start value and an end value. When the parser finds repeat statement, it just reads and stores the following lines until it runs into an "endrepeat". Then, the counter variable is set to the start value and when this value is not larger than the end value, the stored text is executed. Then, for the next iteration, the couter variable is incremented by 1 and the stored text is evaluated again until the counter value is larger than the end value. The following example places five (previously defined) proteins along the x axis with 5 nm between the centers. Any indentation is ignored and can be used to improve readability.

```
repeat PLACE_CTR 1 5   # runs PLACE_CTR through 1, 2, 3, 4, 5
    evaluate XPOS  5 * ($PLACE_CTR - 3)
    echo ++++ placing protein $PLACE_CTR at x position $XPOS
    startwith  myProtein  $XPOS 0.0 0.0  0.0 1.0 0.0 0.0
endrepeat
```

Repeat-endrepeat loops in the main setup file are independent of loops in protein or box definitions, but currently, loops can not be nested within one file.

The first section of the global setup file usually defines the timestep and the simulation duration in timesteps. The output interval is defined in units of timesteps. There is no specific order of these three lines, but they all have to be present.

```
# parameters for the simulation
timestep    100.0            # [ps]
runsteps    5000000          # length of sim in timesteps
outsteps    100              # output every .. timesteps
```

Some of the global parameters can also be changed from their defaults. Currently, these are the solvent temperature (defaulting to 293 K = 20°C), the inverse Debye length $\kappa = l_D^{-1}$, which defaults to 0.99 nm^{-1}, i.e., a physiological ion concentration of 0.09 M, and the relative dielectric constant of water, $\varepsilon_{H2O} = 78$. For simulations with explicit ions, where no implicit shielding of the Coulomb interactions occurs, set $\kappa = 0$.

```
# solvent temperature in Kelvin
# temperature     293
# shielding parameter for Coulomb interactions: kappa = 1/lD
# kappa           0.99
# relative dielectric constant of water
# eps_H2O         78.0
```

Then, there are some optional keywords to change the simulation's behavior.

```
# The following keyword prevents the final dump of the proteins after the sim
noProteinDump
```

When you want to continue a simulation from the final dump, you can use the keyword "offsetsteps" to start it from a given timepoint. When offsetsteps is set, the initial particle positions are not put out so that the output files can be concatenated directly.

```
# start simulation at a later time, e.g., to continue a too short run
# offsetsteps     20000

# if this parameter is present,
# the random number generator is seeded with this fixed value
# randomseed      1998

# define a global cut-off for hydrodynamic interactions
# hicutoff  20.0        # [nm]
#
# a scaling factor for the strength of the HI
# hiscale 1.0
```

The HI cutoff is currently an untested experimental setting and should not be used.

The output of the simulation is piped into an analyzer script, where a certain pre-processing or sorting of the sometimes vast amounts of data can be performed. This script is defined with the "analysis" keyword. It can be as simple as "cat", which dumps the output to the console, or a pipe of scripts and programs doing a sophisticated on-the-fly analysis. Here are some examples. Note that anything after the keyword and the first following whitespace is passed on to the shell within single quotes. Consequently, in this line there should not be any comments at the end of the command.

```
# analyzer-script
# analysis   cat
# analysis   cat > fish.txt
analysis     grep Punkt | cut -f4-6 > fish.txt
# analysis   grep timeline > anzahl_x-20.txt
```

Global Interaction Definitions

The first set of interaction parameters concern the van-der-Waals interaction. Each vdWShape of a protein is labelled with a vdW-color (starting with 0) according to its type. Here the respective parameters are defined for all potential pairs of vdW interactions. Undefined color pairs are initialized with zero, i.e., no interaction. In the following example, a single vdw color will be used. Various forms for this effective interaction can be modelled with the most important being the Lennard-Jones type. Of course, different types will take different numbers of parameters.

```
# number of different vdW colors to be used
vdwarraysize      3
# Lennard-Jones type: label LJ
# vdwparms  clr1 clr2 type  C_12      C_6  C_3  R_0  DR        Rmin         Rmax
vdwparms     0    0    LJ   9.744594  0.0  0.0  1.0  1.122562  -0.14088552  10.0
# Nano-Bead potential: label MB (here: potential min -2kT at 0.8nm separation)
# vdwparms  clr1 clr2  NB    ASS    ACC    LC    DELTA    Rmin     Rmax
vdwparms    1    1     NB    125    1.67   1.5   0.95     0.65     10.0
# linear slope: label LIN (force when distance < DR, positive = repulsive)
# vdwparms  clr1 clr2  LIN   force   DR    RMIN   RMAX
vdwparms    2    2     lin   100     0.0   -0.1   0.1
```

(Note: for historic reason, also "MB" can be used for the nano bead potential)

For the sticky shapes a similar definition scheme is used, but here only a single index is used. If two sticky spheres have different indices, then no interaction takes place. The stiction is defined independently for radial and angular displacements. For each direction, a force constant for the harmonic potential and a maximal displacement where the bond breaks have to be given. The binding energy is then determined by the potential at the maximal bondlength. A "viscosity" parameter between 0 and 1 defines whether the attachment point remains where it was initially set (at a value of 0) or moves to the new center between the spheres (at a value of 1). At an intermediate value the attachment point is moved by just that fraction of the displacement.

```
# parameters for the sticky beads
stickyarraysize  1
# Parameter line: color C_radial  DR_radial    C_angular DR_angular  viscosity
stickyParms         0   $C_STICKY $DR_STICKY   $C_STICKY $DR_STICKY   0.0
```

The hydrogen bonds interaction parameters may also be changed from their default values with the following definitions:

```
# hbond_depth   25.0    # depth of the potential well in kJ/mol
# hbond_width   0.05    # width of the inverted-Gaussian H-H-potential
# hbond_cutoff   1.5    # cut-off for hbond calculation (H-H-distance)
# hbond_spring  10.0    # spring constant for the donor-acceptor repulsion
```

External (position dependent) forces are defined with the externalForce keyword, a label, the type, and a number of contect specific numbers. Currently, the type may be "constant" for a position independent vectorial force, "harmonic", which is defined by the position of its minimum and the spring constants in *x*, *y*, and *z* direction, or "shear" with its position and the two vectors as explained above. The simulation does not care whether the two vectors are orthogonal, i.e., skewed force fields may be used.

These definitions of the external forces must preceed the protein definitions, because currently the external interactions on the particles are connected to the external fields during protein assembly.

```
# definition of external forces
# single vector for constant force
externalForce   towardsX  constant   10.0 0.0 0.0
# position of min and spring constants for harmonic
externalForce   Well      harmonic   0.0 0.0 0.0   10.0 100.0 1000.0
# origin and two directions for shear
externalForce   Drift     shear    0.0 0.0 0.0  1.0 0.0 0.0  0.0 100.0 0.0
```

Protein Definitions

The next part of the configuration file defines the proteins used in the simulation. Each protein type is defined on a line of its own starting with the keyword "protein", followed by a label for further reference, the filename containing the definition (see below for their format) and the respective position in this file (a protein definition file may contain multiple protein definitions). These definitions must preceed any defintions of the simulation box.

```
# protein definitions: label  file  position (starting with 1)
protein     singleBead  chargedBead.browndef  1
```

The labels assigned to the proteins are used within the general setup process only. They are used to place proteins directly into the box. Reservoirs, which are then attached to the walls, are defined as follows with the keyword, a unique label, the (initial) density, and an inverse volume (see [T. Geyer, C. Gorba, V. Helms,

"Interfacing Brownian dynamics simulations", *J. Chem. Phys.* **120** (2004) 4573-80] for how these constant density boundary conditions work).

```
# Reservoirs of constant density
# reservoir label      initial density   inv. Vol
reservoir   PunktRein     4e-4             0.0
reservoir   PunktRaus     0.0              1.0
```

The inverse volume corresponds to the change in the reservoir density when one particle is added or taken out. In the above example PunktRein is an inifinite reservoir with a constant density, whereas PunktRaus can be used to directly count the particles in it. A reservoir with a volume of 10 x 10 x 10 nm^3 consequently has an inverse volume of `0.001`.

Simulation Box

If one or a few freely diffusing polymers in an unbounded volume are simulated, this part can be omitted. Otherwise the simulation box will be bounded by walls, which can be simple reflecting walls, reservoirs, or walls with a gestalt. A special class of reflecting walls are periodic walls, which displace the particles by one box length upon contact. Such a quasi periodic setup can be defined directly with the following line:

```
# (quasi-)periodic scenery
# with the boxsizes in X, Y, and Z directions
periodicBox       0.0  20.0  20.0    # each in [nm]
```

If any of the dimensions is ≤ 0, no periodicity in that direction is set up. The above 2D box thus has walls at y = ±10 nm and z = ±10 nm, but none bounding the x-direction.

With periodicBox an internal flag is set that tells the shapes to consider an extra copy of the interaction partner one box length away on either side. A periodicBox should thus be larger than a reasonable cutoff for the interactions. Any other scenery is defined in an extra file and given to the parser as

```
# the scenery: walls, etc - only one scene file allowed
box         box.browndef
```

The format of this file is explained further down. Note that periodicBox may be followed by a box definition to, e.g., complete a 2D periodic setup.

Initial Protein Positions

The last part of the global setup file consists of optional lines that directly place proteins into the simulation box. In the following example the simulation is seeded with four of the above defined proteins with the label singleBead. The seven numbers define the initial position: three positions (x, y, z), a rotation angle (in degrees) and a unit vector defining the rotation axis.

```
# start sim with these proteins in place
# (use labels from above)
startWith   singleBead        -4.0 -4.0 -4.0   0.0 1.0 0.0 0.0
startWith   singleBead         4.0 -4.0 -4.0   0.0 1.0 0.0 0.0
startWith   singleBead        -4.0  4.0 -4.0   0.0 1.0 0.0 0.0
startWith   singleBead         4.0  4.0 -4.0   0.0 1.0 0.0 0.0
```

Random initial position can be specified with the "startWithIn" keyword. It takes the protein label and the maximum and minimum positions along the x, y, and z axis, repectively, plus an exclusion radius that should enclose the complete protein. When no exclusion radius is given, startWithIn estimates the size of the

molecule from the vdW spheres. During the random placement, collisions of these exclusion spheres are detected and a new protein position is tried when an overlap with any of the previously placed proteins occurs. After 10000 unsuccessfull tries to place a protein the parser gives up and you have to start brownmove another time. These two keywords can be combined when an optional exclusion radius is given to startWith. Note that startWith itself does not care for collision, so startWith should preceed startWithIn.

```
startWith  largeProt   15.0 0.0 0.0  0.0 1.0 0.0 0.0  12.5
startWith  largeProt  -15.0 0.0 0.0  0.0 1.0 0.0 0.0  12.5
repeat PROT_CTR 1 100   # add another 100 randomly placed proteins
    startWithIn  myProt  -$XMAX $XMAX -$YMAX $YMAX -$ZMAX $ZMAX  $PROT_RADIUS
endrepeat
```

If a region of the simulation box should initially remain empty, exclusion spheres can be placed manually with the excludedvolume keyword.

```
# no randomly placed proteins within 10 nm of position (-3, 5, 2.3)
excludedVolume  -3.5 5.0 2.3  10.0
```

Obviously, such a definition of a reserved region can only be considered in later startWithIn lines.

A molecule placed with "startWith" or "startWithIn" can be immobilized for a specified time so that it does not move during, e.g., an intial equilibration phase. It is however included in the calculation of the interparticle forces. The immobilization time is specified as an additional optional number after the exclusion radius (thus, an exclusion radius has to be specified in this case).

```
# place a protein and make it sit until t = 100000 ps
startWith  largeProt  0.0 0.0 0.0  0.0 1.0 0.0 0.0  12.5  100000
```

Output of Reservoir Densities and Protein Numbers

Reservoir densities and protein numbers can be printed into an additional output file, which has to be defined via the output keyword. Then, reservoir densities and particle numbers can be given:

```
# output of reservoir densities
outputfile    untenraus.txt
density       UntenRaus
number        Punkt
```

The reservoir names are the labels used above and the protein names are the ones that finally appear in the output file, i.e., the ones in the protein definition file. Note that first all densities are given in the output and then all particle numbers.

Protein Definition File

A protein definition file resembles the hierarchical structure of the proteins. Each section starts with a certain keyword and ends with a corresponding "End" tag. Indentation is optional, but may increase readability.

The outermost level starts with the Protein keyword and a label for the protein. This label is then used in the output file to denote the type of protein. This line is followed by the definition of the origin of the protein's coordinate system and an effective CM diffusion coefficient needed for the injectors of the reservoirs.

```
Protein Polymer
   position 0.0 0.0 0.0  0.0 1.0 0.0 0.0
   # eff. translational CM diff. coeff., required for molecule injector
   D0 1e-4  # [nm^2/ps]
```

Bonds

If the protein contains harmonic bonds between the individual Gestalt objects, these are defined next. Each bond is defined with a label for further reference within the scope of its protein, its type, and some context dependent parameters. The most basic type is a harmonic spring defined by its length and the spring constant in units of kJ Mol^{-1} nm^{-2}. Also available are Coulomb type bonds and vdW bonds. For Coulomb bonds, the first parameter is the product of the two charges divided by the local dielectric constant, the sum of the two shielding radii, and an inverse Debye length κ. vdW bonds are specified with the same parameters as for the global vdW definitions (except for the colors which are not required here). Thus, each bond gets its own set of interaction parameters.

```
   Bonds Federn     # the Bonds-label is a actually a dummy argument :-)
      # Label type  parameters...
      # harmonic:    length spring-const
      B12a   HO  0.38    12000
      # Coulomb bond:  q1*q2/eps_rel  B12  kappa
      C12b   Coul     -0.756          0.1  0.0
      # vdW bond: LJ vdW-parameters starting from C12, ...
      V13    vdw  $R12  9.745 0.0 0.0  1.0 1.122 -0.141 10.0
      :
   End Bonds    # saying "Bonds" is optional
```

For some applications the protein conformation should be in one of two states like conformational changes or dihedral angles in cis or trans configurations. These can be modelled with springs that have an additional quartic term. These bonds of type "QU" are parametrized in the setup file with the distances of the two minima, x_1 and x_2, and the height of the potential barrier ΔV inbetween (in units of kJ/Mol). The parser converts these data into the respective prefactors for the quadratic and the quartic part of the potential.

```
      # Label  type  x_1  x_2      deltaV
      B03      QU    4.0  8.9443   20.0
```

For this example, the prefactors are $k_2 = –6.545$ and $k_4 = 0.535$ and the potential is centered at a bondlength of $R_0 = 6.472$.

Note that in the bond definitions the label of the bond is case sensitive, but not the type keywords. So, "QU" is eqivalent to "Qu", "qu", or "qU".

Gestalt Objects

Now, the gestalten, which represent the physical objects, can be defined. Analogous to the protein definition, they start with the opening tag plus a label, their position in the protein's ccordinate system, and the parameters for diffusion. These consist of the translational and rotational diffusion coefficients (in units of nm^2/ps and ps^{-1}, resp.), the mass m in kDa, and an effective radius a used to calculate the rotational moment of inertia $L = 2/5\ m\ a^2$. When the mass is set to zero, a concentional overdamped Brownian propagator is

used, while a Langevin propagator is used for $m > 0$. The required damping parameters are determined from the mass and the (globally set) viscosity.

```
Gestalt SER1
   position 1.266 1.187 -0.675  0.0 1.0 0.0 0.0

   # D0 D0_trans  D0_rot   mass     radius
   D0   9.42E-4   0.01361  0.10509  0.23
```

Inside the gestalt, the various shapes can be defined. All of them are optional. When the above Gestalt section is closed with an End tag, a diffusing particle without any interactions is defined. Every shape gets a label which is used for diagnostic information during the setup and later when the proteins are dumped.

The BondShape references the bonds defined above and defines where they are attached relative to the gestalt's center.

```
BondShape SER1-Hooks
   # position              Bond label
   -0.0323 0.0023 0.0030   B12a
   0.0000 0.0000 0.0000    B12b
   :
End
```

The VdwShape defines one or more vdW spheres, i.e., their position relative to the gestalt's origin, their radius, and their vdW color. Again, any text after End is ignored.

```
VdwShape SER1-vdw
   # pos            r      color
   0.00 0.10 0.00   0.371   0
End SER1-vdw
```

The charges are defined analoguously. They are characterized by their charge (in unit charges) and the shielding radius.

```
EstatShape SER1-E
   # position             charge radius
   -0.016 -0.132 0.069     0.51  0.2
   -0.109 -0.213 0.072    -0.51  0.2
   0.107  0.159 -0.124    -0.66  0.177
End
```

Sticky spheres are defined via a StickyShape. For each of the spheres you have to give the center position, two radii, and a color index. The first radius is used when there is no transient bond yet, i.e., when stiction should set in, while an existing, but just broken bond can be re-established within the second radius.

```
StickyShape Patch
   # position   R_in  R_out  color
   0.0 0.0 0.0  4.0    4.5     0
End
```

Hydrogen bonds can be defined between a donor hydrogen atom and an acceptor atom (note that this is different from the usual definition of an H shared between two N or C atoms). Both the donor and the acceptor are defined by their position and a vector pointing towards the virtual hydrogen atom sitting halfway between donor and acceptor. For a weak donor (acceptor) the multiplier value should be smaller than one, while a value larger than one indicates a strong donor (acceptor).

```
HBondShape HB
```

```
   # keyword   donor position   hydrogen position  scaling factor
   donor       1.8 0.0 0.0      2.0 0.0 0.0        1.0
   # keyword   acceptor pos.    hydrogen pos       scaling factor
   acceptor    -1.4 0.7 0.5     -1.5  0.65  0.5    1.1
End
```

The HI shape may only define a single HI sphere, which is fixed around the gestalt's center. Therefore, the only parameter is its radius. Note that a radius of zero does not mean that there is no HI but that then effectively the Oseen tensor is used for this shape.

```
HiShape SER1-HI
   0.17
End
```

In order to apply external forces to a Gestalt, hook-up points have to be defined via an ExternalShape object. Each hook-up point is defined via its position, the label of the external force as defined in the main setup, and a weight α. The following definition with the fields defined as above keeps the particle centered around the origin via the harmonic constraints of "Well" and its x-axis aligned parallel to the linear field "towardsX".

```
ExternalShape Orienter
   # position     fieldLabel   weight
   2.0  0.0 0.0   towardsX      1.0
   -2.0 0.0 0.0   towardsX     -1.0
   0.0 0.0 0.0    Well          1.0
End
```

To simplify the analysis arbitrary points of the Gestalt can be marked with "MarkerShape" objects. These are ignored during the simulation and only used in the extractor alternatively to the vdW shapes. They are defined with a position and an index, which is used later together with the MarkerShape's label to identify the points. The index can also be a text string without spaces, i.e., a "word"

```
MarkerShape Marks
   0.0 0.0 0.0  1
   1.85 0.0 0.0  two
End
```

After the closing End tag for the Gestalt, more Gestalten within the same protein may follow to build up for example a bead-spring polymer or a flexible protein of multiple rigid parts. A closing End then finalizes the protein.

Scene Definition File

The scene definition file specifies the walls around the simulation box. Its structure is similar to the protein definition files (actually, the same parser is used for both). A wall definition starts with the Wall keyword plus a label, followed by the wall's position. A wall is initially created in the *y*-*z*-plane at $x = 0$, its normal thus points into the positive *x*-direction. The final position and orientation of the wall has to be given relative to this initial setting. Note that walls are mathematical planes, i.e., they are infinitely large.

```
# bottom of the box in the y-z-plane at x = -20
Wall bottom
      position  -20.0 0.0 0.0  0.0 1.0 0.0 0.0
```

If this wall shall be a basic reflecting wall, only the closing End tag has to be added. Here we assume that the wall interacts with the diffusing particles via a vdW potential. We then need a gestalt with its position

relative to the wall and a VdwShape. For a wall, the normal of the vdW surface is the same as for the wall itself, thus only the relative displacement along this normal and the color have to be specified. However, a wall may also contain vdW spheres, which are defined as in a protein.

```
   Gestalt bottom
      position 0.0 0.0 0.0  0.0 1.0 0.0 0.0

      VdwShape vdwUnten
         # type:wall  offset  color
         wall         -0.3     0
         # also possible: a vdW sphere attached to the wall
         # position   radius  color
         0.0 0.0 0.0  4.0     0
      End
   End
```

This gestalt may also contain charges in the same way as a protein (not shown here).

A vdW shape specific for a wall is a so called "block" with a centered cylindrical hole along the x-axis. It is defined via its position (including orientation), its extension along the three directions, the radius of the cylindrical hole, and its vdW color. The following code defines a quadratic slab of area 60 x 60 nm^2 with a thickness of 4 nm and four holes at (y, z) = (±15, ±15) nm. This slab is centered at the origin and not rotated. When the side walls of the simulation box are placed at y = ± 30 nm and z = ± 30 nm, this slab is a barrier with four holes. If the side walls are further away, the proteins may circumvent the slab.

```
Wall Mitte
   position 0.0 0.0 0.0   0.0 1.0 0.0 0.0

   Gestalt Block
      position 0.0 0.0 0.0   0.0 1.0 0.0 0.0

      VdwShape vdwBlock
         # block  x y z          lx ly lz      cyl_radius  color
         block   0.0 -15.0 -15.0   2.0 15.0 15.0    10.0    0
         block   0.0 -15.0  15.0   2.0 15.0 15.0    10.0    0
         block   0.0  15.0 -15.0   2.0 15.0 15.0    10.0    0
         block   0.0  15.0  15.0   2.0 15.0 15.0    10.0    0
      End
   End
End
```

To allow for particle exchange and interfacing to reservoirs, injectors and acceptors are used. They are defined with the respective keywords and a label. They both require a definition of a rectangular mask with the mask's center and how far it extends from there. These coordinates are defined relative to the wall's internal coordinate system. Then, one (or more) molecules are assigned to injector and acceptor via the labels from the protein configuration file. A top wall in a cubic simulation box centered around the origin and interfacing to a constant density region at $x > 20$ nm would be defined as follows:

```
# top: y-z-plane at x = +20, normal pointing towards -x
Wall Oben
   # move to (20, 0, 0) and rotate by 180 degrees around y-axis
   position  20.0 0.0 0.0  180.0 0.0 1.0 0.0

   Injector Rein
      # Mask  Center     extension (+-ly +-lz)
```

```
      Mask    0.0  0.0   20.0  20.0     # mask area is 40x40 nm2 = 1600 nm2
      # Molekule  Name   Reservoir
      Molecule    Punkt  PunktRein
   End

   Acceptor Raus
      # Mask  Center  extension (+-ly +-lz)
      Mask  0.0  0.0   20.0  20.0
      # Molekule  Name  Reservoir
      Molecule    Punkt PunktRaus
   End
End
```

Output File Format

The raw output of the simulation consists per output step of a time line with the actual simulation time and the number of Gestalt objects in the simulation. This line is followed by one line per Gestalt, which lists the label from the protein definition file, the numerical ID of the protein (assigned at insertion into the cup), and the Gestalt label from the protein definition. These are followed by the Gestalt's position and orientation (rotation angle plus vector for the rotation axis). A typical output at a given timestep with two proteins, a NA and a CL ion, would like this:

```
timeline    4.000    2
CL    1     CL_Bead     0.899 0.175 -0.177      0.000 1.000 0.000 0.000
NA    2     NA_Bead     -1.029 0.130 0.137      0.000 1.000 0.000 0.000
```

Changelog:

Aug. 30, 2012: V1.7

* added description for nano-bead potential and linear potential in vdW shapes

* new description of stickyShape

Dec. 14, 2011: V1.6

* molecules can be immobilized in startWith and startWithIn

June 15, 2011: V1.6

* new bond types: Coulomb and vdW

* new H-bonds

* parser keywords: echo, evaluate, repeat/endrepeat

* new keyword "startWithIn"

Apr. 11, 2011: V1.5

* definition of constants on the commandline

* keywords are case-insensitive

Oct 7, 2010: V1.4

* added explanation of ExternalShape objects of the proteins