Tremolo-X Manual[1]
(Version 1.6)

Fraunhofer SCAI

For questions, errors and updates please contact:
Christian Neuen
christian.neuen@scai.fraunhofer.de

Contributors:
Gregor Bollerhey
Dr. Jan Hamaekers
Christian Neuen
Ralph Thesen

Thanks go to:
Lukas Jager
Ralf Wildenhues

January 9, 2014

TODO: Recompile with frutiger font if available!

Contents

Chapter 1

Overview

TODO: Check urls in bibliography (obtained from (very) old comments!!!

This is a user manual into the TREMOLO-X Molecular Dynamics simulation package. Should you be interested in an in-depth introduction into the numerics of molecular dynamics simulation, you might want to read [36]

For up-to-date news and other interesting information about the TREMOLO-X software, please visit:

`www.tremolo-x.com`

Tremolo-X is a molecular dynamics simulation package. The program consists of two major components, which allow either sequential or parallel computation of the simulation.

Tremolo-X supplies a choice of several common bonded and non-bonded potentials, which can be combined in order to simulate systems which are dependent on intra- and intermolecular forces.

You may choose from a variety of statistical ensembles and boundary conditions, in order to tailor the simulation to your particular settings and needs.

Furthermore, the dynamic simulations may be augmented or replaced with energy optimization routines.

1.1 How to read the manual

This manual begins with a descriptions of the different capabilities (ensembles, integrators, thermostats,...) of the program, and their interdependencies. We tried to add as much mathematical detail as necessary, but as little as possible, however we always indicate a source, where you can find more details, when you are interested beyond the mere usability of the product. We continue with an in-depth description of the different potentials you can specify in Tremolo-X and finish with a detailed specification the syntax of options in the parameter file.

Chapter 2

Configuration and Setup

TODO: How will the setup work for the enduser?

Chapter 3

First steps

The first section of this chapter is identical to the Quickstartguide provided next to the manual. For a detailed description please read the rest of the manual. The second section mirrors the first, giving an example for the use of the parallel version.

3.1 First sequential steps

Switch to the Tremolo-X folder and enter the `example/Argon/` folder therein. Have a glance at the files currently present in the directory. Now type

```
tremolo argon.tremolo
```

Congratulations! You are running your first Tremolo-X simulation!

Now that did not look like there was much happening, did it? So let's try that again and have tremolo generate some more output:

```
tremolo -v argon.tremolo
```

While some of the information is self explanatory, some is not, so if you want to know about everything, have a look at the manual!

As soon as the program is finished and console command has been returned to you, have another look at the content of the directory. The additional files contain the measurements of Tremolo-X. The files argon.e* contain information about the energy in the ensemble, e.g. in `argon.etot` you find the **total energy**, which remained constant through the simulation (of course there are some oscillations due to numeric effects). Similarly you find the **potential energy** in `argon.epot` and the **kinetic energy/temperature** in `argon.ekin`.

Here, as in most output files, columns are printed for different particle groups, so if you have more than one type of particle in your simulation, more columns with the respective values will appear.

Opening the .msd file you find several columns which contains data towards the computation of the **diffusion coefficient(s)**, in particular in the column labeled *Dif_sum*. (Note that the numeric values are computed in reduced units, please refer to the manual.)

In most cases it is beneficial to look at a graph, instead of those long columns of numbers. If you have no favorite plotting program of your own, you could try gnuplot[1], it is free and fairly easy to use.

The files `argon.vis.####.[xyz,pdb,data]` contain the **positions** (and also other data) of the atoms at specific time points in the simulation, e.g. in `argon.vis.0100.[xyz,pdb,data]`, you find the positions at the end of the simulation.

As before you might prefer to look at pictures instead of the numbers, so if you do not have a favorite program for visualizing .xyz or .pdb output, you could try vmd or ovito[1], as it is also free and easy to use.

3.2 First parallel steps

Switch to the Tremolo-X folder and enter the example/Argon folder therein. Have a glance at the files currently present in the directory. Now type

```
mpirun -np 8 tremolo_mpi -p 2,2,2 argon.tremolo
```

Congratulations! You are running your first parallel Tremolo-X simulation!

As before, you will obtain more output with the verbosity option by typing

```
mpirun -np 8 tremolo_mpi -p 2,2,2 -v argon.tremolo
```

As soon as the program is finished and console command has been returned to you, have another look at the content of the directory. The additional files contain the measurements of Tremolo-X. The files argon.e* containing the energy measurements appear as they did in the serial case and are organized in the same way.

What changed are the output files for the visualization (.xyz and .pdb) and the data state file (.data.9999). Instead of `argon.vis.####.xyz` (`argon.vis.####.pdb`) you find `argon.vis.####.####.xyz` (`argon.vis.####.####.pdb`), similarly you see `argon.data.9999.####` files. This is due to the fact that all processes write output for their segment simultaneously to speed up the process. In order to combine these segments, an easy to use tool is provided in the utility section of tremolo.

Typing

[1] Neither gnuplot, nor vmd, nor ovito are developed, maintained, distributed or in any other way associated with Fraunhofer SCAI.

```
/PATH_TO_TREMOLO_BINARIES/MergeOutput.py argon xyz
```

will combine the separate xyz files and produce a unified one for the complete domain of the simulation. Adding the option `-d` will cause the original, partial files to be deleted. Similarly `pdb` and `data` files can be merged by typing

```
/PATH_TO_TREMOLO_BINARIES/MergeOutput.py argon pdb
```

or

```
/PATH_TO_TREMOLO_BINARIES/MergeOutput.py argon data
```

respectively. For more options type

```
/PATH_TO_TREMOLO_BINARIES/MergeOutput.py -h
```

Visualization of those files functions exactly as described for the serial case. (For the use of the `.data.9999` file please refer to the manual.)

Chapter 4

Running Tremolo

Tremolo-X allows two distinct modes of operation: Control from a command line, which allows the user to take advantage of scripts for task automation, and control from a graphical user interface, which provides an interactive environment and relieves the user of the burden to handle the parameter syntax.

4.1 Command line

Assuming, the Tremolo-X installation path belongs to the $PATH environment of your system, the most basic command to call (serial) tremolo is

```
tremolo project.tremolo
```

In order to call the parallel version, the call is modified to

```
tremolo_mpi project.tremolo
```

Both versions share the following parameters:

```
Usage: tremolo [-hvx ] [-a alarmtime] [-d debugstr|-D \
↪   debugstr] [-n nicelevel] [-o verbosity] [-s \
↪   sleeptime] [-f] mainparameterfile
 -a alarmtime  Sets alarm to alarmtime seconds. Code will \
 ↪   finish after next timestep, writing out the current \
 ↪   state.
 -f mainfile   Specify main parameter file (last \
 ↪   argument: '-f' may be omitted)
 -h            Displays this help page and exits \
 ↪   successfully.
 -n nicelevel  Decreases priority of process.
 -v            Increases verbosity level by one. Default \
 ↪   value: 0.
 -V            Print version.
```

Only for developers and in developer version the following options are accepted: `-d debugstr Starts debugger right away.`

`-D debugstr Starts debugger when encountering an error. Valid syntax for debugstr: 0 sleep for a minute, no debugger [host[:display][,debugger start debugger xterm on host:display (if DISPLAY is set or for local connection set host to local). Valid values for debugger are: gdb, dbx, ddd, cvd, totalview, ups.`

`-s sleeptime wait sleeptime seconds for debugger to come up (-d, -D)` *TODO: Create new lst listing environment which is sensitive to the draft option*

TODO: detail verbosity syntax

while the parallel version takes one additional parameter,

```
-p px,py,pz   Start parallel code with domain \
↪   decomposition,
              using px slices on x-axis, py slices on \
              ↪   y-axis and pz slices on z-axis \
              ↪   (total: px*py*pz processes)
```

which must be supplied to actually benefit from parallel execution, since the default would be one process. However, Tremolo-X will always check whether the number of processes in the MPI-call matches the number of sub domains in the domain decomposition and will throw an error if this is not the case.

For parallel codes it may be necessary to specify the main parameter file with an absolute path.

4.2 Using the GUI

TODO: separate instructions for the GUI?

Chapter 5

Domain and Boundaries

The first step of setting up a Molecular Dynamics simulation is to decide and specify the domain of the system. The domain is determined by its shape and the type of boundary, e.g. what happens, when a particle crosses the boundary domain. Several choices are available in Tremolo-X which, depending on the setup desired, can be combined freely. The shape of domain used determines the type of input required:

- A *cube*(=`cube`) requires only one parameter, an edge length.
- A *cuboid*(=`diag`) requires three edge length.
- A *parallelepiped*(=`matrix`) requires its complete matrix, i.e. the collection of the three edge vectors. The vectors are written column wise in the matrix, i.e. each column corresponds to one box-vector. (xx, xy and xz are first vector and so on).

Note that one can also give the box matrix by a `# BOX` line in the `.data` file. In particular, a `# BOX` line given in the `.data` file will overwrite values from the `.parameter` file.

The boundary conditions have a direct correspondence to the simulated physical environment of sample.

- *Reflecting* boundary conditions are self-describing, momentum is preserved.
- *Leaving* boundary conditions allow particles to leave the domain and thus remove them from the simulation. Note that leaving boundary conditions destroy the conservation of particles and thus may violate the statistical ensemble of your choice.
- *Periodic* boundary conditions transfer particles to the respective opposite face on the same axis, preserving momentum and direction.

Chapter 6

Ensembles and Thermostats

Ensembles can be chosen from the set {NVE, NVT, NPT, NPE} *TODO: check NPE* The letters specify a conservation value, which in combination define the statistical ensemble to be simulated.

N = Number of particles
V = Volume of domain
E = Energy (total = kinetic + potential)
T = Temperature (kinetic energy)
P = Pressure

The choice of ensembles which conserve temperature and/or pressure require the use of a thermostat and/or barostat. Furthermore, note that if you choose leaving boundary conditions for at least one boundary, the conservation of particles and thus the chosen statistical ensemble may be violated.

6.1 Propagators

The simulation of particle motion may be executed with several different propagation algorithms. In particular, Tremolo-X supports the choice of:

- verlet = Verlet algorithm
- {beeman2, beeman3, beeman4, beeman5} = Beeman-Verlet algorithm of order 2–5.
- {beeman2v, beeman3v} = Velocity-Beeman-Verlet algorithms with order 2 or 3.

Note, that the use of some thermostats may demand or prohibit the choice of some propagators: The Berendsen thermostat does not work with {beeman2v, beeman3v}, whereas the Nose-Hoover thermostat works only with those two.

The parameter `delta_T` fixes the time step length in units specified in the .tremolo file.

The `endtime` defines the duration of the simulation, again in units specified in the .tremolo file.

TODO: timeinteps and maxiteration

6.2 Thermostats

When running a simulation with an ensemble which specifies T, the use of a thermostat is required. Tremolo-X supplies several thermostats to use, which fit different demands. All thermostats use the relation between kinetic energy of the particles and temperature:

$$E_{kin} = \sum_{i=1}^{N} \frac{m_i}{2} \vec{v}_i^2 = \frac{3N}{2} k_B T$$

$$T = \frac{2}{3N k_B} \sum_{i=1}^{N} \frac{m_i}{2} \vec{v}_i^2$$

Thus the temperature of the ensemble is controlled by influencing the individual particles. Several different schemes for influencing particles are available, which have very different properties and corresponding advantages and disadvantages.

6.2.1 Berendsen Thermostat

The Berendsen thermostat works by direct scaling of all velocities. The scaling factor is determined as

$$\beta := \sqrt{E_{kin}^D / E_{kin}} = \sqrt{T^D / T},$$

where T_D is the desired value, while T is the currently measured value. The scaling of the velocities is then

$$\vec{v}_i^{new} = \beta \vec{v}_i^{old}.$$

This scaling is performed after a time-interval (specified in parameter T_Interval) has passed. Time intervals larger than the integration timestep allow the ensemble to achieve some equilibration between the potential and kinetic energy. At the correction point, the velocities of all particles are scaled, so that the kinetic energy exactly matches the specified temperature.

Note that the Berendsen thermostat does not work in combination with propagators of the type Velocity-Beeman-Verlet.

This thermostat does not conserve an abstract energy definition. While this thermostat does conserve the total momentum of the system, the angular momentum will not be preserved.

6.2.2 Nose Thermostat

TODO: not implemented yet

6.2.3 Nose-Hoover Thermostat

This thermostat uses a virtual heat bath, coupled to the system via a non-physical traction term. This acts as an additional force term in the equations of motion:

$$\begin{aligned}\dot{\vec{x}}_i &= \vec{v}_i,\\ m_i\dot{\vec{v}}_i &= \vec{F}_i - \xi m_i \vec{v}_i.\end{aligned}$$

The magnitude of the traction term ξ is determined by a differential equation, which ties it to the difference between the current and desired temperature.

$$\frac{d\xi}{dt} = \left(\sum_{i=1}^{N} m_i \vec{v}_i^2 - 3Nk_BT^D\right)/M$$

The strength of this coupling is determined by the parameter M, called the Hoover-Mass. This thermostat does not cause instantaneous effects, it rather allows (and by design causes) "overshooting" the desired temperature, leading to an oscillating kinetic energy. The choice of the Hoover mass has a substantial impact on the behavior of the thermostat, however to the best of the authors knowledge there is not precise method to help in the choice of this constant. While it can be derived from the differential equation, that the mass should scale with the total mass of the ensemble, there are indications that target temperature and ensemble pressure have an influence as well. Thus it is necessary to make educated guesses or find suitable values in the reports of prior, similar experiments.

On a good note, this thermostat does conserve an abstract energy, the so called Hoover energy.

The Nose-Hoover thermostat only works in combination with propagators of the type Velocity-Beeman-Verlet.

6.2.4 DPD Thermostat

TODO: DPD

6.3 Parrinello-Rahman(-Nosé) Barostat

6.3.1 Idea

The pressure and temperature should be controlled by the use of additional degrees of freedom.

We use a scaling of the coordinate vector:

$$\vec{r}_i = h\vec{s}_i \tag{6.1}$$

where $h = [a_0, a_1, a_2]$ is a 3x3-matrix, with basis vectors a_0, a_1, a_2 of the periodic simulation domain (volume: $\Omega = \det h$) and consequently we have $s_i \in [0, 1)^3$.

*TODO: What is alpha for scaled velocities: Dependence on h?*Scaled velocities:

$$\dot{\vec{r}}_i' = \alpha\dot{\vec{r}}. \tag{6.2}$$

The matrix h and the variable α will be used control the pressure and temperature of a suitably extended system. Towards this purpose we use the fictitious (non-physical!) potentials of the thermodynamic variables P and T, defined as

$$\begin{aligned} V_P &= P \det h \\ V_T &= \frac{g}{\beta} \ln \alpha \end{aligned} . \tag{6.3}$$

Here, $P = P_{\text{ext}}$ is the external system pressure, $\beta = k_B T$ and g is a constant corresponding to the number of degrees of freedom of the system.

6.3.2 Parrinello-Rahman-Nosé Hamilton-function

Definition: The extended Lagrange function of a NPT system is defined as follows:

$$\mathcal{L} = \frac{1}{2}\sum_i^N m_i \alpha^2 \dot{\vec{s}}_i^T h^T h \dot{\vec{s}}_i + \frac{1}{2} W \alpha^2 \text{tr}(\dot{h}^T \dot{h}) + \frac{1}{2} Q \dot{\alpha}^2 - V - P_{\text{ext}}\Omega - \frac{g}{\beta}\ln\alpha, \tag{6.4}$$

where m_i is the mass of the i^{th} particle, Q the fictitious mass of the Nosé-Thermostat and W the fictitious mass of the simulation cell (barostat).

Thus the following conjugated momenta can be extracted:

$$\begin{aligned} \vec{T}_i &= m_i G \alpha^2 \dot{\vec{s}}_i, \quad G = h^T h \\ P_h &= \alpha^2 W \dot{h} \\ p_\alpha &= Q\dot{\alpha} \end{aligned} \tag{6.5}$$

An from those we obtain the Hamiltonian:

$$\mathcal{H} = \frac{1}{2}\sum_i^N \frac{\vec{T}_i G^{-1} \vec{T}_i}{m_i \alpha^2} + \frac{1}{2}\frac{\text{tr}(P_h^T P_h)}{\alpha^2 W} + \frac{p_\alpha^2}{2Q} + V + P_{\text{ext}}\Omega + \frac{g \ln \alpha}{\beta} \tag{6.6}$$

To obtain the corresponding equations of motion one has to deal with varying timesteps. Thus the following relations are used:

$$\begin{aligned} \vec{T}_i &\rightarrow \frac{\vec{T}_i}{\alpha} \\ P_h &\rightarrow \frac{P_h}{\alpha} \\ p_\alpha &\rightarrow \frac{p_\alpha}{\alpha} \end{aligned} \tag{6.7}$$

Furthermore, we introduce the new conjugated momenta

$$\vec{p}_i \equiv G^{-1}\vec{T}_i \tag{6.8}$$

in order to write

$$\dot{\vec{s}}_i = \frac{\vec{p}_i}{m_i} \tag{6.9}$$

for the corresponding velocities (no knowledge of h required).
Consequently

$$\begin{aligned} t &\rightarrow \frac{t}{\alpha} \\ \eta &\equiv \ln\alpha. \end{aligned} \tag{6.10}$$

Also,

$$\mathcal{H} = \frac{1}{2}\sum_i^N \frac{\vec{p}_i G^{-1}\vec{p}_i}{m_i} + \frac{1}{2}\frac{\mathrm{tr}(P_h^T P_h)}{W} + \frac{p_\eta p_\eta}{2Q} + V + P_{\mathrm{ext}}\Omega + gk_BT\eta \tag{6.11}$$

Thus we obtain the equations of motion:

$$\begin{aligned} \dot{\vec{s}}_i =& \frac{\vec{p}_i}{m_i}, \quad \dot{h} = \frac{P_h}{W}, \quad \dot{\eta} = \frac{p_\eta}{Q} \\ \dot{\vec{p}}_i =& h^{-1}\vec{f}_i - m_i G^{-1}\dot{G}\vec{p}_i - \frac{p_\eta}{Q}\vec{p}_i, \\ \dot{P}_h =& \mathcal{V} + \mathcal{K} - h^{-T}P_{\mathrm{ext}}\det h - \frac{p_\eta}{Q}P_h, \\ \dot{p}_\eta =& \sum_i^N \frac{\vec{p}_i^T\vec{p}_i}{m_i} + \frac{\mathrm{tr}(P_h^T P_h)}{W} - gk_BT \end{aligned} \tag{6.12}$$

where

$$\begin{aligned} f_i =& -\nabla_{r_i} V \\ \mathcal{V} =& \sum_i^N \vec{f}_i\vec{s}_i^T \\ \mathcal{K} =& \sum_i^N m_i h\dot{\vec{s}}_i\dot{\vec{s}}_i^T. \end{aligned} \tag{6.13}$$

For the pressure tensor and thus the pressure we have:

$$\begin{aligned} \Pi_{\text{int}} =& \frac{1}{\Omega} \sum_i^N \left(\frac{\vec{p}_i \vec{p}_i^T}{m_i} + \vec{f}_i \vec{r}_i^T \right) \\ P_{int} =& \frac{1}{3} \text{tr}(\Pi_{\text{int}}) \end{aligned} \tag{6.14}$$

specifically:

$$\frac{1}{\Omega}(\mathcal{V} + \mathcal{K}) h^T = \Pi_{\text{int}} \tag{6.15}$$

holds.

6.3.3 Implementation

The equations of motion (6.12) can be implemented as usual, but one has to pay attention to the propagator, since:

$$\begin{aligned} \dot{\vec{p}}_i =& h^{-1} \vec{f}_i - G^{-1} \dot{G} \dot{\vec{s}}_i - \frac{p_\eta}{Q} \dot{\vec{s}}_i \quad \text{oder} \\ \ddot{\vec{s}}_i =& \frac{1}{m_i} \left(h^{-1} \vec{f}_i - G^{-1} \dot{G} \dot{\vec{s}}_i - \frac{p_\eta}{Q} \dot{\vec{s}}_i \right) . \end{aligned} \tag{6.16}$$

Also, $\ddot{\vec{s}}_i(t)$ depends on $\dot{\vec{s}}_i(t)$. This problem can be solved by using an iterative variant of the Beeman algorithm (3^{rd} order).

Iterative Beeman

$$i) \quad x(t+\delta t) = x(t) + \delta t \dot{x}(t) + \frac{\delta t^2}{6} \left[4\ddot{x}(t) - \ddot{x}(t-\delta t) \right] \tag{6.17}$$

$$ii) \quad \dot{x}^{(p)}(t+\delta t) = \dot{x}(t) + \frac{\delta t}{2} \left[3\ddot{x}(t) - \ddot{x}(t-\delta t) \right] \tag{6.18}$$

$$iii) \quad \ddot{x}(t+\delta t) = F\left(\{ x_i(t+\delta t), \dot{x}_i^{(p)}(t+\delta t) \}, i = 1 \dots n \right) \tag{6.19}$$

$$iv) \quad \dot{x}^{(c)}(t+\delta t) = \dot{x}(t) + \frac{\delta t}{6} \left[2\ddot{x}(t+\delta t) + 5\ddot{x}(t) - \ddot{x}(t-\delta t) \right] \tag{6.20}$$

$$v) \quad \text{Replace } \dot{x}^{(p)} \text{by } \dot{x}^{(c)} \text{ and go to iii} \tag{6.21}$$

For each cycle of the iteration the only change for the computation of forces in step iii is the velocity, thus the (usually expensive) computation of the gradient of the potential needs to be performed only once. A break criteria can be constructed by the relative changes of the velocities (e.g. from experience: 10^{-7} yields 2 to 3 cycles).

Constraints on h

The nine degrees of freedom of h are unphysical, at least rotations should be excluded.

Naive Fix the entries of the force acting on h ($=F_h$) to zero below the main diagonal. This corresponds to a virtual constraining force to avoid rotations.

Symmetric Only allow symmetric contribution of F_h, this is enforced by setting $F_h^S = \frac{1}{2}(F_h + F_h^T)$.

Isotrop 5 constraints

$$\frac{h_{\alpha\beta}}{h_{11}} - \frac{h^0_{\alpha\beta}}{h^0_{11}} = 0 \quad \alpha \leq \beta \tag{6.22}$$

and corresponding changes for the velocities (e.g. for the use of RATTLE).

$$\dot{h}_{\alpha\beta} - \frac{h^0_{\alpha\beta}}{h^0_{11}}\dot{h}_{11} = 0. \tag{6.23}$$

Due to the symmetry of the stress-tensor only one degree of freedom remains, which can be interpreted as a volume. For the reference matrix h^0 the initial matrix is chosen.

TODO: Man kann nun damit nicht nur Π_{int11} einen Einfluss hat, die Kraft die auf h_{11} wirkt gleich dem Druck setzen. (Analog zu Andersen). Dies machen wir in unserem Code.

6.3.4 parameters

The above description is reflected in the following parameters: *TODO: experiment with constraints.*

```
parinello:        state={on, off},        f_mass=#FLOAT;
constraint:       type=none;
constraintmap:    xx=1, xy=1, xz=1, yx=1, yy=1, yz=1, zx=1, \
↪    zy=1, zz=1;
constantpressure:          state={on | off}, Pressure=#FLOAT;
Timeline: state={on | off}, [time,  pressure, \
↪    interpolation=(0, 0, constant)];
stresstensor: [time, stress, interpolation,  xx, xy, xz, \
↪    yy, yz, zz=(0, 0, constant, 0, 0, 0, 0, 0, 0)];
```

TODO: timeline problems

6.3.5 Note

TODO: move to appendix

Molecule Scaling

The described and implemented approach is the so called *atomic scaling*. In addition there is another variant called *molecule scaling*. In the following equations i is the index of a molecule, while k is the index of an atom (belonging to the k^{th} molecule). R_i are the coordinates of the *center of mass* and S_i the respective scaled coordinates. w_{ik} specifies the relative coordinates of the atom k with respect to the center of mass of molecule i.

$$\begin{aligned} r_{ik} =& R_i + w_{ik} = hS_i + w_{ik} \\ \dot{R}_i' =& \dot{R}_i \alpha \\ \dot{w}_i' =& \dot{w}_i \alpha \end{aligned} \tag{6.24}$$

From those equation corresponding Lagrange- and Hamilton functions with their respective equations of motion can be derived. (Those are are essentially extended by „w-terms“. This approach is not implemented, since it would interfere with efficient parallelization.

It can be shown that both approaches display the same statistical behavior of pressure and stress.

Degrees of Freedom

atomic: $N_f = 3N$, where N is the number of atoms. Then the following holds:

- *nonperiodic:* $N_f = 3N - 6$ since translation and rotation of the *center of mass* may be ignored.
- *periodic:* (as in the Parrinello-Rahman case (else this would be insensible)) $N_f = 3N - 3$ since only translation may be ignored.

Temperature

$$T_{instan} = \frac{2K}{N_f k_B}, \quad K = \frac{1}{2} \sum_i^N \frac{\vec{p}_i^T \vec{p}_i}{m_i}$$

The T from (6.11) is the desired temperature.

Fictitious Masses

Educated guessing! Compare to other successful experiments. However there is very little information on this particular aspect.

6.3.6 Euler-Lagrange and Hamilton equations

Only as a reminder *TODO: Move to appendix?*

Newton

$$\begin{aligned} F &= \frac{dp}{dt} \quad \text{with} \\ p &= m\dot{q} \quad \text{and with} \quad \dot{m} = 0 \\ F &= m\ddot{q} \end{aligned} \tag{6.25}$$

Lagrange

$$\begin{aligned} L(q,\dot{q}) &= T(\dot{q}) - V(q) \\ \frac{\partial L}{\partial q} &= -\frac{dV}{dq} \\ \frac{\partial L}{\partial \dot{q}} &= -\frac{dT}{d\dot{q}} = m\dot{q} = p \end{aligned} \tag{6.26}$$

Euler-Lagrange equations of motion

$$\begin{aligned} \frac{d}{dt}\left(\frac{L}{\dot{q}}\right) &= \frac{\partial L}{\partial q} \quad \text{or} \\ \dot{p} &= \frac{\partial L}{\partial q} \end{aligned} \tag{6.27}$$

Hamilton

$$\begin{aligned} H(p,q) &\equiv p\dot{q} - L(q,\dot{q}) \quad \text{with equation of motion} \\ \frac{\partial H}{\partial p} &= \dot{q} \quad \text{and} \\ -\frac{\partial H}{\partial q} &= \dot{p} \quad \text{furthermore} \\ \frac{\partial H}{\partial q} &= -\frac{\partial L}{\partial q} \end{aligned} \tag{6.28}$$

For $T = \frac{1}{2}m\dot{q}^2$ and $V \neq V(\dot{q})$ follows $H = T + V$.

6.4 Hybrid Monte Carlo

6.4.1 Idea

Due to numerical instabilities the integration step size in a conventional molecular dynamics (MD) simulation cannot be chosen arbitrarily large. *Hybrid Monte Carlo (HMC)* is a method to sample the NVT ensemble that applies a Metropolis acceptance-rejection criterion to integration steps such that the step size can be increased to some degree. Nevertheless reasonable step sizes are bounded above as higher step sizes result in higher rejection rates. Note that HMC does not require a thermostat.

The main advantage of HMC is that a larger region of phase space may be sampled with respect to the time the simulation consumes than a conventional MD simulation may do. However, it needs fine tuning of the parameters to profit from this advantage.

To ensure that HMC indeed samples the NVT ensemble the chosen integrator must be *time reversible* and *area preserving*. The most prominent example of such an integrator is the Verlet integrator.

In general the Metropolis criterion is not applied to a single integration step but to a moderate number (such as 10) of steps that shall be referred to as a Monte Carlo (MC) step. Let X and X' and be the states of the simulation before and after an MC step. Then HMC will accept this step with probability given by

$$w\left(X \rightarrow X'\right) = \begin{cases} 1, & \text{if } \Delta \leq 0 \\ \exp(-\Delta), & \text{if } \Delta > 0 \end{cases}, \quad \text{where}$$
$$\Delta = \beta\left(H(X') - H(X)\right),$$

where the inverse temperature β is defined by $\beta = 1/k_B T$, k_B is the Boltzmann constant and $H(\cdot)$ denotes the Hamiltonian or total energy.

If X' is accepted, the coordinates of the configuration will be the input for the next MC Step, while the velocities of the system will be randomly chosen according to the temperature that is supposed to be sampled. Otherwise the simulation will be reset to X and new velocities will be randomly chosen in order to obtain a different result of the MC step which then may be accepted or rejected.

For detailed information see [59] and references therein.

6.4.2 Implementation

Tremolo-X will output measurements only after the acceptance or rejection of MC steps but never in between. Furthermore, if the last MC step was declined, the last MC step to be accepted or in extreme cases the initial data will be written to output.

In addition to the usual output files Tremolo-X will create a .hmc file that contains the acceptance rate during the simulation and mean values as well as the standard deviations of all measured energies and the pressure.

6.4.3 Usage

Parameter file

To use Hybrid Monte Carlo add or edit the following line to/in the `dynamics` section of the .parameter file:

```
hybridmc:    state=on,    N_MD=#INT;
```

`N_MD` is the number of time integration steps per MC step.

6.5 Parallel Tempering

6.5.1 Idea

For many applications it is difficult to sample an NVT ensemble at low temperatures with an MD simulation as the system may easily get trapped in one of many local minimum energy configurations. A way to overcome this problem is *Parallel Tempering (PT)* which is also known as *Replica Exchange Method*.

In a PT simulation N non-interacting replicas (or copies) of one original system are simulated at N different temperatures T_n $(n = 1, \ldots, N)$. With the progress of the PT simulation a replica may be held at different temperatures. Nevertheless there is always exactly one replica at each temperature.

At some points of the simulation replicas are allowed to exchange temperatures such that systems can cross energy barriers at higher temperatures to sample different local minimum energy configurations later on. If temperature exchanges occur, particle velocities are scaled uniformly such that replicas have the designated temperatures. In between the temperatures are held constant by an appropriate thermostat.

To ensure that the PT simulation converges towards an equilibrium distribution, the decision whether temperatures are exchanged is based on a Metropolis condition as follows: Let q_1 and q_2 be two replica configurations which are held at temperatures T_m and T_n, respectively. Furthermore, let X and X' be the state of the overall PT simulation before and after a possible temperature exchange. The probability for this exchange is then given by

$$w\left(X \rightarrow X'\right) = \begin{cases} 1, & \text{if} \Delta \leq 0 \\ \exp(-\Delta), & \text{if} \Delta > 0 \end{cases}, \qquad \text{where}$$

$$\Delta = \left(\beta_n - \beta_m\right)\left(V\left(q_1\right) - V\left(q_2\right)\right),$$

where the inverse temperature β is defined by $\beta = 1/k_B T$, k_B is the Boltzmann constant and $V(\cdot)$ denotes the potential energy.

For detailed information see [90] and references therein.

6.5.2 Implementation

Parallel Tempering is only implemented in the parallel version of Tremolo-X. The sequential version simply ignores parameters that are set for Parallel Tempering. Furthermore, the number of started processes must be a multiple of the number of temperatures such that each replica simulation is shared by the same number of processes.

The temperatures are enumerated from 0 to $N - 1$ in increasing order. Analogically, the replicas among which the temperatures are exchanged are enumerated such that the i-th replica is held at the i-th temperature before the first exchange.

As the acceptance probability of a temperature exchange between two replicas decreases exponentially with the difference in inverse temperature, exchanges are only tried between replicas that are held at neighboring temperatures. Every second time Tremolo-X tries to swap the following pairs of temperatures (0,1),(2,3),(4,5),... Every other time it tries to swap the pairs (1,2),(3,4),(5,6),...

Every simulated replica produces a .xxxx.ptlog file, where xxxx is the replica number padded with leading zeros. For each temperature exchange that the respective replica undergoes, the file contains the point in time (given in simulation time) and the temperature number before and after the exchange.

All other output files carry the number of the temperature at which the replica was held when the respective measurement was made in their names just after the simulation prefix. After that, the non-visual output files carry a 4-digit serial number in their name such that the files can be ordered chronologically without reading their content.

6.5.3 Usage

Parameter file

To use Parallel Tempering add or edit the following lines to the `dynamics` section of the .parameter file:

```
paralleltempering    {
  temperatures:    state=on, t_start=#INT, t_delta=#INT, \
  ↪    [temperature=(#FLOAT),...,(#FLOAT)];
  };
```

`t_delta` is the number of integration steps between exchanges.
`t_start` is an additional number if integration steps that shall be passed before the first exchanges are tried.
`temperature` is followed by a list of the temperatures at which the replicas shall be held. The list must be given in increasing order.

Invocation

Start `tremolo_mpi` with the number of process that is the number of temperatures times the number of processes for each single simulation.

To set the number of processes per single replica enter the parameters for a single replica in the section `parallelization` of the .parameter file or use the command line switch `-p` analogically.

Postprocessing

To concatenate the data of the non-visual output for each temperature see the script *MergePTDataAccToTemp.py*.

Chapter 7

Optimization

While the optimization feature of Tremolo-X can be viewed as a non-dynamic simulation in its own right, it is very beneficial to temper a simulations starting ensemble prior to a dynamic simulation.

The optimization manifests itself in a minimization of the potential energy, by rearranging particle positions and/or the simulation domain.

There are several methods available to minimize a given function value. Gradient based methods are computationally convenient for Molecular Dynamics simulations, since the potential energy and its derivative need to be computed anyway. The method used by the Tremolo-X software is based on the modified **c**onjugated **g**radient (CG) method by Polak-Ribiére.[35]

When comparing a **s**teepest **d**escent to a CG method, we notice that the SC is more stable than the CG if the starting position is distant from a local minimum point. On the other hand the SD converges more slowly in the near neighborhood of such a point than the CG does. While it is possible to use both methods and switch after some iterations it is also possible to use a CG method with periodic resets to combine the benefits of both approaches.

Furthermore convergence can be stabilized by choosing *(strong) Wolfe* conditions for the line search:

With d_k the search direction from point q_k (here denoting the vector of all particle positions) let:

$$\Phi(\eta) := E_{pot}\left(\mathbf{q}_k + \eta \cdot \mathbf{d}_k\right) \Phi'(\eta) := \nabla E_{pot}\left(\mathbf{q}_k + \eta \cdot \mathbf{d}_k\right) \cdot \mathbf{d}_k$$

The so called *(strong) Wolfe* condition is then the combination of the *sufficient decrease* or *Armijo* condition

$$\Phi(\eta) \leq \Phi(0) + \alpha\eta\Phi'(0), \qquad 0 < \alpha < 1$$

and the *curvature* condition

$$\Phi'(\eta) \geq \beta\Phi'(0), \qquad 0 < \alpha < \beta < 1$$

.

The *Armijo* condition fixes an upper limit on acceptable new function values, whereas the *curvature* condition imposes a lower bound on the step size η.

TODO: Choosing a suitable η to satisfy the above conditions can be restricted to a certain interval (with respect to the original step length). Not implemented.

Another method to control the steps of the iteration method is to apply a prefactor to the step, in essence reducing the maximum change of the ensemble*TODO: without reset prefactor kills convergence?*.

The minimization iteration finishes, when the change in the mean force acting on a particle between consecutive iteration steps drops below a fixed threshold (drops below a fixed factor).

The minimization may be extended to the structure of the simulation domain as well, by observing changes in the potential energy when scaling with the box matrix. Depending on the type of box one wishes to retain from the input values, constraints have to be specified (to (de-)couple box edge length and or box angles). The matrix takes inputs 0 and 1, specifying accessible degrees of freedom, in addition constraints have to be specified as off (*TODO: is there even an off?*), none (no constraint, this is unphysical and should not be used), isotropic (box stays in cubic shape, only diagonal elements are changed[Andersen constraint]),standard (*TODO: effect of standard*) and symmetric (*TODO: effect of symmetric*).

Specifying an external pressure and/or stress values for the box is essential for the optimization results to conform with physical reality.

Chapter 8

Longrange Algorithms

The longrange interaction in Tremolo-X applies generally to the Coulomb (electrical) potential, describing the interaction between (partially) charged particles. Thus it is necessary to specify the dielectric constant by setting the permittivity. Since MD simulations are almost exclusively conducted in a medium free setting (meaning apart from the particles there is no background substance but free space in between particles) it is designated as ϵ_0, however that does not restrict you from setting this value to a material dependent value.

Another possibility is to set a negative permittivity coefficient, the resulting potential will be of a gravity type ("charges" with the same sign attract each other). The interpretation of the permittivity is then the gravitational constant.

For the computation of the long range potentials different methods are available. When using mesh-base methods, attention needs to be paid to the singularities encountered at the point charges (the particle positions).

8.1 N^2 Algorithms

These methods compute the actual particle-particle interaction as done for the short range potentials. While the method asks for a cutoff radius you should be aware that in contrast to the fast decaying short range potentials the cutoff-term of the potential energy does not converge to zero.

8.1.1 Hard Cutoff

The most simple algorithm simply computes interactions for all particles within the specified radius and ignores all particles beyond. There is no smoothing of the cutoff, leading to some discontinuities in the energy.

8.1.2 Spline

In order to avoid discontinuities in the potential, this methods smoothes the cutoff by fitting a spline between the potential value at the inner cutoff and zero at the outer cutoff.

8.2 Smooth-Particle-Mesh-Ewald Method

The **S**mooth-**P**article-**M**esh-**E**wald Method (SPME) combines the functionality of two approaches, a particle interaction and a mesh based approach. In order to handle the aforementioned singularities at the particle positions when solving the Poisson equation for the Coulomb potential, this method splits the potential into a short range and a long range part by screening functions. The short range part is handled by computing particle-particle interactions, while the long range part is treated by transferring the charge distribution onto a grid and solving the Poisson equation there (with modifications due to boundary conditions).

As the solver for the Poisson equation a **F**ast **F**ourier **T**ransform (FFT) or a multipole method may be used.

The screening function are Gaussian bell curves:

$$\rho(x) := \left(\frac{G}{\sqrt{\pi}}\right) e^{-G^2\|x\|^2}$$

where *operator splitting coefficient* G may be chosen by the user. Other choices of the user include again the cutoff radius, the number of meshcells per linked cell (if not a power of 2 rounded up to the next such number) and the interpolation degree of B-Splines (as the basis functions on the mesh).

The SPME method is a prudent choice in simulations with uniform particle densities. In cases where the particles are distributed in an inhomogeneous fashion, mesh based methods become inefficient, due to the requirements on mesh density.

Note that in Tremolo-X the SPME method is not implemented sequentially and thus requires a version with parallel capabilities. It can however be started with only one process.

8.3 Fast-Multipole-Method

The **F**ast-**M**ultipole-**M**ethod (FMM) is a tree based method. As the SPME method in the previous section, this approach is based on a separation of short and long range contributions as well. Particles are grouped in hierarchical cells and their compound momenta are recursively computed, allowing to compute the interaction for each particle in detail from its immediate surrounding and with approximated momenta for particle groups in distant cells.

For some domain Ω we define the momenta:

$$M_{\mathbf{j}}(\Omega, \mathbf{y}_0) := \int_\Omega \rho(\mathbf{y})(\mathbf{y}-\mathbf{y}_0)^{\mathbf{j}} d\mathbf{y}$$

and we introduce the following notation for the multidimensional Taylor series of a function G:

$$G(\mathbf{x},\mathbf{y}) = \sum_{|\mathbf{j}|\le p} \frac{1}{\mathbf{j}!} G_{0,\mathbf{j}}(\mathbf{x},\mathbf{y}_0)(\mathbf{y}-\mathbf{y}_0)^{\mathbf{j}} + R_p(\mathbf{x},\mathbf{y})$$

where $\mathbf{j} = (j_1, j_2, j_3)$ is a multi-index with factorial $\mathbf{j}! = j_1! \cdot j_2! \cdot j_3!$ and vector exponential $\mathbf{y}^{\mathbf{j}} = \mathbf{y}_1^{j_1} \cdot \mathbf{y}_2^{j_2} \cdot \mathbf{y}_3^{j_3}$. With the short hand $\frac{d^{\mathbf{j}}}{d\mathbf{y}^{\mathbf{j}}} = \frac{d^{j_1}}{d\mathbf{y}_1^{j_1}} \frac{d^{j_2}}{d\mathbf{y}_2^{j_2}} \frac{d^{j_3}}{d\mathbf{y}_3^{j_3}}$ we write

$$G_{\mathbf{k},\mathbf{j}}(\mathbf{x},\mathbf{y}) := \left[\frac{d_{\mathbf{k}}}{d\mathbf{w}^{\mathbf{k}}} \frac{d_{\mathbf{j}}}{d\mathbf{z}^{\mathbf{j}}} G(\mathbf{w},\mathbf{z}) \right]_{\mathbf{w}=\mathbf{x},\mathbf{z}=\mathbf{y}}.$$

Then

$$\Phi(\mathbf{x}) \approx \sum_\nu \sum_{|\mathbf{j}|\le p} \frac{1}{\mathbf{j}!} M_{\mathbf{j}}(\Omega^\nu, \mathbf{y}_0^n u) G_{0,\mathbf{j}}(\mathbf{x}, \mathbf{y}_0^\nu)$$

with $\mathbf{y}_0^\nu \in \Omega^\nu$, $\bigcup_\nu \Omega^\nu = \Omega$ and pairwise $\Omega_{\nu_1} \cap \Omega_{\nu_2} = \emptyset$.

In order to control the accuracy of the method, it is sensible to use small cells in a medium distance from the particle evaluated and large cells in further distance. This is controlled by specifying a multipole acceptance criterion θ: $\frac{diam(\Omega^\nu)}{\|\mathbf{x}-\mathbf{y}_0^\nu\|} \le \theta$ which must be satisfied by cell to be included in the sum. If the cell does not satisfy this requirement, the method proceeds to the cells children. Note that you also have to specify a maximum tree level and if the acceptance requirement and the maximum tree level have been chosen poorly, the algorithm drops to N^2 performance.

Further parameters that can be controlled are the degree of the multipole and Taylor expansion p and the cutoff radius for the short range interaction.

8.4 Ewald

TODO: Not implemented.

8.5 P3M

TODO: Not implemented.

8.6 PME

TODO: Not implemented.

8.7 Barnes-Hut

TODO: Not implemented.

Chapter 9

Parallelization

One of the key strength of Tremolo-X is the parallelization. Naturally there are some associated parameters, which you can set.

The most basic decision is the number of processes to be started. They are organized by a division of the simulation domain along the three major axes. The total number of processes is then the product of the number of processes in x direction times the number of processes in y direction times the number of processes in z direction.

There are some restrictions on the choice of processes: Each process must at least contain one cell, thus the choice of the number of processes must at least satisfy the inequality:
$\frac{edge\_length}{number\_processes} > lc\_rcut$,
where lc_rcut is the cut radius (edge length) of the linked cells.

A prudent choice of this parameter deserves some thought, as both a good and bad choice may reflect in the performance of the code. The range of the used potentials and the domain size and shape should be considered.

In the parameter file, the processor numbers are set as follows:

```
n1=<n1>      number of processes in x direction
n2=<n2>      number of processes in y direction
n3=<n3>      number of processes in z direction
all=<all>    product of n1 n2 n3
```

It may be overridden with the command line switch -p <n1>,<n2>,<n3>, as documented in chapter 4.

After the division is established, cells are distributed via (dynamic) load-balancing. You can choose a number of parameters to fit the use to your liking, apart from switching it on and off, you may specify intervals, after which the balancing is reexamined. In addition you may also choose, whether the load is measured linear or quadratic (with respect to the particle number assigned to a process).

Chapter 10

Measurements and Output

10.1 Time/Step Settings

Most measurements share the parameters `T_Start, T_Delta` and `Step_Delta`. The first two specify when in a dynamic simulation the code first starts measuring a given property `T_Start` and at what time intervals the measurement is to be repeated `T_Delta`. The time interval does not need to be an exact multiple of the timestep used in the propagator. While the measurement is carried out in the next timestep, after a time of `T_Delta` has passed and writes out the propagator time as the time of measurement, the program notes the exact time at which measurement should take place and continues to add the interval to that exact point in time. As a consequence the number of timesteps between measurements may vary by one. Note that for obvious reasons it is also not sensible to choose `T_Delta` smaller than the propagator timestep, consequently Tremolo-X will produce an error if you do.

Contrary to intuition, tremolo will accept negative starting times and treat them as zero.

The parameter `Step_Delta` applies, when an optimization is performed and indicates the measurement interval in terms of steps. Since only integers are valid input much of the above caution does not apply.

However it is noteworthy, that while the output files still indicate that the first column contains time, this is not a time in the physical sense but indicates the number of iteration steps performed.

10.1.1 Outvis

This regulates the output of visual(izable) data. Specifically in each output steps two files are created, a $PROJECTNAME.vis.####.xyz, $PROJECTNAME.vis.####.pdb and $PROJECTNAME.vis.####.data file. The first records particle elements and positions and features the timestep in which the file was written in the comment line, while the latter is PDB equivalent output and does not carry the exact time. The naming of the files

indicates one 4 digit number in the serial case: The consecutive numbering of the .vis output files. In the parallel case the naming convention changes to $PROJECTNAME.vis.####.####.xyz, where the first four digit number corresponds to the numbering in time, whereas the second number refers to the numbering of the processors. The files can be merged into one by using one of the utilities delivered with Tremolo-X *TODO: utilities*.

Uses the three general parameters described above.

10.1.2 Outdata

Here intervals are set for writing intermediate data files ($PROJECTNAME.data.9999). Intermediate data files enable a (re)start of a simulation with the current configuration. This may be useful in the case that a simulation was ended due to a time limit, or when some parameter setting is used for equilibration, while measurements should take place in another. When the program terminates normally a $PROJECTNAME.data.9999 file is written with the latest configuration, unless `endtime < T_Start - T_Delta`. As with the visual data, when the program is run in parallel mode, four extra digits ($PROJECTNAME.data.9999.####) are used to have each process write its own file. The files can be merged into one by using one of the utilities delivered with Tremolo-X *TODO: utilities*.

Uses the three general parameters described above.

10.1.3 Outm

This sets the measurement intervals for all actual measurements described below. Uses the three general parameters described above.

10.1.4 Outmm

In addition to measurements at a point in time Tremolo-X provides the ability to perform mean measurements, to measure a parameter over the course of a time interval and average the results. This feature smoothes out some of the unavoidable irregularities in the measurements due to discretization effects. In addition to the three general time parameters, `T_Deltam` and `Step_Deltam` need to be specified. Those establish the interval, within which the measurements and averaging are applied. Measurements will be performed by the following rules:

- The timer is started with `T_Start`.
- Let $n \in \mathbb{N} \setminus 0$
- Beginning with n·`T_Deltam-Step_Deltam` desired magnitudes are measured and summed up for each propagator timestep, until
- at n·`T_Deltam` the average is computed and printed to the output file.

10.2 Visual Output

10.3 Energy Measurements

Here the decision is made, whether energies and/or mean energies will be measured. All output files list the time in their first column and respective measurements in following columns. Energy measurements encompass the following:

- kinetic energies – stored in a single file $PROJECTNAME.ekin
  - One column for the physical kinetic energy of each particle type.
  - One column for the physical kinetic energy of all ensemble.
  - One column for the temperature of the whole ensemble.
- potential energies – stored in multiple files.
  - $PROJECTNAME.epot contains the complete potential energy
  - $PROJECTNAME.e_lj contains the potential energy of non bonded interactions
  - $PROJECTNAME.ebond – contains the potential energy stored in direct bonds. Due to the data structure the file contains some zero-filled columns.
  - $PROJECTNAME.eangle – contains the potential energy stored in angles between bonds. Due to the data structure the file contains some zero-filled columns.
  - $PROJECTNAME.etorsion – contains the potential energy of torsion forces. Due to the data structure the file contains some zero-filled columns.
- total energy – stored in a single file $PROJECTNAME.etot
  - One column contains the physical energy
  - In case a thermostat is used an additional column appears, which is either identical to the first, or – if the thermostat in use is a Hoover-type thermostat – is the sum of the physical energy and the Hoover energy. The case is similar for the use of a barostat, where columns are added for hoover and box energy by themselves and the sum over all three columns.

10.4 Radial Distribution

```
radial:   measure=on, meanmeasure=off, vis=off, \
↪   r_cut=1.470588, n_bin=50;
        radialdistribution
          {
            radialdist: particle_type1=Fe, \
            ↪   particle_type2=Ni;
          };
```

The options measure and meanmeasure work as described in 10.1. Another `r_cut` needs to be supplied, for the radial analysis this will determine the radius in which particles are taken into account for the distribution. With `n_bin` you can specify in how many bins the particles shall be sorted in the interval from 0 to `r_cut`. Lastly you have to list the pairs for which you would like to receive the relevant data. For each pair which you intend to examin you need to add a line

```
radialdist: particle_type1=Fe, particle_type2=Ni;
```

within the `radialdistribution` environment, where `Fe` and `Ni` in this example take the place of any particle types you would like to analyze.

The data is added to the `$PROJECTNAME.histogram` files. By reading the first line of the file, you find in which columns the data of a particluar measurement is stored.

The first column has index 0 and indicates a point in time to which the rest of the information in the line corresponds. For the following columns you have to check for an expression such as `RadialDist[Fe-Ni](1)(50)`, where `Fe` and `Ni` are the particle types which we added in our example, you need to check for your particle types. The following two numbers indicate the columns occupied by the measurement: The first column contains the index for the column in which the measurement begins, while the second indicated the number of columns occupied, which corresponds to the number of bins chosen in the `.parameter` file. Always remember that counting starts at 0 so that in the above example the data for the radial distribution of `Fe` and `Ni` starts in the 2^{nd} columns and ends in the 51^{st} column.

The value in each entry corresponds to the number of particles found in the respective bin, however instead of calculating the radial distribution from that on your own, you can use the functionality of the CalcRadialHisto script, which is documented in the Tremolo_Tool_Documentation, which will also compute some other useful information, such as the coordination numbers.

10.5 Bond Distances

```
bond: measure=on, meanmeasure=off, vis=off;
bondDistances {
    bondDistance: particle\_type1=Fe, particle\_type2=Ni, \
    ↪   distance=2.700000e+00;
  };
```

The options measure and meanmeasure work as described in 10.1. When the option `vis` is switched on, additional files are produced which contain the respective bonds by listing adjacent particle id's ($PROJECTNAME.dbond.####). Theses are produced at the intervals of the visual output. Bonds are identified by checking pairs of particles. Those whose distance is smaller than `distance` are considered bonded for the sake of measuring bond distances, however this does not have an impact on any bond dependent potentials.

The average bond length and numbers are written in the output file `$PROJECTNAME.generalmeas`. The first column (index 0) in this file contains the time, whereas the position of the bondtypes can be found in the first line of the file. By checking for the keyword `BondDist` followed by particle types (e.g. `BondDist[Fe-Ni](1)(2)`), you can identify the columns giving the result for the bonds between these particular particle types. The later of the two numbers will always be two, as two columns are associated with every pair. The first number following gives the index of the column with the average bond distance measured, the number of such bonds will always be in the following column.

Note: The file named `$PROJECTNAME.info.bonds` does not list bonds measured by this analysis, but bonds defined in the `$PROJECTNAME.data` file.

10.6 Diffusion measurements

10.6.1 Definition

Diffusion coefficients are an equilibrium property. Thus for the computations to be sensible, the ensemble should be in equilibrium or steady state *TODO: ref and/or write up steady state*.

The derivations of the equations can be found in the standard literature, e.g. [33, 38] or in the authors words in [77].

Let $r_i(t) = x_i(t) - x_i(0)$. The **m**ean **s**quare **d**istance (MSD), reads as

$$\left\langle r^2(t) \right\rangle = \frac{1}{n} \sum_{i=0}^{n} (r_i(t))^2, \tag{10.1}$$

resulting in the traditional formula for the diffusion coefficient

$$D = \frac{1}{6} \frac{\left\langle r^2(t) \right\rangle}{t}, \tag{10.2}$$

which is commonly referred to as the Einstein relation.

There is a mathematically equivalent approach, which is not based on the averaging over distances but integrating over the velocities. This numerically different approach to the calculation of the diffusion coefficient is the

Velocity **A**uto**c**orrelation (VAC). Using the classic equation

$$r(t) = \int_0^t v(\tau) d\tau,$$

we obtain the VAC or Green-Kubo relation for diffusion:

$$\begin{aligned} D &= \frac{1}{6}\frac{\partial}{\partial t}\left\langle r^2(t)\right\rangle && (10.3)\\ &= \frac{1}{6}\frac{\partial}{\partial t}\int_0^t\int_0^t \left\langle v(\tau')v(\tau'')\right\rangle \, d\tau' \, d\tau'' \\ &= \frac{2}{6}\frac{\partial}{\partial t}\int_0^t\int_0^{\tau'} \left\langle v(\tau')v(\tau'')\right\rangle \, d\tau' \, d\tau'' \\ &= \frac{1}{3}\frac{\partial}{\partial t}\int_0^t\int_0^{\tau'} \left\langle v(\tau''-\tau')v(0)\right\rangle \, d\tau' \, d\tau'' && (10.4)\\ &= \frac{1}{3}\int_0^t \left\langle v(t-\tau)v(0)\right\rangle \, d\tau \end{aligned}$$

where in (10.4) the fact that this is an equilibrium property is explicitly used to justify the shift in time [33].

Corrected Diffusion Measurement

TODO: ARTICLE OUT BEFORE MANUAL?!? Particle motion is not always only random, may have directed components as well (*TODO: ref outer forces*). The separation of the particle movement in diffusive and convective parts is not taken into account in the traditional computations of the diffusion coefficients. Thus, when in addition to the diffusive motion an additional directed component is present during the measurement, it is necessary to correct for this convection induced movement, which might taint the results [77].

Obtaining the undirected mobility of the particles is achieved by subtracting the mean displacement before squaring in (10.1), resulting in

$$\left\langle L^2(t)\right\rangle = \frac{1}{n}\sum_{i=0}^{n}\left(r_i(t) - \left\langle r(t)\right\rangle\right)^2.$$

In this equation we measure in fact the variance of the variable "distance traveled", which allows the use of standard error estimation techniques for variance expressions. From this variance we can compute the diffusion over the time interval t

$$D = \frac{1}{6}\frac{\left\langle L^2(t)\right\rangle}{t}.$$

FIXME:This needs some more discussion, since steady state is violated. We now turn towards a corresponding convection correction for the VAC.

Beginning with the respective convection corrected variant, we find that

$$\begin{aligned}\left\langle L^2(t)\right\rangle &= \frac{1}{n}\sum_{i=0}^{n}(r_i(t) - \langle r(t)\rangle)^2 \\ &= 2\int_0^t\int_0^\tau \left(\left\langle v(\tau)v(\tau')\right\rangle - \langle v(\tau)\rangle\left\langle v(\tau')\right\rangle\right) d\tau' d\tau.\end{aligned}$$

We now perform the same shift in the variable as in the original derivation (10.4) and insert the above result in 10.3 to obtain

$$D = \frac{1}{3}\lim_{t\to\infty}\int_0^t \langle v(t-\tau)v(0)\rangle - \langle v(t-\tau)\rangle\langle v(0)\rangle\, d\tau, \qquad (10.5)$$

which is the complete form of the convection corrected velocity autocorrelation. *TODO: write about steady state*

10.6.2 MSD-Options

Things to set:

- `starttime` – Self explanatory
- `resetinterval` – The interval, after which starting positions and velocities are reinitialized to the current values. In essence a restart of the measurement procedure.
- `convection_correction` – Allows to choose, whether measurements shall be corrected for convective contributions.
- `measureflag` – 0 == no measurement, 1 == only point-wise measurement, 2 == only mean measurement, 3 == point-wise and mean measurement
- `msdusegroups` – *TODO: This option is currently buggy.*
- `particle_type` – *TODO: This option is currently buggy.*
- `groupno` – *TODO: This option is currently buggy.*

TODO: groups & types currently out of order

10.6.3 MSD-Output

The output of diffusion measurements can be found in the file $PROJECTNAME.msd ($PROJECTNAME.msd.mean). It is organized in 5 groups (6 if you count time), each of which has multiple columns (as e.g. the .ekin file). All groups have a column for the values of each particle type measured and one for the values of the complete system. The number is indicated in

the head line of the file by \$GROUP$(i)(n)$, where i indicates the starting column, and n the number of respective columns (equal to the number of particles measured). Additionally, there is a column \$GROUP_sum, which is the sum of the previous (n) columns. The groups are:

- Time – scaled time, needs to be converted to physical time
- MSD – mean square displacement in chosen unit of length $\langle r^2(t) \rangle$.
- Dif – Diffusion computed by MSD in chosen units of length and time $D = \frac{1}{6} \frac{\langle r^2(t) \rangle}{t}$.
- VAC – VAC for single time step - VAC averaged over particles but not time-scaled $\sum_i v_i(t-\tau)v_i(0)$.
- VACDif_unscaled – VAC summed for all particles - VAC summed over time and particles, but not averaged and time-scaled. $\int_0^t \sum_i v_i(t-\tau)v_i(0)$.
- VACDif – VAC integral - averaged, time-scaled and dimension distributed - the diffusion value in chosen units of length and time $\frac{1}{3} \int_0^t \langle v(t-\tau)v(0) \rangle \, d\tau$. (The relation VACDif $= \frac{1}{3} \frac{\Delta t}{\#Particles} \cdot$ VACDif_unscaled holds.)

Note, that as always the units are tremolo reference units. Units of Dif and VACDif are equal.

Note: Formulas given refer to the default, for the corrected method, the corrected formulas above apply!

10.7 Velocity Distribution

```
velocity:                    measure=on,                    \
↪   meanmeasure=off,                    vis=off,          \
↪   min=0,   max=0.006382352,           n_bin=50;
```

The options measure and meanmeasure work as described in 10.1. With `n_bin` you can specify in how many bins the particles shall be sorted in the interval from 0 to `r_cut`. Lastly you have to list the pairs for which you would like to receive the relevant data.

The data is added to the `$PROJECTNAME.histogram` files. By reading the first line of the file, you find in which columns the data of a particluar measurement is stored.

The first column has index 0 and indicates a point in time to which the rest of the information in the line corresponds. For the following columns you have to check for an expression such as `VelocityDist[GMT-Argon](51)(50)`, where `Argon` is a particle type present in the simulation. (The shorthand `GMT` stands for `General Measure Type`. In most cases this defaults to the

particle types, however if you declared separate group types (`GrpTypeNo`) in the `.data` file, you will find the respective group type numbers in four digit format.)

The two numbers following the `GMT` identifier indicate the columns occupied by the measurement: The first column contains the index for the column in which the measurement begins, while the second indicated the number of columns occupied, which corresponds to the number of bins chosen in the `.parameter` file. Always remember that counting starts at 0 so that in the above example the data for the velocity distribution of `Argon` starts in the 52^{nd} column and ends in the 101^{st} column.

The value in each entry corresponds to the number of particles found to have the velocity associtated with the bin.

10.8 Stress

TODO: Not all of the information in stress is suitable for a user, so some rewriting is sensible

10.8.1 Definitions

Let $\vec{a}_1, \vec{a}_2, \vec{a}_3$ be the basis vectors of the simulation box. Let $h = [\vec{a}_1, \vec{a}_2, \vec{a}_3]$ be a 3x3 matrix. Let $\vec{r}_i$ be the positions and $\vec{v}_i$ the velocities of the n particles. The volume comes out to $\Omega = \det h$. Defining the scaled positions

$$\vec{s}_i = h^{-1}\vec{r}_i$$

we get $\vec{v}_i = \dot{\vec{r}}_i = h\dot{\vec{s}}_i$, with $\dot{h} = 0$. Force: $F_i = m_i\ddot{\vec{r}}_i$ with mass m_i.

Internal Stress Tensor I

Let $E(\vec{r}_1, \ldots, \vec{r}_n; \vec{v}_1, \ldots, \vec{v}_n) = E(\{\vec{r}_i\}; \{\vec{v}_i\})$ be the total energy. It can also be written as a function of h: $E^h(\{\vec{s}_i\}; \{\dot{\vec{s}}_i\}; h) = E(\{h\vec{s}_i\}; \{h\dot{\vec{s}}_i\})$ The stress-tensor (internal stress tensor) Π is defined as follows:

$$\Pi_{\alpha\beta} = -\frac{1}{\Omega}\sum_{\gamma}\frac{\partial E^h}{\partial h_{\alpha\gamma}}h_{\beta\gamma}$$

$$\Pi = -\frac{1}{\Omega}\sum_{i}\left(m_i\vec{v}_i\vec{v}_i^T + \vec{r}_i\vec{F}_i^T\right)$$

By definition of the pressure tensor P we have

$$\begin{pmatrix} P_{xx} & P_{xy} & P_{xz} \\ P_{yx} & P_{yy} & P_{yz} \\ P_{zx} & P_{zy} & P_{zz} \end{pmatrix} = -\Pi$$

and for the instantaneous hydrostatic pressure $p = \frac{1}{3}\text{Spur } P$. These magnitudes are computed in the program. (Identified as stress throughout

the code and is by this definition the pressure tensor.) (Also note: e.g. $P_{xy} = \frac{1}{\Omega}\sum_i \left(m_i \vec{v}_{ix}\vec{v}_{iy} + \vec{r}_{ix}\vec{F}_{iy}\right)$: This definition is the usual one. It is only valid in case of a finite volume. *TODO: periodic, reflecting?*. **Thus the formula may not be used as is in the periodic case!(For discussion see section 10.8.2)**

Internal Stress Tensor II

Let

$$\epsilon = \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \epsilon_{zz} \end{pmatrix}$$

In analogy to the above let: $E^\epsilon(\{\vec{r}_i\};\{\vec{v}_i\};\epsilon) = E(\{(1+\epsilon)\vec{r}_i\};\{(1+\epsilon)\vec{v}_i\})$. Then

$$-\frac{\partial E^\epsilon}{\partial \epsilon_{\alpha\beta}}\bigg|_{\epsilon_{\alpha\beta}=0} = \Omega\Pi_{\alpha\beta}.$$

holds. The gradient on the left side can be used for a steepest descend or conjugated gradient procedure. To compute the stress tensor one only has to choose a normalization coefficient: Ω.

Note: In the case of tubes the literature is not specific about what volume to choose. Some use the full cylinder, some a pipe with some thickness, other assume a volume per particle and even other people define a surface-stress-tensor by dividing through the surface instead of the volume of the tube.

Stress-Strain

TODO: more of an experiment suggestion? The following description refers to an experiment with a nanotube: The tube is oriented along the z-axis. The tube is observed in a reference state (normally equilibrium) with a deflection l_0 in z direction. Now the experiment is performed (pushing/pulling in z direction) and the deflection in z direction and the zz-component of the stress-tensors Π_{zz} are computed. For a stress-strain-diagram use the measured stress and the computed strain $\epsilon_z = \frac{l-l_0}{l_0}$.

Young Modulus

TODO: more of an experiment suggestion? We assume (and experiments confirm), that a tube behaves as a spring. Thus one chooses l_0 as equilibrium. Now we obtain a linear stress-strain diagram for some domain (as for a spring). In this case the Young Modulus is the slope of the line (corresponding to Hooks constant). Generally the definition is:

$$Y_{\alpha\beta} = \frac{1}{\Omega}\frac{\partial^2 E}{\partial^2 \epsilon_{\alpha\beta}}\bigg|_{\epsilon_{\alpha\beta}=0}$$

Sometimes (possibly to circumvent the volume problem) the following definition is used:

$$Y_{\alpha\beta} = \frac{1}{n} \left. \frac{\partial^2 E}{\partial^2 \epsilon_{\alpha\beta}} \right|_{\epsilon_{\alpha\beta}=0}.$$

Observing the stress-strain diagrams, those are found to be parabolas, with a minimum for strain = 0 (with l_0 the equilibrium). And the Young Modulus is (corresponding to the spring assumption) the curvature in 0 (lacking normalization). *TODO: Sign might be wrong.*

10.8.2 Implementation

According to section (10.8.1) the formula for the stress-tensor reads:

$$\Pi = -\frac{1}{\Omega} \sum_i \left(m_i \vec{v}_i \vec{v}_i^T + \vec{r}_i \vec{F}_i^T \right) \tag{10.6}$$

In the periodic case the term

$$\vec{r}_i \vec{F}_i^T \tag{10.7}$$

needs separate treatment, since also the forces between the particles and the periodic images need to be taken into account.

Question: Where is the problem?
Answer: In the derivation of 10.6 and 10.7 it was assumed that:

$$E_{pot}(\{\vec{r}_i\}) = E_{pot}(\{h\vec{s}_i\}) \tag{10.8}$$

However this assumption does not necessarily hold in the periodic case, since E_{pot} is not only dependent on $\{h\vec{s}_i\}$. We have for the periodic potential:

$$E_{pot}(h, \{\vec{r}_i\}) = E_{pot}(h, \{h\vec{s}_i\}) \tag{10.9}$$

Thus we don't obtain a formula reducing the stress to dependencies of $\vec{F}_i$ and $\vec{r}_i$, however for the commonly used potentials the following approach is valid:

$$E_{pot}(h, \{\vec{r}_i\}) = E_{pot}(h, \{\vec{r}_{i'}\}) = E_{pot}(\{\vec{r}_{i'}\}) = E_{pot}(\{h\vec{s}_{i'}\}). \tag{10.10}$$

where

$$\{\vec{r}_{i'}\} = \{\vec{r}_{i'} \,|\, \vec{r}_{i'} = h\mathbf{a}\,,\, \mathbf{a} \in \mathbb{Z}^3\} \tag{10.11}$$

This set can generally be reduced to a finite one (*cutoff*).

Using pair potentials (The third law of Newton holds) we can do the following:

$$\vec{r}_{ij} := \vec{r}_j - \vec{r}_i$$
$$\vec{F}_i = \sum_{j \neq i} \vec{F}_{ij} \quad \text{mit } \vec{F}_{ij} \text{ von i nach j}$$

s.t. without use of periodic images

$$\sum_i \vec{r}_i \vec{F}_i^T = \sum_{i<j} -\vec{r}_{ij} \vec{F}_{ij}^T \tag{10.12}$$

holds.

The right hand side of (10.12) contains only differences of $\vec{r}_{ij}$. Thus the stress can be computed by use of the forces over the course of a standard linked cell method. Together with periodic images we have

$$\sum_i \vec{r}_i \vec{F}_i^T + \sum_{i''} \vec{r}_{i''} \vec{F}_{i''}^T = \sum_{i<j} -\vec{r}_{ij} \vec{F}_{ij}^T + \sum_{i<j''} -\vec{r}_{ij''} \vec{F}_{ij''}^T \tag{10.13}$$

where: $\{\vec{r}_{i''}\} = \{\vec{r}_{i'}\} \backslash \{\vec{r}_i\}$. Specifically, there is no contribution of an interaction within $\{\vec{r}_{i''}\}$. Of course there are other methods to deal with this problem, such as:

As done in the parallel case anyways, (periodic) boundary cells could be added, in which the images with respective coordinates are held. All that is left to do then is *correct* summation. (Left hand side of (10.12) also for periodic images.)

With the different potential types (2-body, 3-body, Coulomb, ...) the solutions to the *periodic image* problem vary

Note: With the exception of coulomb all force contributions are reduced to pair terms (analogous to the right hand side of (10.12)).

Example 3-body (j-i-k) with $F_i = -F_j - F_k$

$$\vec{r}_i \vec{F}_i^T + \vec{r}_j \vec{F}_j^T + \vec{r}_k \vec{F}_k^T = \vec{r}_{ij} \vec{F}_j^T + \vec{r}_{ik} \vec{F}_k^T$$

or 4-body (i-j-k-l) with $F_i = -F_j - F_k - F_l$ and

$$\begin{aligned} F_i &= -F_0 \\ F_j &= F_0 - F_1 \\ F_k &= F_1 - F_2 \\ F_l &= F_2 \end{aligned}$$

$$\vec{r}_i \vec{F}_i^T + \vec{r}_j \vec{F}_j^T + \vec{r}_k \vec{F}_k^T + \vec{r}_l \vec{F}_l^T = \vec{r}_{ij} \vec{F}_0^T + \vec{r}_{jk} \vec{F}_1^T + \vec{r}_{kl} \vec{F}_2^T$$

The coulomb case requires extra formulas.

10.8.3 Output syntax

The stress measurements are written to the `$PROJECTNAME.stress` file. In this file the information is organized as follows:

- The first column contains the time.

- The following 6 columns (indices 2-7) contain the stress/strain matrix entries (taking into account symmetrie) in the order `xx, xy, xz, yy, yz, zz`. *TODO: Only the non velocity part? I compared to the .mbox fiel.*
- The next 6 columns (indices 8-13) contain the velocity part of the stress tensor.
- In the last 12 columns the respective minimal and maximal contributions to the stress are recorded. The min/max pairs are listed together for each matrix entry. *Note that here the diagonal entries are listed first, followed by the off-diagonal entries* `xy, xz, yz`.

10.8.4 Individual Stress

Tremolo-X provides the ability to compute the individual stress of each particle, or to be more precise: The equivalent stress contributions originating at an individual particle. In order to switch those computations on, you need to add the following line(s) in the parameters file:

```
output {
  analyze {
    local_stress: localstress=on;
  };
};
```

(The `output` and `analyze` environments might already be present for other measurements; the relevant keyword is the `local_stress`.)

10.9 Box Dimensions and Forces

.mbox and .mforce

mbox *TODO: Double check all (specifically positions)!* This file contains:

- time/optimization step: 1 column
- box volume: 1 column
  The volume is computed as the determinant of the boxmatrix: $det(H_x)$
- box pressure: 1 column
- Box H_x: NDIM*NDIM columns
  The box vectors describing the parallelepiped (the box matrix).
- (Box H_u: NDIM*NDIM columns)
  The vectors denoting the change of the box (virtual velocity) (if in dynamics mode).

- Box H_F: NDIM*NDIM columns
  The vectors denoting the virtual force on the box. (see e.g. Section 2.1.1 in [35])

- Box Stress Tensor: 2*NDIM columns
  Since the stress tensor is symmetric, there are only entries `xx, xy, xz, yy, yz, zz`, while `xy=yx` etc.

- (Box Vel StressTensor: 2* NDIM columns)
  The velocity contribution to the above stress tensor.

- HVecOPNorm: 3 columns (x, y, z)
  These three lines contain the projection of the box vectors on the orthogonal system of the box. In most cases these values can be ignored, since they serve mainly internal purposes. However if you do not know the orthogonal projection (size) of your box those values can be of help. Note that the orthogonal system of the box is not necessarily the Cartesian system, if your box vectors include a rotation. (In the Cartesion case these values are identical to the vector norm of the three box vectors as a special case.)

mforce In the optimization case we find the output of the norm of the full force vector in this file (normalized by degrees of freedom).

$$\sqrt{\frac{P->Dynamics.ParOpt.NormSqF}{(NDIM \cdot \#Particles)}}$$

mforce.mean *TODO: empty?*

10.10 Runtime measurement

Tremolo-X also provides you with some information about its runtime performance. The information is amended to `$PROJECTNAME.speed`, meaning that the information of previous simulations with the same name in the same folder is retained. Each simulation adds two lines to the file, one line with column headers, which is always identical and one line with the respective data.

The information displayed is the following:

- `init`: The time used for initialization of the simulation.

- `sim`: The time used for the actual simulation

- `output`: The time used to output the simulation results.

- `average`: The average time that was used to compute one time/optimization step.
- `min`: The minimum time that was used for one time/optimization step.
- `max`: The maximum time that was used for one time/optimization step.
- `stddev`: The standard deviation of the time use per time/optimization steps.
- `steps`: The number of time/optimization steps.
- `particles`: The number of particles in the simulation.

10.11 Information files

In these files some information from the .data file is recollected in a different format.

- $PROJECTNAME.info.bonds
  This file contains an adjacency list for all bonded particles. The first non-comment line contains the number of bonds, followed by list of bonds by particle id's. There are no repetitions (e.g. bond [3 25] is listed, [25 3] is not).
- $PROJECTNAME.info.particlegroups
  This file lists the different groups to which each particle belongs. The first column contains the particles by id, followed by `ParTypeNo` `GrpMesTypeNo` and `ExtTypeNo`. *TODO: ExtTypeNo*
- $PROJECTNAME.info.particletypes
  This file lists some properties of the different particle types used in the simulation. Each particle type is assigned its number, chemical element and name in addition to the *TODO: DefaultZ*, *TODO: DefaultRadius* and *TODO: DefaultRGB* *TODO: Are these EAM (Embedded Atom Method) parameters?*

10.12 Pressure

TODO: inner vs. outer pressure?

10.13 Center of Mass

This file is self explanatory, it contains for columns, of which the first is the time and the remaining three contain the x-, y-, and z-coordinates of the center of mass.

Chapter 11

Potentials

TODO: translate potential explanations to english.
TODO: Citations need to be collected!

11.1 From PDB to Simulation

TODO: Is this section (PDB) sensible for the distributed version of Tremolo-X?
TODO: For current parser?

11.1.1 Protein Data Bank

The coordinates of proteins can be obtained from the Protein Data Bank (PDB) [4]. The data therein is mostly obtained from NMR-measurements or X-ray crystallography, thus coordinates for hydrogen atoms are usually missing since those have little mass and are hard to measure in Experiment. The PDB-format is published in a standard [3].

11.1.2 Hydrogenbuilder

Due to chemical descriptions of the standard residues of proteins and some empirical bonding rules for atoms it is possible to generate coordinates for hydrogen atoms in retrospect. Commercial products such as HyperChem [1] or CHARMM are available online. Using so called topology descriptions the relevant data can also be generated by hand. *TODO: reference to other commercial/open?*

11.1.3 Waterbath

For protein simulations usually the behavior in a solution (of water) is of interest. Thus water molecules are added in a periodic cuboid around the molecule under investigation. This can be done by HyperChem as well (but a

small script serves the same purpose). *TODO: document utilities separately? TODO: reference to other commercial/open?*

11.1.4 CHARMM data: Potential- and Topologyfiles

Now the PDB data (augmented with hydrogen atoms and water molecules) needs to be converted and parsed. Within the molecular dynamics program CHARMM two essential files, describing this conversion, exist. (Other Force Fields, s.a. AMBER, MM+ (HyperChem), GROMOS etc. have similar descriptions.) Descriptions: The particles are distinguished by atom type. This is a finer differentiation then by chemical elements, so that empirical knowledge and modeling (such as bonding structure, etc.) may be used as well.

In the potential file a set of Lennard-Jones (ϵ, σ) parameters is kept for each particle type. For pairs of particle types values for harmonic bonds (k_b, r_0), for atom triple angle data (k_θ, θ_0), and for 4-tuple (k_ϕ, n, δ) and (k_ψ, ψ_0) for torsion potentials and improper torsion potentials are kept. (The CHARMM-potential field uses a sum of torsion contributions with a variety different of n).

Example [2] (following ! you find a comment):

```
BONDS
!V(bond) = Kb(b - b0)**2
!Kb: kcal/mole/A**2
!b0: A
!atom type Kb          b0
CT1  C     250.000     1.4900
CT1  CC    200.000     1.5220

ANGLES
!
!V(angle) = Ktheta(Theta - Theta0)**2
!Ktheta: kcal/mole/rad**2
!Theta0: degrees
!atom types      Ktheta    Theta0
H    NH1  C       34.000   123.0000
H    OH1  CA      65.000   108.0000
```

The topology file describes the structure of the individual residues. The name of the residue and the atom are taken from the PDB file, together with the topology file the partial charges are determined, which are required to determine the potential parameters for a particle and particle type. Neighbor and bonding information is present as well. The description in internal coordinates may be used for the addition of hydrogen atoms to the PDB data.

Example: Description of the residue Glycin [2]:

```
RESI GLY              0.00
GROUP
ATOM N      NH1      -0.47   !       |
ATOM HN     H         0.31   !       N-H
ATOM CA     CT2      -0.02   !       |
ATOM HA1    HB        0.09   !       |
ATOM HA2    HB        0.09   ! HA1-CA-HA2
GROUP                        !       |
ATOM C      C         0.51   !       |
ATOM O      O        -0.51   !       C=O
                             !       |
BOND N HN   N  CA   C CA
BOND C +N   CA HA1  CA HA2
DOUBLE O  C
IMPR N -C   CA HN   C CA    +N O
DONOR HN N
ACCEPTOR O C
IC -C    CA    *N    HN     1.3475  122.8200  180.0000  115.6200 \
↪     0.9992
IC -C    N     CA    C      1.3475  122.8200  180.0000  108.9400 \
↪     1.4971
IC N     CA    C     +N     1.4553  108.9400  180.0000  117.6000 \
↪     1.3479
IC +N    CA    *C    O      1.3479  117.6000  180.0000  120.8500 \
↪     1.2289
IC CA    C     +N    +CA    1.4971  117.6000  180.0000  124.0800 \
↪     1.4560
IC N     C     *CA   HA1    1.4553  108.9400  117.8600  108.0300 \
↪     1.0814
IC N     C     *CA   HA2    1.4553  108.9400 -118.1200  107.9500 \
↪     1.0817
PATCHING FIRS GLYP
```

An accurate description of the syntax can be found in the CHARMM-documentation [21].

Furthermore, CHARMM permits additional terms for hydrogen bods, however those are not strictly necessary — hydrogen bonds are modeled by coulomb interaction. (Assuming accurate partial charges.)

11.1.5 Conversion and Parsing

A (Perl-)script the reads the PDB-data, the potential parameters and the topology information, coordinates atoms with their particle types and partial charges, constructs a list of neighbors and produces a file with all necessary particle and potential information. Those files are then used by Tremolo-X.

11.2 Lennard-Jones Potentials

Tremolo-X supplies several implementations of Lennard-Jones potentials. In addition to the standard 12-6 potential, the user may set the exponents on his own. Since the use of a cut-off radius may lead to discontinuities in the particle forces and energies, we supply two different spline interpolations, smooth the potential to a continuously differentiable one.
Note that the Lennard-Jones parameters ϵ (ϵ_{14}) and σ (σ_{14}) are used as parameters for some other potentials as well. The parameters for the potential between particles of different types are created using the Lorentz-Berthelot mixing methods:

$$\epsilon = \sqrt{\epsilon_1 * \epsilon_2}$$
$$\sigma = \frac{\sigma_1 + \sigma_2}{2}$$

TODO: SlaterKirkwood written but not implemented yet.

Standard 12-6 Lennard-Jones

$$U = 4 \cdot \epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

```
nonbonded_2body_potentials        {
        lennardjones:     particle_type1=H,
                          particle_type2=H,
                          r_cut=1;
        };
```

Particle types may be set to any particle you specified in the particle section of this file.

The `r_cut` denotes the radius, at which the potential is cut of.

M-N-Lennard-Jones *TODO: Bug/Inconsistency in tremolo with units/power of A and B.*

$$U = 4 \cdot \epsilon \left[\left(\frac{A_{ij}}{r_{ij}} \right)^{m} - \left(\frac{B_{ij}}{r_{ij}} \right)^{n} \right]$$

```
nonbonded_2body_potentials        {
        m_n_lennardjones:         particle_type1=H,
                                  particle_type2=H,
                                  a=0.44117647,
                                  b=0.44117647,
                                  m=12,
                                  n=6,
                                  r_cut=1.6176471;
        };
```

Particle types may be set to any particle you specified in the particle section of this file.

The `r_cut` denotes the radius, at which the potential is cut of.

11.2.1 Spline Interpolation

The smoothing for the Lennard-Jones potential is implemented by the multiplication of a spline to the potential function: $U_{smooth} = U \cdot S_i$. The following two spline interpolations are available in Tremolo-X:

Spline I *TODO: find first spline in code and verify*

$$S_I(r) = \begin{cases} 1 & : r \leq r_l \\ 1 - (r - r_l)^2 \left(3r_{cut} - r_l - 2r\right) / \left(r_{cut} - r_l\right)^3 & : r_l < r < r_{cut} \\ 0 & : r \geq r_{cut} \end{cases}$$

```
nonbonded_2body_potentials      {
      ljspline:         particle_type1=H,
                        particle_type2=H,
                        r_cut=3,
                        r_l=2.5;
      };
```

Particle types may be set to any particle you specified in the particle section of this file.

The `r_cut` denotes the radius, at which the potential is cut of.

`r_l` is the radius, where spline interpolation from the actual potential to 0 (at `r_cut`) starts.

Spline II *TODO: interpret second spline parameters in code.*

$$S_{II}(r) = \begin{cases} 1 & : r \leq r_s \\ & : r_s < r < r_m \\ & : r_m < r < r_e \\ & : r_e < r < r_b \\ 0 & : r \geq r_b \end{cases}$$

```
nonbonded_2body_potentials      {
      ljspline2:        particle_type1=H,
                        particle_type2=H,
                        r_s=0.88,
                        r_m=1.029,
                        r_e=1.470,
                        r_b=1.617;
      };
```

Particle types may be set to any particle you specified in the particle section of this file.

11.3 Non-bonded Potentials

11.3.1 Brenner

TODO: Double check potential (taken from book): This potential is designed for the use with Carbon and Hydrogen atoms.

$$U = \sum_{i=1}^{N} \sum_{j=1,j>i}^{N} f_{ij}(r_{ij}) \frac{c_{ij}}{s_{ij} - 1} \left[U_R(r_{ij}) - \bar{B}_{ij} U_A(r_{ij}) \right]$$

with a repulsive (U_R) and an attractive (U_A) component:

$$U_R(r_{ij}) = \exp\left(-\sqrt{2s_{ij}}\beta_{ij}(r_{ij} - r_{ij,0})\right)$$
$$U_A(r_{ij}) = s_{ij} \cdot \exp\left(-\sqrt{\frac{2}{s_{ij}}}\beta_{ij}(r_{ij} - r_{ij,0})\right).$$

Furthermore, we have:

$$f_{ij}(r) = \begin{cases} 1 & : r < r_{ij,1} \\ \frac{1}{2}\left[1 + \cos\left(\pi \frac{r - r_{ij,1}}{r_{ij,2} - r_{ij,1}}\right)\right] & : r_{ij,1} \leq r < r_{ij,2} \\ 0 & : r_{ij,2} \leq r \end{cases}$$

and

$$\bar{B}_{ij} = \frac{(B_{ij} + B_{ji})}{2} + K(N_i, N_j, N_{ij}^{conj}),$$

where

$$B_{ij} = \left(1 + H_{ij}(Ni^H, N_i^C) + \sum_{k=1,k\neq i,j} G_i(\theta_{ijk}) f_{ik}(r_{ik}) \exp\left(\alpha_{ijk}(r_{ij} - R_{ij} - r_{ik} + R_{ik})\right)\right)^{-\delta_i}.$$

The index of $G_i()\theta_{ijk}$ is used to indicate the dependency on the type of the i^{th} atom:

$$G_H = 12.33$$
$$G_C = a_0 \left(1 + \frac{c_0^2}{d_0^2} - \frac{c_0^2}{d_0^2 + (1 + \cos(\theta_{ijk}))^2}\right).$$

Additionally we define:

$$N_i^C = \sum_{j \in C} f_{ij}(r_{ij})$$
$$N_i^H = \sum_{j \in H} f_{ij}(r_{ij})$$
$$N_{ij}^{conj} = 1 + \sum_{k \in C, k \neq i,j} f_{ik}(r_{ik}) F(N_k - f_{ik}(r_{ik})) + \sum_{k \in C, k \neq i,j} f_{jk}(r_{jk}) F(N_k - f_{jk}(r_{jk}))$$

and

$$F(z) = \begin{cases} 1 & : z \leq 2 \\ \frac{1}{2}\left[1 + \cos(\pi(z-2))\right] & : 2 < z \leq 3 \\ 0 & : z \geq 3 \end{cases} .$$

The functions H_{ij} and K are spline functions, smoothing the change from bonded to non-bonded state and among neighbors, respectively.

Values fitting the fixed parameters can be found in the following table:

$a_0 =$	0.00020813
$c_0 =$	330
$d_0 =$	3.5

TODO: Complete table Brenner parameters.

```
brenner {
        brenner:        particle_type1=H,
                        particle_type2=H,
                        brennerhydrogen=off,
                        parameterset=I,
                        brennerlj=off;
        };
```

TODO: document parametersets and brennerlj.

11.3.2 Coulomb-ERFC

TODO: Double check potential (taken from tooltip):

$$u(r_{ij}) = \frac{e^2}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}}\mathrm{erfc}\left(\frac{r_{ij}}{\beta_{ij}}\right)$$

```
nonbonded_2body_potentials      {
        coulomb_erfc:   particle_type1=H,
                        particle_type2=H,
                        r_cut=3.2352941,
                        beta=0.68823529;
        };
```

Particle types may be set to any particle you specified in the particle section of this file. Note that the coulomb-force-constant `epsilon0inv` has to be specified, see also section 13.5.4.

11.3.3 Coulomb-QTaper

TODO: Double check potential (taken from tooltip):

The QTaper is only applied if the Coulomb potebtial is used. It is used to avoid a collaps of a positive and a negative charged particle. In Tremolo-X it is implemented in the form:

$$u(r_{ij}) = left(\frac{e^2}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} - C) \times (f(r_{ij}) - 1)$$

where the taper is given by

$$f(r) = \frac{e^2}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}}.$$

and r_0 is the cutoff. The Coulomb potential together with the QTaper results in the short range $[0, r_0]$ in:

$$\frac{e^2}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} f(r_{ij}) + C \times (1 - f(r_{ij})).$$

See also [34].

The parameters can be specified by:

```
nonbonded_2body_potentials {
  qtaper: particle_type1=N,      particle_type2=B,  \
  ↪   r_0=2.35, c=-17.9986;
  qtaper: particle_type1=N,      particle_type2=Si, \
  ↪   r_0=2.6,  c=-20.4023;
};
```

Particle types may be set to any particle you specified in the particle section of this file. Note that the coulomb-force-constant `epsilon0inv` has to be specified, see also section 13.5.4.

11.3.4 RSL2

$$U = \frac{a_{ij}}{1 + \exp\left(b_{ij}(r_{ij} - c_{ij})\right)}$$

```
nonbonded_2body_potentials      {
      rsl2:   particle_type1=H,
              particle_type2=H,
              r_cut=1.6176471,
              a=98.384782,
              b=20.4, c=0.64705882;
      };
```

Particle types may be set to any particle you specified in the particle section of this file.

11.3.5 Stilling-Weber

The Stillinger-Weber potential model [88] implemented in Tremolo-X reads as:

$$U = \sum_{i<j} u_2^{ij}(r_{ij}) + \sum_{i<j<k} \left(u_3^{ijk}(r_{ji}, r_{jk}, \theta_{ijk}) + u_3^{jik}(r_{ij}, r_{ik}, \theta_{jik}) + u_3^{ikj}(r_{ki}, r_{kj}, \theta_{ikj}) \right),$$

where θ_{ijk} is the angle between $\vec{r}_{ji} := \vec{r}_i - \vec{r}_j$ and $\vec{r}_{jk} := \vec{r}_k - \vec{r}_j$.

Stilling-Weber 2body

TODO: Double check potential (taken from tooltip): The Stillinger-Weber two-body potential reads as:

$$u_2^{ij}(r) = A \cdot (B \cdot r^{-p} - 1) \cdot \exp\left(\frac{\gamma}{r - r_{cut}}\right)$$

Note that all parameters A, B, γ and r_{cut} depend on the particle types of particle pair (i, j) and are symetric, i.e. e.g. $A_{ij} = A_{ji}$.

The parameters can bespecified by:

```
nonbonded_2body_potentials      {
  stiwe:  particle_type1=Si,
          particle_type2=Si,
          r_cut=3.77118,
          p=4,
          A=15.2855528754191,
          B=11.6031922833963,
          gamma=2.0951;
        };
```

Particle types may be set to any particle you specified in the particle section of this file.

Stilling-Weber 3body

TODO: Double check potential (taken from tooltip): The three-body potential reads as:

$$u_3^{type1}(r_{ji}, r_{jk}, \theta) = (\cos(\theta) - \cos(\theta_0))^2 \cdot \lambda \cdot \exp\left(\frac{\gamma_0}{r_{ji} - r_0} + \frac{\gamma_1}{r_{jk} - r_1}\right)$$

$$u_3^{type2}(r_{ji}, r_{jk}, \theta) = (\cos(\theta) - \cos(\theta_0)) \cdot \sin(\theta) \cdot \cos(\theta) \cdot \lambda \cdot \exp\left(\frac{\gamma_0}{r_{ji} - r_0} + \frac{\gamma_1}{r_{jk} - r_1}\right)$$

Note that the parameters γ_0, γ_1, r_0, r_1 depend on the particle types of the particle triple (i, j, k), where symmetry $(i, j, k) = (k, j, i)$ is assumed.

The parameters can be specified by:

```
stiwe3bodys     {
        stiwe3body: particle_type1=Si,
                    particle_type2=Si,
                    particle_type3=Si,
                    r_0=3.77118,
                    gamma_0=2.51412,
                    lambda=45.5343,
                    r_1=3.77118,
                    costheta0=-0.333333333333,
                    gamma_1=2.51412,
                    type=1;
                  };
```

Table 11.1: Different parameter sets for the Stillinger-Weber potenial model

Elements	References	`.potentials` file
Si	[88]	`stiwe-Si_1985`
Mo, S	[44]	`stiwe-MoS_2013`

Particle types may be set to any particle you specified in the particle section of this file, where `particle_type2` is the center particle.

Parameters

Examples of parameter sets for the Stillinger-Weber potential model are given in Table **??**.

11.3.6 Sutton-Chen

The so called Sutton-Chen potential for fcc metals was introduced in [92] and for metal alloys in [84]. The energy is composed of

$$U = \sum_i \left[\sum_{j>i} \left(\frac{\epsilon_{ij}\sigma_{ij}}{r_{ij}} \right)^{n_{ij}} - \epsilon_i c_i \sqrt{\rho_i} \right], \qquad \rho_i = \sum_{j \neq i} \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{m_{ij}},$$

where $\epsilon_i = \epsilon_{ii}$, $m_i = m_{ii}$ and $n_i = n_{ii}$.

User entries

```
nonbonded_2body_potentials      {
      suttonchen:     particle_type1=Ag,
                      particle_type2=Ag,
                      epsilon=0,
                      sigma=1.0,
                      r_cut=0,
                      m=6,
                      n=12,
                      c=0;
      suttonchen:     particle_type1=Ag,
                      particle_type2=Au,
                      r_cut=0,
                      m=6,
                      n=12;
                    };
```

Particle types may be set to any particle you specified in the particle section of this file. Note that in the case of `particle_type1==particle_type2`, the parameters `r_cut`, `m`, `n`, `c` are necessary to be specified. In the case of

Table 11.2: Different parameter sets for the Sutton-Chen potenial model

Elements	References	`.potentials` file
Ni, Cu, Rh, Pd, Ag, Ir, Pt, Au, Pb, Al	[84]	`suttonchen-original-1991`

Ni, Cu, Rh, Pd, Ag, Ir, Pt, Au	[47]	`suttonchen-original-1998`
		`suttonchen-classical-1998`
		`suttonchen-quantum-1998`

Ni, Cu, Ag, Au, Pt, Rh	[22]	`suttonchen-NiCuAgAuPtRh-1999`
Fe	[15]	`suttonchen-Fe-2000`
Ni, Al	[46]	`suttonchen-NiAl-2008`

`particle_type1`!=`particle_type2`, no further parameters have to be specified, e.g.

```
suttonchen:     particle_type1=Ag,
                particle_type2=Au;
```

since a not given parameter is given by the respective mixing rule:

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j},\ \sigma_{ij} = \frac{\sigma_i + \sigma_j}{2},$$
$$m_{ij} = \frac{m_i + m_j}{2},\ n_{ij} = \frac{n_i + n_j}{2},\ r_{ij}^{cut} = \frac{r_i^{cut} + r_j^{cut}}{2},$$

Note that, if `epsilon` or `sigma` is not set within `sutttonchen`, then their values are taken from the particle data. Note finally that the parameter `c` does not need to be specified for pairs of atom types, since its unused in that case.

Parameters

Examples of parameter sets for the Sutton-Chen potential model are given in Table 11.2.

11.3.7 EAM

The so called Embedded-Atom Method potential for metals was introduced in [23, 24]. In the so-called EAM/alloy variant the energy is composed of

$$U = \sum_{i<j} \phi_{ij}(r_{ij}) + \sum_i F_i \left(\sum_{j \neq i} \rho_j(r_{ij}) \right) \tag{11.1}$$

where F is the embedding function, ρ is the electron density and ϕ a pair potential. Note that these functions are usually given by tabulated functions. There, exists a slightly modified so-called EAM/FS variant which allows for

pair dependent ρ like in the case of Finnis-Sinclair potential model [31]. The energy is given by

$$U = \sum_{i<j} \phi_{ij}(r_{ij}) + \sum_i F_i \left(\sum_{j \neq i} \rho_{ij}(r_{ij}) \right) \tag{11.2}$$

User entries

For the EAM/alloy variant (11.1) a potential file in the EAM/alloy setfl format (see e.g. `http://www.ctcms.nist.gov/potentials`) has to be specified:

```
eam       {
  setfl:            file="Fe-Ni.eam.alloy";
};
```

For the EAM/FS variant (11.2) a potential file in the EAM/alloy setfl format (see e.g. `http://www.ctcms.nist.gov/potentials`) has to be specified:

```
eam       {
  fssetfl:            file="Fe-C_Hepburn_Ackland.eam.fs";
};
```

Particle types are given in the 4th line of the setfl potential file and may be set to any particle you specified in the particle section of this file.

Parameters

Examples of parameter sets for the EAM potential model are given in Table 11.3 and Table 11.4. Note that these parameter sets were not created by Fraunhofer SCAI, but compiled from public available source for your convenience. As with any potential set, it is the responsibility of the user to check whether a given parameter set is suitable for the intended application. We list the sources of all compiled parameter sets in the table, as well as in the respective potential file.

In particular, see also the web site `http://www.ctcms.nist.gov/potentials`.

TODO: Note that potential files by Zhou(2004)[103] are all single species files, which can be used as they are, but can also be combined. For further information on viability of these combinations and how to produce them, see `http://www.ctcms.nist.gov/potentials/Zhou04.html`. *This is created and maintained by NIST Interatomic Potentials Repository[13] and in particlur is not developed by Fraunhofer SCAI, nor is it sold by Fraunhofer SCAI or their distribution in other ways restricted. Fraunhofer SCAI is not responsible for functionality or maintenance of these potentials.*

Table 11.3: Selection of different parameter sets for the EAM potenial model.

Elements	References	`.potentials` file
Ag	[7]	`eamfs-Ag-Ackland-1987`
	[99]	`eamalloy-Ag-Williams-2006`
	[103]	`eamalloy-Ag-Zhou-2004`
Ag, Cu	[99]	`eamalloy-AgCu-Williams-2006`
	[101]	`eamalloy-AgCu-Wu-2009`
Al	[69]	`eamalloy-Al-Mishin-1999`
	[89]	`eamfs-AlMDSL-Sturgeon-2000`
	[106]	`eamalloy-Al-Zope-2003`
	[54]	`eamalloy-Al-Liu-2004`
	[103]	`eamalloy-Al-Zhou-2004`
	[63]	`eamfs-Al1-Mendelev-2008`
	[100]	`eamalloy-Al-Winey-2009`
Al, Fe	[60]	`eamfs-AlFe-Mendelev-2005`
Al, H, Ni	[11]	`eamalloy-AlHNi-Angelo-1995`
Al, Ni	[70]	`eamalloy-AlNi-Mishin-2002`
	[72]	`eamalloy-AlNi-Mishin-2004`
	[83]	`eamalloy-AlNi-Mishin-2009`
Al, Mg	[55]	`eamalloy-AlMg-Liu-1997`
	[61]	`eamfs-AlMG-Mendelev-2009`
Al, Mn, Pd	[85]	`eamalloy-AlMnPd-Schopf-2012`
Al, Pb	[49]	`eamalloy-AlPb-Landa-2000`
Al, Ti	[106]	`eamalloy-AlTi-Zope-2003`
Au	[7]	`eamfs-Au-Ackland-1987`
	[103]	`eamalloy-Au-Zhou-2004`
	[37]	`eamalloy-Au-Grochola-2005`
C, Fe	[41]	`eamfs-CFe-Hepburn-2008`
Co	[103]	`eamalloy-Co-Zhou-2004`
	[82]	`eamalloy-Co-PujaPun-2012`
Cu	[7]	`eamfs-Cu-Ackland-1987`
	[71]	`eamalloy-Cu-Mishin-2001`
	[103]	`eamalloy-Cu-Zhou-2004`
	[63]	`eamfs-Cu-Mendelev-2008`
Cu, Fe, Ni	[19]	`eamalloy-CuFeNi-Bonny-2009`
Cu, Zr	[65]	`eamfs-CuZr-Mendelev-2009`
	[66]	`eamfs-CuZr-Mendelev-2007`
Fe	[5]	`eamfs-Fe-Ackland-1997`
	[62]	`eamfs-Fe2-Mendelev-2003`
	[62]	`eamfs-Fe5-Mendelev-2003`
	[103]	`eamalloy-Fe-Zhou-2004`
Fe, Ni	[18]	`eamalloy-FeNi-Bonny-2009`
	[68]	`eamalloy-FeNi-MeyerEntel-1995`
Fe, P	[6]	`eamfs-FeP-Ackland-2004`
Fe, V	[67]	`eamfs-FeV-Mendelev-2007`
H, Pd	[104]	`eamalloy-HPd-Zhou-2007`
Mo	[103]	`eamalloy-Mo-Zhou-2004`
Mo, U, Xe	[86]	`eamfs-MoUXe-Smirnova-2013`
Mg	[103]	`eamalloy-Mg-Zhou-2004`
	[91]	`eamfs-Mg-Sun-2006`
Nb	[29]	`eamalloy-Nb-Fellinger-2010`

Table 11.4: Selection of different parameter sets for the EAM potenial model.

Elements	References	`.potentials` file
Ni	[7]	`eamfs-Ni-Ackland-1987`
	[69]	`eamalloy-Ni-Mishin-2009`
	[103]	`eamalloy-Ni-Zhou-2004`
	[64]	`eamfs-Ni1-Mendelev-2012`
Ni, Zr	[64]	`eamfs-NiZr-Mendelev-2012`
Pb	[103]	`eamalloy-Pb-Zhou-2004`
Pd	[103]	`eamalloy-Pd-Zhou-2004`
Pt	[103]	`eamalloy-Pt-Zhou-2004`
Ru	[32]	`eamfs-Ru-Fortinin-2008`
Ta	[52]	`eamalloy-Ta-Li-2003`
	[103]	`eamalloy-Ta-Zhou-2004`
Ti	[8]	`eamfs-Ti-Ackland-1992`
	[103]	`eamalloy-Ti-Zhou-2004`
U	[87]	`eamalloy-U-Smirnova-2012`
W	[103]	`eamalloy-W-Zhou-2004`
Zr	[103]	`eamalloy-Zr-Zhou-2004`

11.3.8 Tersoff

Introduction

The so called Tersoff type II potential for silicon was introduced in [94]. A slightly modified parameter set produces the so called Tersoff type III potential for silicon [93]. Specifically for C, Si and Ge a general Tersoff potential has been constructed in [96].

Multicomponent Potential

The energy is composed of

$$E = \sum_i E_i = \frac{1}{2} \sum_{i \neq j} \underbrace{f_C(r_{ij}) \left(\chi_{Rij} f_R(r_{ij}) + b_{ij} f_A(r_{ij}) \right)}_{V_{ij}},$$

where the *repulsive* and the *attractive* terms

$$\begin{aligned} f_R(r_{ij}) &= A_{ij} \exp(-\lambda_{ij} r_{ij}) \\ f_A(r_{ij}) &= -B_{ij} \exp(-\mu_{ij} r_{ij}) \end{aligned}$$

and the smoothed cut off function

$$f_C(r_{ij}) = \begin{cases} 1 & r_{ij} < R_{ij} \\ \frac{1}{2} + \frac{1}{2} \cos\left(\frac{\pi (r_{ij} - R_{ij})}{S_{ij} - R_{ij}} \right) & R_{ij} < r_{ij} < S_{ij} \\ 0 & r_{ij} > S_{ij} \end{cases}$$

are symmetric in i and j. An equivalent formulation is

$$E = \sum_{i<j} \underbrace{f_C(r_{ij}) \left(\chi_{Rij} f_R(r_{ij}) + \underbrace{\frac{(b_{ij} + b_{ji})}{2}}_{\tilde{b}_{ij}} f_A(r_{ij}) \right)}_{\tilde{V}_{ij}}$$

In addition, the following holds:

$$\begin{aligned} b_{ij} &= \chi_{ij}(1 + \beta_{ij}^{n_{ij}} \zeta_{ij}^{n_{ij}})^{-\frac{1}{2n_{ij}}} \\ \zeta_{ij} &= \sum_{k \neq i,j} f_C(r_{ik}) \omega_{ijk} e^{\alpha_{ijk}^{m_{ijk}} (r_{ij} - r_{ik})^{m_{ijk}}} g(\theta_{ijk}) \\ g(\theta_{ijk}) &= 1 + \frac{c_{ik}^2}{d_{ik}^2} - \frac{c_{ik}^2}{d_{ik}^2 + (h_{ik} - \cos\theta_{ijk})^2}. \end{aligned}$$

The mixture rules for the parameters are as follows;

$$\begin{aligned} &\lambda_{ij} = \frac{\lambda_i + \lambda_j}{2},\ \mu_{ij} = \frac{\mu_i + \mu_j}{2} \\ &A_{ij} = \sqrt{A_i A_j},\ B_{ij} = \sqrt{B_i B_j},\ R_{ij} = \sqrt{R_i R_j},\ S_{ij} = \sqrt{S_i S_j}, \end{aligned} \tag{11.3}$$

$$\omega_{ijk} = \omega_{ik},\ \alpha_{ijk} = \alpha_{ik},\ m_{ijk} = m_{ik} \tag{11.4}$$

and in [96] it is set also $\alpha_{ijk} = \alpha_{ik} = 0$, $m_{ijk} = m_{ik} = 0$ and

$$\beta_{ij} = \beta_i,\ n_{ij} = n_i,\ c_{ij} = c_i,\ d_{ij} = d_i,\ h_{ij} = h_i, \tag{11.5}$$

Typically the special pair parameters χ_{ij} and ω_{ij} are symmetric in i and j as well (generally they are identical to 1, unless specified differently). Furthermore, the common notation of r_{ij} as the distance of i-j and θ_{ijk} as the angle enclosed by i-j and i-k is used.

For high energergetic simulations, one can modify the repulsive behaviour of the potential by using instead of V_{ij} the modified potential

$$\tilde{V}_{ij}(r) = (1 - F(r))V^{ZBL}(r) + F(r)V_{ij}(r), \tag{11.6}$$

where V^{ZBL} is the well-known Ziegler–Biersack–Littmark universal repulsive potential [105] and F is the Fermi function

$$F(r) = \frac{1}{1 + \exp(-b_f(r - r_f))}.$$

Alternatively one may just modufy the repulsive potential f_R by

$$\tilde{f}_R(r) = (1 - F(r))V^{ZBL}(r) + F(r)f_R(r). \tag{11.7}$$

Note that

$$V^{ZBL}(r) = \frac{e^2}{4\pi\epsilon_0}\frac{Z_1 Z_2}{r}\phi\left(\frac{r}{a}\right)$$

with

$$a = \frac{0.8854 a_0}{Z_1^{0.23} + Z_2^{0.23}}$$

and

$$\phi(x) = 0.1818e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x}.$$

Here, a_0 is usually the bohradius, i.e. 0.529Å. Note that the coulomb-force-constant `epsilon0inv` has to be specified, see also section 13.5.4.

User entries

To set a Tersoff particle one should use:

```
tersoff {
        tersoffparticle:        particle_type=C,
                                A=1393.6,
                                B=346.74,
                                lambda=3.4879,
                                mu=2.2119,
                                beta=1.5724e-07,
                                n=0.72751,
                                c=38049,
                                d=4.3484,
                                h=-0.57058,
                                chiR=1.0,
                                chi=1.0,
                                omega=1.0,
                                alpha=0.0,
                                m=0,
                                R=1.8,
                                S=2.1;
        };
```

To mix particle types, one may use:

```
        tersoffmixit:           particle_type1=C,
                                particle_type2=Si,
                                chiR=1.0,
                                chi=0.9776,
                                omega=1.0,
                                alpha=0.0,
                                m=0;
```

which applies the mixing rules (11.3), (11.5) and (11.4).

Alternative one may use:

```
tersoffoffdiag:          particle_type1=C,
                         particle_type2=Si,
                         A=1597.311,
                         B=395.145,
                         lambda=2.9839,
                         mu=1.97205,
                         chiR=1.0,
                         chi=1.0,
                         omega=1.0,
                         alpha=0.0,
                         m=0,
                         R=2.21,
                         S=2.51;
```

which applies only the mixing rules (11.5) and (11.4).

Alternative one can use:

```
tersoffoffdiag2:         particle_type1=C,
                         particle_type2=Si,
                         A=1779.36144,
                         B=225.189481,
                         lambda=3.26563307,
                         mu=1.76807421,
                         beta=1.0,
                         n=1.0,
                         c=273987,
                         d=180.314,
                         h=-0.68,
                         chiR=1.0,
                         chi=1.0,
                         omega=0.011877,
                         alpha=0.0,
                         m=0,
                         R=2.2,
                         S=2.6;
```

which just applies the mixing rules (11.4).

The parameters ω_{ijk}, α_{ijk} and m_{ijk} can also be specified for each triple by

```
tersofftriple:           particle_type1=C,
                         particle_type2=Si,
                         particle_type3=Si,
                         omega=0.011877,
                         alpha=0.0,
                         m=0;
```

The modified Tersoff/ZBL potential can be used by:

```
tersoffzbl:              particle_type1=Si,
                         particle_type2=Si,
                         type=2,
```

```
                                    Z1=14,
                                    Z2=14,
                                    a0=0.529,
                                    rf=0.95,
                                    bf=14;
```

Here, to use variant (11.6) set `type=2` and to use variant (11.7) set `type=1`. Here, a_0 is usually the bohradius, i.e. , i.e. 0.529Å. Note that the coulomb-force-constant `epsilon0inv` has to bespecified, see also section 13.5.4.

Particle types may be set to any particle you specified in the particle section of this file.

Parameters

Examples of parameter sets for the Terosoff potential model are given in Table 11.5.

11.3.9 Miscellaneous models

There are miscellaneous combinations of potential terms for several systems given in the potentials directory. For example a combination of potential terms given in section 11.5 is by Marian and Gastreich in [56] to model Si/B/N(H) compounds.

An overview is of miscellaneous potential models is given in Table 11.6.

11.4 Bonded Potentials

TODO: explicit range of sum TODO: up to date?

$$\begin{aligned} E &= E_{LJ} + E_C + E_{bonded} \\ E_{bonded} &= E_b + E_\theta + E_\phi + E_\psi \\ E_b &= \sum k_b(r - r_0)^2 \\ E_\theta &= \sum k_\theta(\theta - \theta_0)^2 \\ E_\phi &= \sum k_\phi(1 + \cos(n\phi - \delta)) \\ E_\psi &= \sum k_\psi(\psi - \psi_0)^2 \end{aligned}$$

TODO: In comment

11.4.1 Bonds

Harmonic Bonds

$$U = k_B(r - r_0)^2$$

Table 11.5: Different parameter sets for the Tersoff potenial model

Elements	References	`.potentials` file
Al, As, Ga	[79]	`tersoff-AlGaAs_2000`
Al, N	[48]	`tersoff-AlN_2008`
Al, N, O	[80]	`tersoff-AlNO_2009`
	[80]	`tersoff-AlNO_2009b`
As, In	[39]	`tersoff-InAs_2008`
As, Ga	[10]	`tersoff-GaAs_2002`
	[39]	`tersoff-GaAs_2008`
	[30]	`tersoff-GaAs_2011`
As, Ga, In	[79]	`tersoff-InGaAs_2000`
Au	[14]	`tersoff-Au_2012`
B, C, N	[57]	`tersoff-BNC_2000`
B, N	[73]	`tersoff-BN_2003`
B, N, O	[80]	`tersoff-BNO_2009`
	[80]	`tersoff-BNO_2009b`
B, N, Si	[58]	`tersoff-SiBN_2001`
Be, C, H	[17]	`tersoff-BeCH_2009`
Be, H	[17]	`tersoff-BeH_2009`
Be, W	[16]	`tersoff-BeW_2010`
C	[53]	`tersoff-C_2010`
C, Fe	[40]	`tersoff-FeC_2009`
C, H	[45]	`tersoff-CH_2005`
C, H	[45, 53]	`tersoff-CH_2010`
C, H, W	[45]	`tersoff-WCH_2005`
	[45, 27]	`tersoff-WCH_2005b`
C, Pt	[9]	`tersoff-PtC_2002`
C, Si	[96, 97]	`tersoff-SiC_1989`
	[98]	`tersoff-SiC_1994`
	[26]	`tersoff-SiC_1998`
	[27]	`tersoff-SiC_2005`
Cu, Fe	[43]	`tersoff-FeCu_2012`
Er, H	[81]	`tersoff-ErH_2011`
Fe	[74]	`tersoff-Fe_2007`
Fe, Pt	[75]	`tersoff-FePt_2007`
Ga, N	[78]	`tersoff-GaN_2003`
Ga, N, O	[80]	`tersoff-GaNO_2009`
	[80]	`tersoff-GaNO_2009b`
Ge, Si	[96, 97]	`tersoff-SiGe_1989`
H, N, Si	[25]	`tersoff-SiNH_2009`
H, W	[51]	`tersoff-WH_2011`
In, N	[48]	`tersoff-InN_2008`
In, N, O	[80]	`tersoff-InNO_2009`
	[80]	`tersoff-InNO_2009b`
O	[28]	`tersoff-O_2006`
O, Si	[76]	`tersoff-SiO_2007`
O, Zn	[28]	`tersoff-ZnO_2006`
Pt	[9]	`tersoff-Pt_2002`
Si	[95]	`tersoff-SiC_1988`
	[27]	`tersoff-Si_2005`
Zn	[28]	`tersoff-Zn_2006`

Table 11.6: Miscellaneous potential models given by combinations of various potential terms.

Elements	References	`.potentials` file
Si, B, N, H	[56]	`mg-SiBNH_2000`
Si, B, N	[34]	`mg-SiBN_2003spline`
		`mg-SiBN_2003spme`

Note: In this particular instance k_B does NOT denote the Boltzmann constant, but the bond force constant.

```
bonds   {
        bond:   particle_type1=H,
                particle_type2=H,
                bond_type=harmonic,
                k_b=22208.534,
                r_0=0.28147059,
                k=0,
                e_0=0;
        };
```

Morse Bonds

$$U = E_0 \cdot \left(\left[1 - \exp^{-k(r-r_0)} \right]^2 - 1 \right)$$

```
bonds   {
        bond:   particle_type1=H,
                particle_type2=H,
                bond_type=morse,
                k_b=22208.534,
                r_0=0.28147059,
                k=0,
                e_0=0;
        };
```

11.4.2 Angles

Harmonic Angles

$$U = k_\theta \cdot (\theta - \theta_0)^2$$

```
angles  {
        angle:  particle_type1=H,
                particle_type2=H,
                particle_type3=H,
                angle_type=harmonic,
```

```
                k_th=234.80792,
                theta_0=104.52,
                k_1=168.07514,
                k_2=92.444488,
                k=0,
                cos_theta_0=0,
                k_ub=792.20868,
                S_0=0.75323529;
        };
```

Harmonic cosine Angles

$$U = \frac{k}{2} \cdot (\cos(\theta) - \cos(\theta_0))^2$$

Song Hi Lee Angles

$$U = k_1 \cdot (\theta - \theta_0)^2 - k_2 \cdot (\theta - \theta_0)^3$$

Harmonic Urey-Bradley Angles

$$U = k_\theta \cdot (\theta - \theta_0)^2 + k_{ub} \cdot (S - S_0)^2$$

11.4.3 Diederpotential (Torsion)

These potentials apply to four-body structures of atoms. Proper torsions are those which are used on "linear" configurations. For non-linear four-body structures (e.g. three atoms bound to one atom in the center) so called improper torsions are used. (see section 11.4.4) *TODO: torsion image?*

Torsion cosine formula

$$E_\phi = \sum_{i=1}^{mult} k_i(1 + \cos(n_i\phi - \delta_i))$$

with $1 \leq n_i \leq 6$, mult < 6, ϕ the angle between the plains spanned by (x_i, x_j, x_k) and (x_j, x_k, x_l), for the case (x_i, x_j, x_k, x_l) are connected in a chain.

With $r_i = x_j - x_i, \quad r_j = x_k - x_j, \quad r_k = x_l - x_k$ we can write

$$\cos(\phi) = \frac{(r_i \times r_j) \cdot (r_j \times r_k)}{|r_i \times r_j||r_j \times r_k|}$$

TODO: Formula with scalar products, so user can decide it is the same?

```
torsions        {
        torsion:        particle_type1=H,
                        particle_type2=H,
                        particle_type3=H,
                        particle_type4=H,
                        torsion_type=cosine,
                        mult=5,
                        k_1=0,  n_1=0,  delta_1=0,
                        k_2=0,  n_2=0,  delta_2=0;
                        k_3=0,  n_3=0,  delta_3=0;
                        k_4=0,  n_4=0,  delta_4=0;
                        k_5=0,  n_5=0,  delta_5=0;
                };
```

For the computation of the diederpotential and the differentiation the angle phi does not always need to be computed. Singularities in the derivative are caught by the program.

Alternative Torsion (polynomial)

$$U = k \sum_{i=0}^{5} a_i \cos(\phi)^i \tag{11.8}$$

```
torsions        {
        torsion:        particle_type1=H,
                        particle_type2=H,
                        particle_type3=H,
                        particle_type4=H,
                        torsion_type=polynomial,
                        k=0,
                        a0=0,
                        a1=0,
                        a2=0,
                        a3=0,
                        a4=0,
                        a5=0,
                };
```

11.4.4 Improper Torsion

Improper torsions apply to quadruples of atoms with a non-linear configuration (e.g. three atoms connected to one in the center). Torsions for linear configurations are handled in section 11.4.3

TODO: computation does not interest user!?

TODO: but design of improper torsion does.

$$E_\psi = k_\psi(\psi - \psi_0)^2$$

```
impropers       {
        improper:       particle_type1=H,
                        particle_type2=H,
                        particle_type3=H,
                        particle_type4=H,
                        improper_type=harmonic,
                        k_psi=85.384684,
                        psi_0=1;
        };
```

Equivalently we use the first terms of the Taylor series for small angles for improper torsion terms to get a numerically stable computation:

$$\begin{aligned} E &\simeq k\psi^2 \\ \frac{\partial E}{\partial \cos\psi} &\simeq 2k\left(1+\frac{\psi^2}{6}\left(1+\frac{7\psi^2}{60}\right)\right) \end{aligned}$$

For large angle (starting about $6\pi/180$) we simply compute the force by

$$\frac{\partial E}{\partial \cos\psi} = -2k\frac{\psi-\psi_0}{\sin\psi}$$

The derivative is done as for the diederpotential. Improper Torsion does not apply to atoms in a chain, but considers a "star" of four atoms (x_i, x_j, x_k, x_l) with x_i in center, (x_j, x_k, x_l) each sharing a bond with x_i. The angle ψ is between the plains spanned by (x_i, x_j, x_k) and (x_j, x_k, x_l). In addition, some force field model consider improper torsion contributions for atoms not directly bonded.

Alternative Improper Torsion

$$E_\psi = k_\psi\left(\sin(\psi-\psi_0)\right)^2$$

```
impropers       {
        improper:       particle_type1=H,
                        particle_type2=H,
                        particle_type3=H,
                        particle_type4=H,
                        improper_type=squaredsine,
                        k_psi=20.0001,
                        psi_0=1;
        };
```

11.5 Tapered Potentials

Tapered potentials, similary to Lennard-Jones with splines, don't use a hard cutoff but instead are multiplied with a decaying fifth order spline for particle distances r: $x_i < r \leq x_o \ \wedge \ U(k) = 0, \ k \geq x_o$. This way the energy is preserved despite the potential cutoff.

11.5.1 BMHFT

The non-coulombic term of the Born-Mayer-Huggins-Fumi-Tosi-Potential as used in [12]. *TODO: Cite actual source instead of "usage example"?*

$$U = A \cdot e^{B\,(\sigma - r)} - \frac{C}{r^6} - \frac{D}{r^8}$$

Listing 11.1: Example taken from NaCl test case using `kcalpermole` units.

```
tbtapered_potentials {
    tbtaper: x_i = 20.0, x_o=23.0;
    tbtosifumi: particle_type1=Na, particle_type2=Cl, \
    ↪   A=4.86167, B=3.1546, C=161.097, D=199.933, \
    ↪   sigma=2.755;
};
```

11.5.2 Morse

TODO: Double check potential (taken from tooltip):

$$U = E_0 \cdot \left(\left[1 - e^{-k(r - r_0)} \right]^2 - 1 \right)$$

```
tbtapered_potentials    {
        tbtaper:        x_i=1.4705882,  x_o=1.7647059;
        tbmorse:        particle_type1=H,
                        particle_type2=H,
                        r_0=0.37428235,
                        k=10.06196,
                        e_0=364.60871;
        };
```

11.5.3 Damped Dispersion

TODO: Double check potential (taken from tooltip):

$$U = \frac{-C_b}{r^6 \left(1 - e^{(-b_b \cdot r)} \sum_{k=0}^{6} \frac{(b_b \cdot r)^k}{k!} \right)}$$

```
tbtapered_potentials     {
          tbtaper:                    x_i=1.4705882,   \
          ↪    x_o=1.7647059;
          dampeddispersion:           particle_type1=H,
                                      particle_type2=H,
                                      C_b=1.2712611,
                                      b_b=9.75375;
          };
```

11.5.4 General

General Type I *TODO: Double check potential (taken from tooltip):*

$$U = \frac{A}{r} \cdot \exp(\frac{-r}{\rho})$$

```
tbtapered_potentials     {
                    tbtaper:                  x_i=1.4705882,   \
                    ↪    x_o=1.7647059;
                    tbtaperedgeneral1:        particle_type1=H,
                                              particle_type2=H,
                                              A=1,
                                              rho=1;
          };
```

General Type II *TODO: Double check potential (taken from tooltip):*

$$U = \frac{A}{r^2} \cdot \exp(\frac{-r}{\rho}) - \frac{C}{r}$$

```
tbtapered_potentials     {
                    tbtaper:                  x_i=1.4705882,   \
                    ↪    x_o=1.7647059;
                    tbtaperedgeneral2:        particle_type1=H,
                                              particle_type2=H,
                                              A=0,
                                              rho=0,
                                              C=0;
          };
```

General Type III

$$U = A \cdot e^{B\,(\sigma - r)}$$

```
tbtapered_potentials     {
                  tbtaper:                    x_i=1.4705882,   \
                  ↪    x_o=1.7647059;
                  tbtaperedgeneral3:          particle_type1=H,
                                              particle_type2=H,
                                              A=0,
                                              B=0,
                                              sigma=0;
          };
```

11.5.5 Buckingham

TODO: Double check potential (taken from tooltip):

$$U = A \cdot \exp(\frac{-r}{\rho})$$

```
tbtapered_potentials     {
          tbtaper:          x_i=1.4705882,   x_o=1.7647059;
          tbbuckingham:     particle_type1=H,
                            particle_type2=H,
                            A=1,
                            rho=1;
          };
```

11.5.6 BN 3 body

TODO: Double check potential (taken from tooltip):

$$U = k \cdot \exp\left(\frac{r_{ij}}{\rho_1} - \frac{r_{ik}}{\rho_2}\right) \cdot \frac{((\theta_0 - \pi)^2 - (\theta - \pi)^2)^2}{8(\theta_0 - \pi)^2}$$

```
tbtapered_potentials     {
          tbtaper:          x_i=1.4705882,   x_o=1.7647059;
          bn3body:          particle_type1=H,
                            particle_type2=H,
                            particle_type3=H,
                            bn3bodyentry=VESSAL,
                            k=2662095.9,
                            theta_0=118.864,
                            rho1=0.093416765,
                            rho2=0.093416765,
                            rmax1=0.79411765,
                            rmax2=0.79411765;
          };
```

Chapter 12

External Forces

Tremolo-X allows to assign outer forces/constraints on individual or groups of particles or on regions within the simulation domain.

As external influence on specific particles, two types of forces/constraints are available, "freezing" the particles into their place and applying a fixed force along an arbitrary vector. *TODO: Tether force is not implemented yet.* These forces/constraints can be modified during the runtime of a simulation, by changing their parameters or simply switching them on/off. *TODO: Timeline is out of order?*

This allows to establish potential geometries by placing fixed particles or to simulate outer force fields such as electric potentials. Force time-lines are specified to apply to particle types in `$PROJECTNAME.external`, which have to be matched to individual particles in `$PROJECTNAME.exttypes`.

Furthermore, you can assign repulsive potentials to zylindrical and spherical regions in the simulation domain. This may either assist in shaping the simulation domain or enable the creation of "pockets" in an otherwise homogenous material matrix during the preprocessing, in order to insert functional structures for further simulation.

For the syntax see section 13.6.

Chapter 13

Essential Files - Input

Each simulation requires the presence of five files which hold the various input data. Those files are required to have the same base (with the exception of the .tremolo file, see there) name, which we call the project name and are distinguished by a particular suffix. The files are:

- `$PROJECTNAME.tremolo` (13.1)
- `$PROJECTNAME.potentials` (13.2)
- `$PROJECTNAME.validates` (13.3)
- `$PROJECTNAME.data` (13.4)
- `$PROJECTNAME.parameters` (13.5)

In the following subsections we describe the options to be set within each file in detail. As a general rule, comments can be added to each file, by starting a line with the #-Symbol and may be placed almost anywhere. The same is true for empty lines. Most options can also be placed in any order, however there are some, where order is relevant, since they influence each other and even less, which require a specific order simply for the parser to function properly. Those exceptions are mentioned at the descriptions below.

Additionally there are two optional files, `$PROJECTNAME.external` and `$PROJECTNAME.exttypes`

Working with the GUI

When using the GUI the five mandatory files are automatically generated when creating a new project (which can be done via the drop-down menu or the corresponding "empty sheet" icon. When loading an existing project, all files must be present though, otherwise the project cannot be properly loaded. For the layout of the main options see figure 13.1.

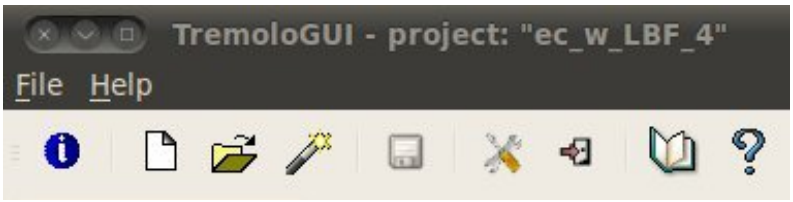


Figure 13.1: The menu bar in the Tremolo-X GUI. Buttons from left to right: Project information, new project, open project, creation wizard (not implemented), save project, GUI settings and preferences, GUI exit, GUI help

13.1 $PROJECTNAME.tremolo

This file could in theory also be named $ANYTHING.tremolo, with anything deviating from the projectname designated in the file.

```
global: defaultpath="./$BASENAME";
```

The default path is used to allow the use of project independent input files. When the program does not find a specific input file in the current directory, it checks the defaultpath for that file. Since the file may have a basename different from the current project, the basename needs to be specified at all times.

```
global: projectname="$PROJECTNAME";
```

Here you actually specify the prefix of all other input files present, which will also be the prefix of the generated output files.

```
global: systemofunits=$SYSTEM_OF_UNITS;
```

This parameter is quite essential as you can choose or customize the system of physical units you wish to use in a particular simulation. The choice made here affects almost any numeric parameter in the simulation and numeric values in other parameter files are expressed in units of the system chosen here. The options are:

```
kcalpermole, evolt, si, custom
```

	kcalpermole	evolt	si
length	1 Å	1 Å	1 m
time	$4.887\,19 \times 10^{-14}$ s	$1.018\,05 \times 10^{-14}$ s	1 s
mass	1 u	1 u	1 kg
current	$3.278\,32 \times 10^{-6}$ A	$1.573\,77 \times 10^{-5}$ A	1 A
temperature	503.556 K	11 604 K	$7.242\,963\,8 \times 10^{22}$ K

TODO: Check whether the current-unit in kcalpermole is ampere.

Note, that there is a relation between the scale of the temperature and the energy, which is derived from length, time and mass units. This must be computed individually by the user by dividing the energy by the boltzmann constant. Below we provide a sample calculation for the eV scaling:

```
# x     sigma = 1 angstrom = 1.e-10 m
# E     u* (sigma/alpha)^2 = epsilon = 1 eV = \
↪   1.6021765e-19 kg m^2 / s^2
# t     alpha = 1.0180505e-14 s
# m     u = 1.6605387e-27 kg
# T     epsilon / k_b = 11604.506 K
```

If the choice `custom` is selected, units must be specified separately according to the following scheme:

```
custom: lengthunit                = {angstrom, nm, m};
custom: lengthscalingfactor       = 1.0;
custom: timeunit                  = {fs, ps, s};
custom: timescalingfactor         = 1.0;
custom: massunit                  = {u, kg};
custom: massscalingfactor         = 1.0;
custom: currentunit               = {"e/s", A};
custom: currentscalingfactor      = 1.0;
custom: temperatureunit           = {K};
custom: temperaturescalingfactor  = 1.0;
```

By setting any of `$PHYSICAL_QUANTITYscalingfactor` to a value different than 1.0 all derived physical quantities are affected as well. Thus when using generated output values you need to remember to extract any scaling you introduced here. (Whether you need to divide or multiply, either the factor alone or its square depends on the specific quantity and its units.)

13.1.1 Optional input

```
global: comment="Some comment";
```

The comment will be printed to the standard output when the simulation is run.

13.1.2 Working with the GUI

In the GUI this file is controlled by the creation dialog and the first tab. In the creation dialog default path and $PROJECTNAME must be specified.

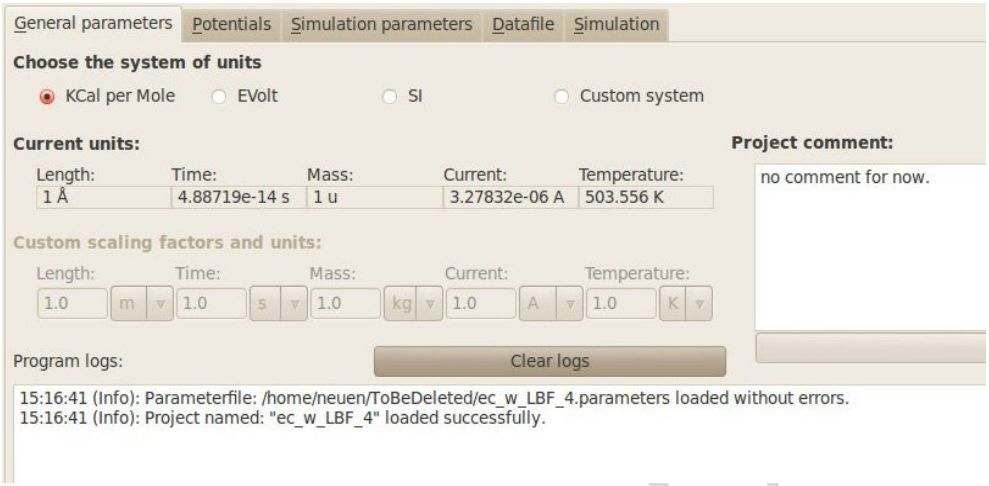
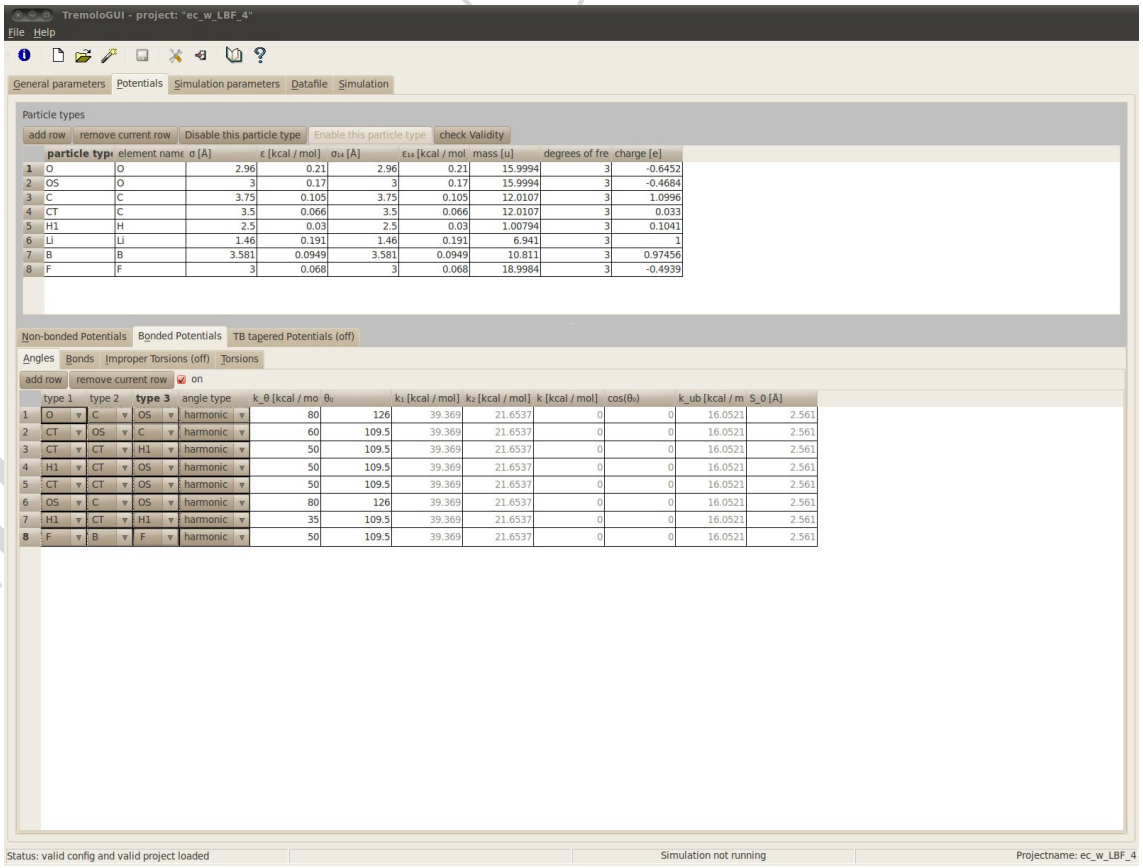


Figure 13.2:

13.2 $PROJECTNAME.potentials



The potentials file is responsible for all parameters, which affect the forces

between atoms in the simulation. It begins with a list of all particle types present in the ensemble, together with some particles which are used more than once: *TODO: layout*

```
particles       {
particle: particle_type=$P1, element_name=$NAME1, \
↪    sigma=1, epsilon=1, sigma14=1, epsilon14=1,     \
↪    mass=1,     free=3, charge=0;
particle: particle_type=$P2, element_name=$NAME1, \
↪    sigma=3, epsilon=1, sigma14=3, epsilon14=0.17, \
↪    mass=15.9, free=3, charge=-0.4684;
};
```

The file then lists the potentials, grouped by the type of potential. Each particle requires a specific set of parameters, which, again depending on the type of potential, need to be listed for all pair types (tripled types, ...) of particles to which it shall apply. *TODO: layout*

```
nonbonded_2body_potentials       {
   lennardjones:  particle_type1=$P1,  particle_type2=$P1, \
   ↪     r_cut=12.5;
   lennardjones:  particle_type1=$P1,  particle_type2=$P2, \
   ↪     r_cut=12.5;
   lennardjones:  particle_type1=$P2,  particle_type2=$P2, \
   ↪     r_cut=12.5;
   ljspline:      particle_type1=$P1,  particle_type2=$P1, \
   ↪     r_cut=12.5,  r_l=11;
   ljspline:      particle_type1=$P1,  particle_type2=$P2, \
   ↪     r_cut=12.5,  r_l=11;
   ljspline:      particle_type1=$P2,  particle_type2=$P2, \
   ↪     r_cut=12.5,  r_l=11;
        };
```

TODO: Make entry for each potential.

13.3 $PROJECTNAME.validates

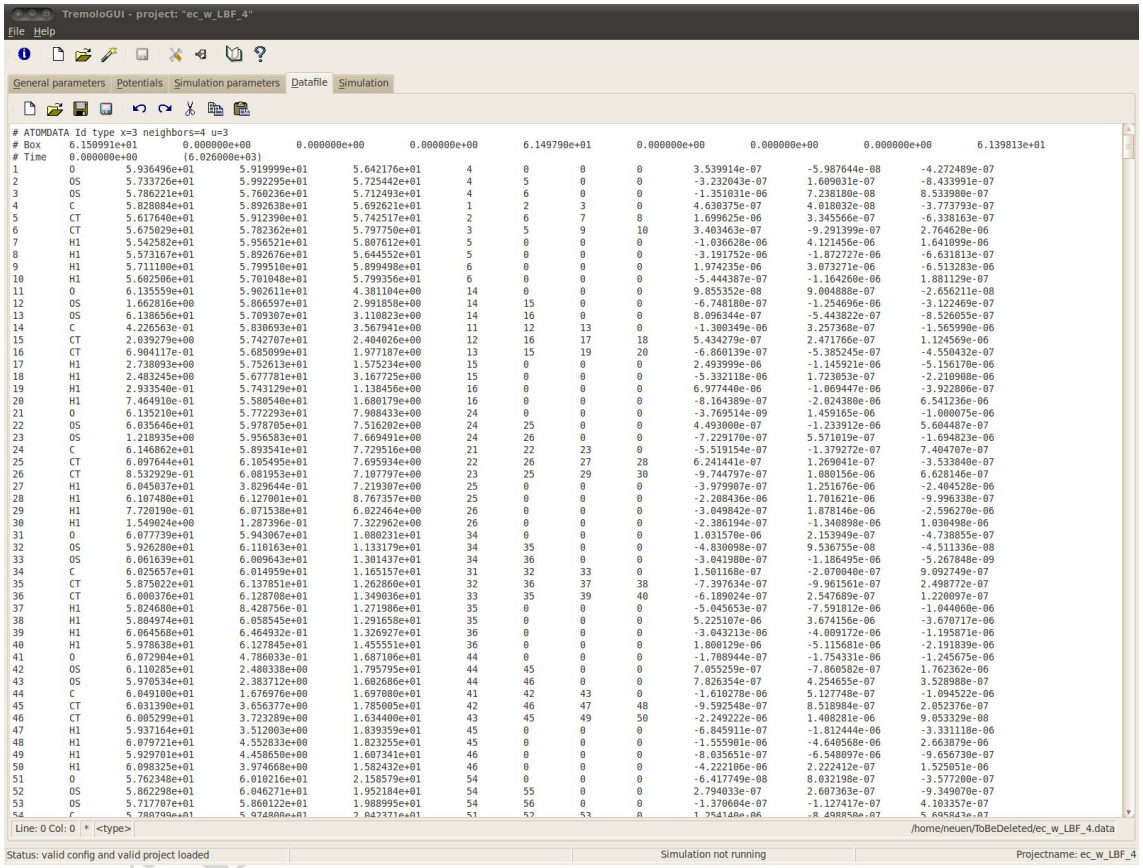
All particles and force fields detailed in the .potentials file need to be validated in this file, by setting their state to on. The reason for this is the converse action, this allows to selectively deactivate particle types and force fields, which shall not be used in this particular situation. This might be relevant, if you wish to experiment with several different (combinations of) force fields: Instead of writing an entire .potentials file for each combination, one such file with all desired forcefields is sufficient and you only need to make minor changes to the .validates file.

```
validate: particle_type=$P1, state=on;
validate: particle_type=$P2, state=off;

validate: force_type=lennardjones, state=off;
```

```
validate: force_type=ljspline, state=on;
```

13.4 $PROJECTNAME.data



This file contains all information about the initial configuration of the atoms. Always present is the line `ATOMDATA` line, which informs the parser about the type and order of the input. It may be followed by some optional `ATOMDATAADDOUT`, `INPUTCONV` and `OUTPUTCONV` lines.

```
ATOMDATA <record_entry_1> ... <record_entry_n>
<record_entry>: <dataname>[=<n>]
<dataname>      : x | xs | u | F | stress | Id | charge | \
↪    GroupMeasureTypeNo | type | neighbors | extType | \
↪    imprData | name | resName | chainID | resSeq | \
↪    occupancy | tempFactor | segID | Charge
```

In the example below the line specifies the options `Id` and `type`, which occupy a single column each. This is followed by the particle positions `x`, which require a three column wide vector, neighbors with a four column vector and finally velocities `u`, which use three columns again. Note that all options are case sensitive. *TODO: layout*

```
# ATOMDATA Id type x=3 neighbors=4 u=3
```

```
1       O       5.936075e+01    5.915340e+01    \
↪   5.643238e+01    4       0       0       0       \
↪   3.540887e-07    -5.986875e-08   -4.272449e-07
2       OS      5.734245e+01    5.991388e+01    \
↪   5.725724e+01    4       5       0       0       \
↪   -3.232931e-07   1.608824e-07    -8.433911e-07
3       OS      5.782428e+01    5.758512e+01    \
↪   5.712316e+01    4       6       0       0       \
↪   -1.351403e-06   7.237250e-08    8.533900e-07
4       C       5.826889e+01    5.890171e+01    \
↪   5.693151e+01    1       2       3       0       \
↪   4.631648e-07    4.017515e-08    -3.773758e-07
```

Possible options and their meaning are:

- `type` (string) This parameter must be present and it must match one of the particle types specified in the .potentials file.
- `Id` An identifier for each particle. When this parameter is used id numbers need to be assigned sequentially, otherwise there may be unexpected behavior. This is a requirement for the use of `neighbors`.
- `x=3` This are the carrtesian coordinates. The only sensible vector length is three.
- `xs=3` This are the scaled/fractional coordinates. The only sensible vector length is three. Note that `x=3` or `xs=3` must be given.
- `u=3` If one wishes to give particular velocities to individual particles, this can be specified here. In the case of a dynamics simulation it is set by default for the data output files. This way, you are able to restart the simulation from the finishing point. Supplying starting velocities also fixes an initial temperature and may conflict with the corresponding option 13.4.1, which takes precedence. The only sensible vector length is three.
- `neighbors=<n>` This parameter(-vector) specifies the particles to which the present one is bonded by specifying its `Id` number. The identifier `Id` numbers must exist and a particle may not be linked to itself. For non-bonds, enter a zero instead. The length of this vector can be any positive integer. This is relevant for bonded force fields.
- `charge` This parameter fixes an (electric) charge for this parameter. To have an effect, some Coulomb potential must be used. (This overwrites the charge set in the .potentials file for the particle type *TODO: test whether this is correct.*
- `F=3` This option allows to write out the current forces on each particle at the output time step. *TODO: It is advised to start with zero entries in these columns. The necessity should be verified.*

- `stress=<n>` This allows the output of this particles contribution to the total box stress. While not a physical stress value by itself (point particles do not have stress) this allows to determine local stress in a macro-structure by comparison among the particles. In the case you set the option in the parameter file to measure the local stress, it is set by default for the data output files.
- `imprData=<n>` When using improper torsion forces, use this as `neighbors` in order to specify neighbors for the respective force.
- `GrpTypeNo` For certain measurement routines, particles may be assigned to groups, which do not need to respect particle types. *TODO: specify relevant options, 4 indices?*

The information below belongs to PDB information, they are only parsed in and written out, they have no effect on simulation

- `extType` type number used for external forces *TODO: Is this actually a PDB option?*
- `name` particle name (used only for PDB data)
- `resName` residue name (used only for PDB data)
- `chainID` chain ID (used only for PDB data)
- `resSeq` residues sequence number (used only for PDB data)
- `occupancy` occupancy (used only for PDB data)
- `tempFactor` temperature factor (used only for PDB data)
- `segID` segment ID (used only for PDB data)
- `Charge` Charge (string) (used only for PDB data)

For the proper handling of PDB parameters, this feature must have been enabled at configure time of Tremolo-X. *TODO: Discuss enable by default.*

You can use an ATOMDATAADDOUT line in the same way to output additional data in the data output files, e.g. to write out forces in the above example:

```
# ATOMDATA Id type x=3 neighbors=4 u=3
# ATOMDATAADDOUT F=3
```

Note that one can also give the box matrix by a `# BOX` line in the `.data` file. In particular, a `# BOX` line given in the `.data` file will overwrite values from the `.parameter` file.

```
# BOX xx xy xz yx yy yz zx zy zz
```

13.4.1 Input conversion

The .data file may contain options, which influence the starting configuration of the sample. In the following list, `{0|1}` indicates that the option is activated by setting the integer to 1, while it is deactivated with 0. The default behavior (on/off) depends on the option and is listed.

- `# INPUTCONV shift` Shift of initial particle coordinates by specifying a vector or by positioning the `(0. 0. 0.)` coordinate in the center of the simulation domain (this would be equivalent to the half of the diagonal vector of the simulation domain.) Either supply vector `x y z` or write `center`. Note that the shift vector is given in scaled/fractional coordinates, e.g. the center is `(0.5 0.5 0.5)`.

- `# INPUTCONV trans` Transformation of initial particle coordinates by a matrix `A` (which needs to be supplied as: `<A_xx> <A_xy> <A_xz> <A_yx> <A_yy> <A_yz> <A_zx> <A_zy> <A_zz>`) Transformation is done after shifting. Note that the transformation is applied on scaled/fractional coordinates.

- `# INPUTCONV periodic 1|0` Enable/Disable periodic correction for coordinates of inserted particles. This feature is active by default.

- `# INPUTCONV temp #FLOAT` is specified all particles are assigned a Maxwell-Boltzmann distributed velocity, such that the total kinetic energy corresponds to the temperature value specified (in units of the chosen system. Using this option overwrites any user specified velocities. Note that the kinetic energy will most likely change over the course of the simulation, unless a thermostat is in use.

- `# INPUTCONV moment 0|1` results in the computation of the total momentum and a correction of all particle velocities, such that the new momentum is zero at the start of the simulation. Note that the use of a thermostat, while maintaining the kinetic energy might cause the momentum to change over the course of the simulation runtime. (On the momentum conservation of thermostats, see section 6.2) This feature is inactive by default.

- `# INPUTCONV angmoment 0|1` results in the computation of the total angular momentum and a correction of all particle velocities, such that the new angular momentum is zero at the start of the simulation. Note that the use of a thermostat, while maintaining the kinetic energy might (will most likely) cause the angular momentum to change over the course of the simulation runtime. This feature is inactive by default.

Note that # is always part of the command. Thus, comments may not be added to this file, as they would interfere with the parsing of the regular commands.

If both momentum correction and temperature distribution are set, then first the temperature distribution is set, then the moment is corrected, then the velocities are scaled again to match the temperature.

Example file: # ATOMDATA x=3 type # INPUTCONV shift center # INPUTCONV temp 0.57 # INPUTCONV moment 1 # INPUTCONV angmoment 1 0 0 0 C 0 1 0 H 0 0 1 H

13.4.2 Output conversion

13.5 $PROJECTNAME.parameters

This file contains the most extensive set of options, which may also interfere with each other. Therefore, this section only serves to explain the syntax of the file, while the comprehensive explanation of the option is moved to the following chapters.

The basic syntax scheme is the following:

```
$ITEM1: $PARAMETER1=#VALUE, $PARAMETER2=#values;
$ITEM2: $PARAMETER1=#VALUE;
$CATEGORY {
        $ITEM1: $PARAMETER1=#VALUE;
        $ITEM2: $PARAMETER1=#VALUE, $PARAMETER2=#values, \
        ↪    ... ;
        $SUBCATEGORY  {
                $ITEM1: $PARAMETER1=#VALUE;
                $ITEM2: $PARAMETER1=#VALUE, \
                ↪    $PARAMETER2=#values, ... ;
                $ITEM3: $PARAMETER1=#VALUE, \
                ↪    $PARAMETER2=#values;
                  };
             };
};
```

Modifications from this scheme occur for the propagator and timelines *FIXME: Timelines working?* and are described there in detail.

With two exceptions the parameters are grouped into the following categories:

- domain 13.5.1, 5
- optimization 13.5.3, 7
- dynamics 13.5.2, 6
- coulomb 13.5.4, 8

- parallelization 13.5.5, 9

- output 13.5.6, 10

There are two parameters, which are not associated with any of the above categories. The basic choice, whether the simulation type should be optimization or dynamics, determines, whether the parser takes options from the dynamics section or from the optimization section:

```
integration: type=$TYPE;
```

where $TYPE = {*dynamics, optimization*}.

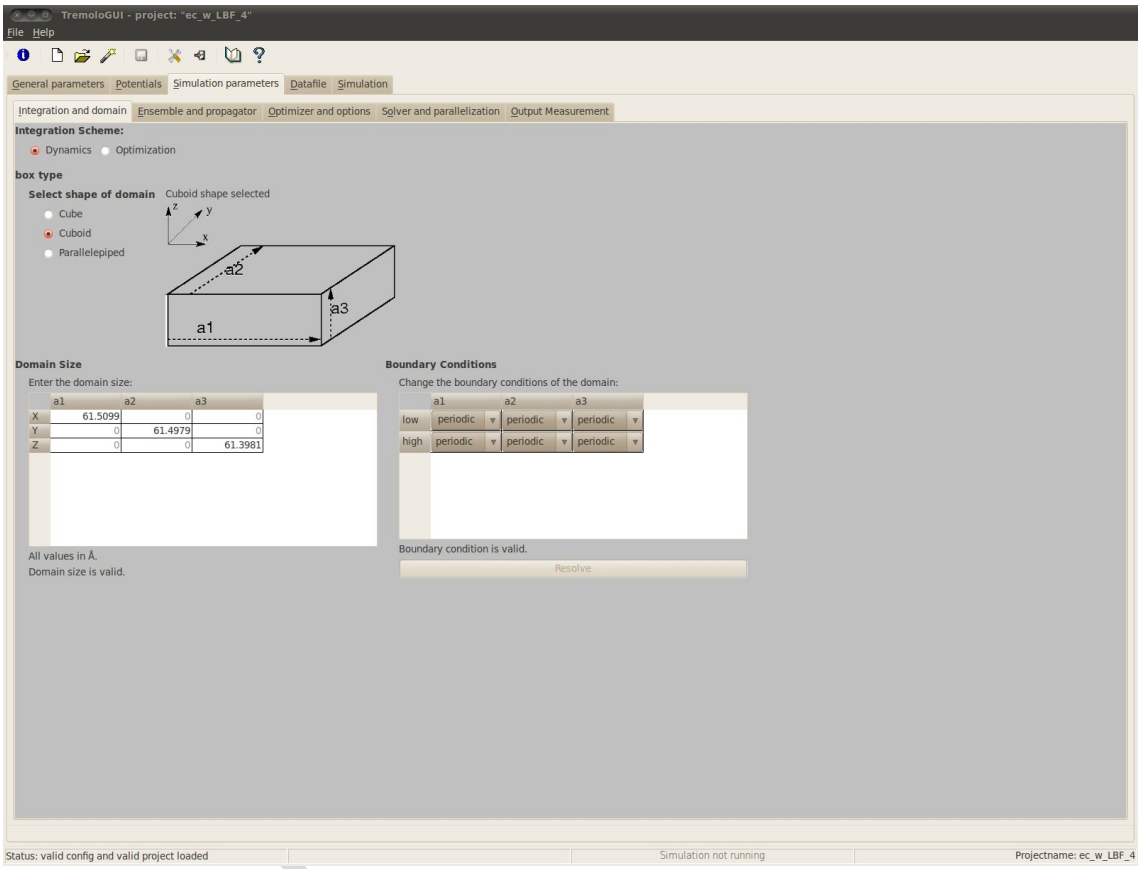
The other stand alone parameter is the cut"radius" of the linked cells, which is set by

```
lcs:     cellrcut=#FLOAT;
```

where #FLOAT is a positive floating point number. Also, the `cellrcut` must be smaller than the box dimensions or in case of parallel computations smaller than the box length divided by the number of processes in each direction.

Example: Your domain is 15x15x16. If you employ 3x3x3 processes, your `cellrcut` may not exceed 5. If you employ 3x3x4 processes your `cellrcut` may not exceed 4.

It is sensible to choose the `cellrcut` just as small, that cut off radii of pair potentials just fit into one cell [36].



13.5.1 domain

The category domain contains the items:

```
size:
type      = {cube, diag, matrix},
size      = #FLOAT,
length_x = #FLOAT, length_y = #FLOAT, length_z = #FLOAT,
xx = #FLOAT, xy = #FLOAT, xz = #FLOAT,
yx = #FLOAT, yy = #FLOAT, yz = #FLOAT,
yx = #FLOAT, zy = #FLOAT, zz = #FLOAT;
```

Depending on the type specified, you may leave out the size entries for the other categories, e.g.

- for `cube` you only need to specify `size`, which is a uniform edge length,
- for `diag` you need to set `length_x, length_y and length_z`,
- for matrix you need to set the edge vector in the matrix (There are vectors x,y,z with components x, y, z.).

The other specifiers may be omitted, however if you supply them the programm will ignore them safely.

Note that one can also give the box matrix by a `# BOX` line in the `.data` file. In particular, a `# BOX` line given in the `.data` file will overwrite values from the `.parameter` file.

```
# BOX xx xy xz yx yy yz zx zy zz
```

```
border:
bt_xlow  = {periodic, leaving, reflecting},
bt_xhigh = {periodic, leaving, reflecting},
bt_ylow  = {periodic, leaving, reflecting},
bt_yhigh = {periodic, leaving, reflecting},
bt_zlow  = {periodic, leaving, reflecting},
bt_zhigh = {periodic, leaving, reflecting};
```

The options `leaving` and `reflecting` may be mixed, whereas `periodic` must always match for the respective low and high borders.

13.5.2 dynamics

we have the items:

```
ensemble:          ensemble={NVE, NVT, NPT, NPE};
```

```
propagator,      $INTEGRATOR: delta_T=#FLOAT,   \
↪   endtime=#FLOAT,      timeinteps=#FLOAT,        \
↪   maxiteration=#INTEGER;
```

$INTEGRATOR = {`verlet, beeman2, beeman3, beeman4, beeman5, beeman2v, beeman3v`}

The parameters `timeinteps` and `maxiteration` are only relevant in the case of an velocity type algorithm, they are used for aborting the iteration procedure, either when the iterative error drops below `timeinteps` or the number of iteration cycles reaches `maxiteration`.

thermostat

The subcategory **thermostat**

```
berendsen:          state={on, off},      T_Interval=#FLOAT;
```

TODO: nose, dpd, timeline (working?)

barostat

The subcategory **barostat**

```
parinello: state={on, off},        f_mass=#FLOAT;
constraint: type={isotropic, standard, symmetric, none};
constraintmap: xx={0|1}, xy={0|1}, xz={0|1}, yx={0|1}, \
↪   yy={0|1}, yz={0|1}, zx={0|1}, zy={0|1}, zz={0|1};
constantpressure: state={on, off},        Pressure=#FLOAT;
Timeline: state={on, off},        [time, pressure, \
↪   interpolation=(0, 0, constant)];
stresstensor: [time, stress, interpolation, xx, xy, xz, \
↪   yy, yz,zz=(0, 0, constant, 0, 0, 0, 0, 0, 0)];
```

For theoretical background and detailed description of the options see section 6.3. The first option obviously switches the barostat on/off. The second option determines the virtual mass assigned to the box. In the constraint line one of the four constraint types must be chosen, which will work together with the following selection from the constraintmap. The selection of the constraintmap must always obey symmetry and in the isotropic case the "1" entries must be restricted to the diagonal. It is possible to

TODO: constraintmap, stresstensor; timeline(working?)

When a constraint type is chosen, this places certain restrictions on the constraintmap, however the user still needs to set it accordingly. (The constraintmap is not locked, since it allows the user to enforce restrictions to subdimensions.) The constraintmap must always observe `xy=yx` etc., even for the non symmetric restrictions.

- The `isotropic` constraint allows only changes on the main (diagonal) axis. Furthermore those changes are enforced to be isotropic, meaning the changes are identical in all allowed directions. (e.g. the user may set only `xx` and `yy` to 1, prohibiting changing of the box in the z direction.)

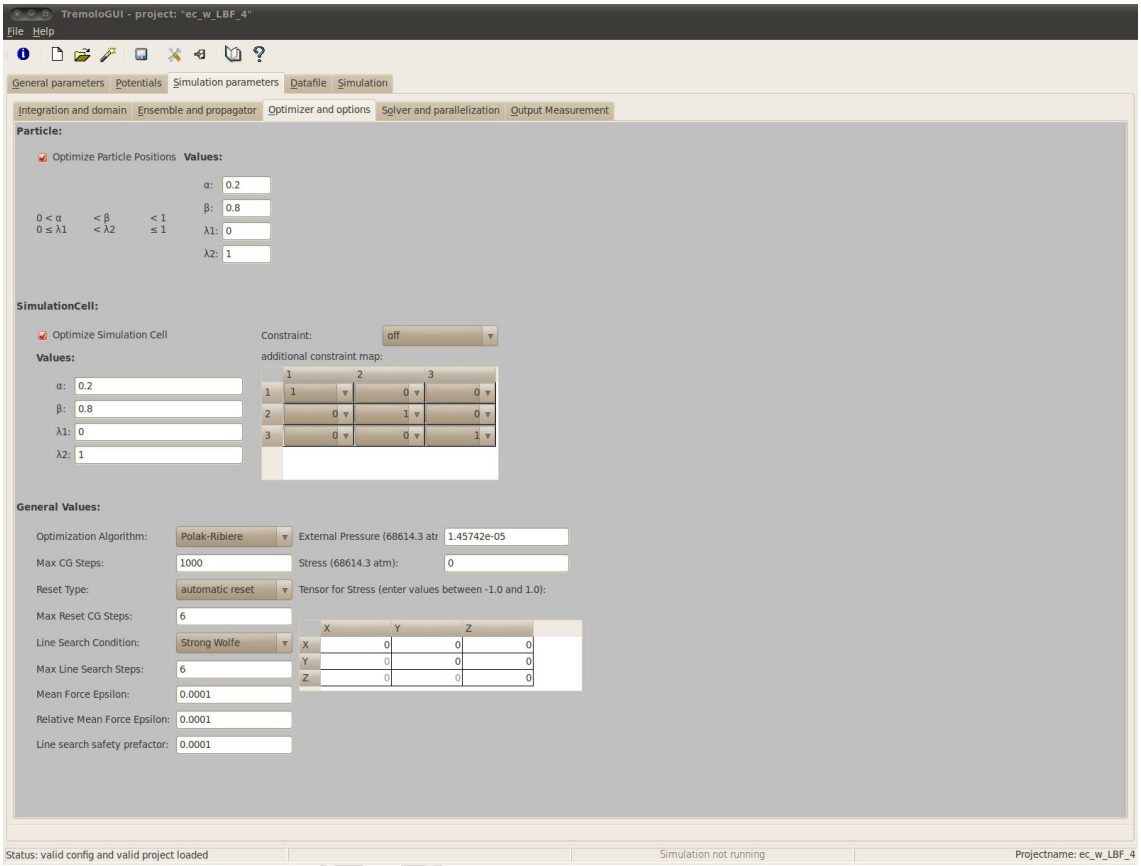
- `standard` the box deformation is restricted to an upper triangular matrix.

- `symmetric` constraints enforce symmetry in the box deformation.

- `none` No additional constraints are placed on the deformation tensor. Note that this is unphysical.

In the following lines we can specify the mean pressure applied to the system (both as a constant and as a timeline), as well as a stresstensor, which is externally applied, independent of the stress exerted by the particles in the system.

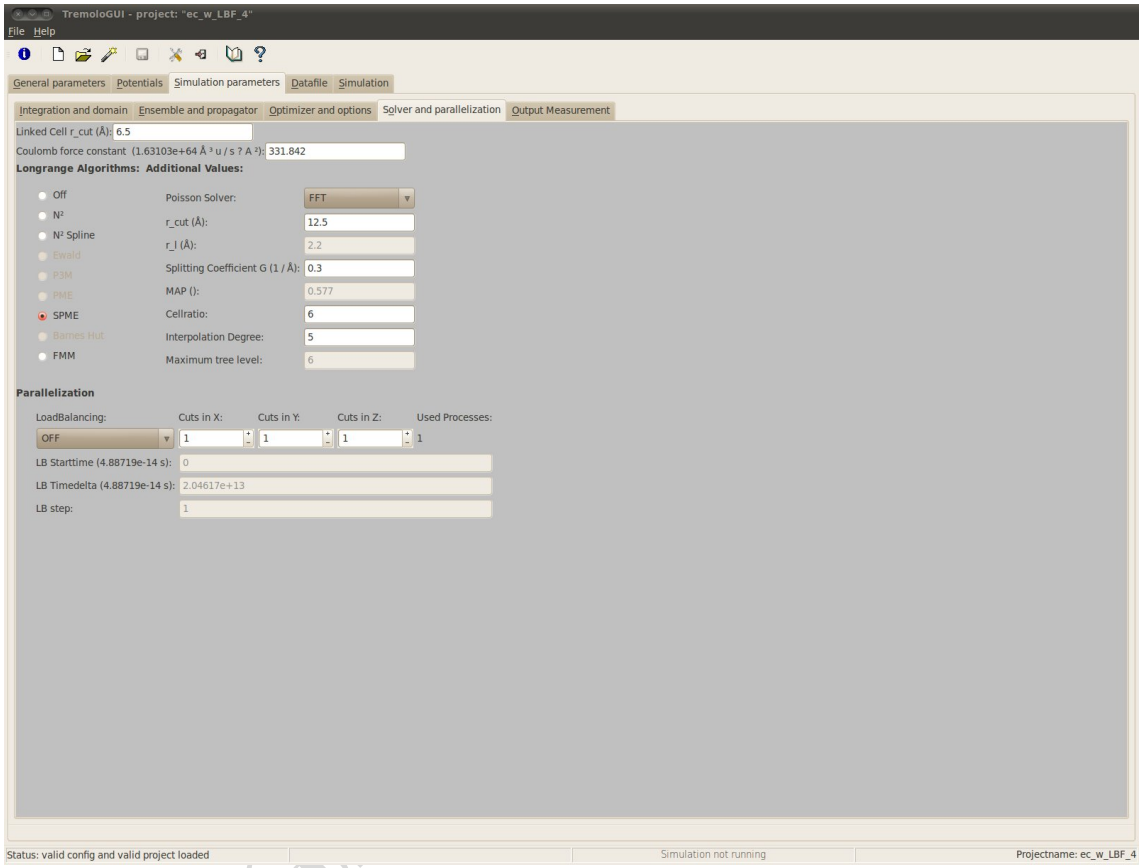
13.5.3 optimization




```
optimization{
  particle: state=on, alpha=0.2, beta=0.8, \
  ↪ lambda1=0, lambda2=1;
  simucell: state=off, alpha=0.2, beta=0.8, \
  ↪ lambda1=0, lambda2=1;
  simucell: constraint=isotropic, XX=1, XY=0, XZ=0, \
  ↪ YX=0, YY=1, YZ=0, ZX=0, ZY=0, ZZ=1;
  common: algorithm=fr, maxcg=102, RT=periodical, \
  ↪ maxresetcg=6, LS=wolfe, maxlinesearch=6, \
  ↪ mean_force_eps=1e-05, mean_force_eps_rel=1e-05, \
  ↪ prefactor=0.001, extpressure=1;
  stress: stress=0, XX=0, XY=0, XZ=0, YX=0, YY=0, \
  ↪ YZ=0, ZX=0, ZY=0, ZZ=0;
};
```

TODO: complete optimization

13.5.4 coulomb




```
coulomb {
permittivity:    epsilon0inv=1.9620959e-21;
n2:          state=off, r_cut=0.7352941, i_degree=5;
n2spline:    state=off, r_cut=0.7352941, r_l=0.6470588, \
↪    i_degree=5;
ewald:       state=off, r_cut=0.7352941, G=1.19e-10,    \
↪    i_degree=5,  cellratio=6;
p3m:         state=off, r_cut=0.7352941, G=1.19e-10,    \
↪    i_degree=5,  cellratio=6,  ps=fft;
pme:         state=off, r_cut=0.7352941, G=1.19e-10,    \
↪    i_degree=5,  cellratio=6,  ps=fft;
spme:        state=off, r_cut=0.7352941, G=1.19e-10,    \
↪    i_degree=5,  cellratio=6,  ps=fft;
barneshut:   state=off, r_cut=0.7352941, mtl=6,         \
↪    i_degree=5;
fmm:         state=off, r_cut=0.7352941, mtl=6,         \
↪    i_degree=5,  Map=0.577;
};
```

Currently `n2`, `n2spline` and `spme` are implemented.
Note that the ***`SPME`*** *method is not implemented sequentially and requires*

a Tremolo-X version with parallel capabilities. You can start the parallel version with one process though and use SPME nevertheless.

Permittivity

The coulomb-force-constant epsilon0inv is the only constant which is truly associated with units. (All others are converted to a reduced state.) As such it is dependent on the choice of the unit system. In this context it is necessary to note, that there are several different definitions of the calory, which need to be distinguished. Despite its name it is not the inverse of ϵ_0.

$$\texttt{epsilon0inv} = \frac{1}{4\pi\epsilon_0} \cdot \frac{e^2}{[length]},$$

where e is the respective charge unit and $[length]$ the unit of length used.

A few common examples, which all use the electron charge and angstrom but convert to different reference energies, are:

- Using angstrom and kcal_{th}/mol:
  $\frac{1}{4*\pi*\epsilon_0} * (e^2/angstrom) = 332.06371 \quad kilocal_{th}/molN_A$
- Using angstrom and kcal/mol:
  $\frac{1}{4*\pi*\epsilon_0} * (e^2/angstrom) = 331.84164 \quad kcal_{mol}$
- Using angstrom and eV:
  $\frac{1}{4*\pi*\epsilon_0} * (e^2/angstrom) = 14.399644 \quad eV$

r_cut, r_l

The r_cut parameter determines the separation of short and long range part of the potential. For the case of the splined N^2-Algorithm r_l defines the radius from which spline interpolation to the cut-off starts. For more information see section 8.

i_degree

This parameter determines the degree of the polynomials used for the field interpolation.

G

Localization parameter for the smooth screening functions. With larger G the localization is tighter. *TODO: Suggest to keep values between 0.24 and 0.35 1/angstrom?*

cellratio

Ratio of interpolation nodes per LC cell in one dimension. (Depending on the method this may be enlarged to the next power of two.)

mtl

Maximal tree level of the multipole method.
TODO: complete coulomb

13.5.5 parallelization

Parameters for domain decomposition can be set on the command line when calling the program. If this is done, those parameters supersede and replace those set in the `.parameters` file.

```
parallelization {
    proc:      n1=1,   n2=1,   n3=1,   all=1;
    loadbal:   wf=off, Lb\_t=0, Lb\_delta=4.608295e+11,  \
    ↪   Lb\_step=1;
};
```

Number of processors

The numbers `n1`, `n2` and `n3` are the number of cell bins per respective dimension. Their product must equal the total number of processes assigned with `all`.

Loadbalancing

TODO: complete parallelization

13.5.6 output

```
Outvis:   T_Start=0,   T_Delta=10,     Step_Delta=100;
Outm:     T_Start=0,   T_Delta=0.01,   Step_Delta=1;
Outmm:    T_Start=0,   T_Delta=10,     T_Deltam=9,      \
↪   Step_Delta=10,  Step_Deltam=9;
Outdata:  T_Start=0,   T_Delta=5,      Step_Delta=10;
energy:   measure=on,      meanmeasure=off;
analyze   {
          radial:       measure=off,  meanmeasure=off,  \
          ↪   vis=off,  r_cut=1.470588,   n_bin=50;
            radialdistribution {
                                radialdist:  \
                                ↪   particle_type1=CT,  \
                                ↪   particle_type2=CT;
                                };
```

```
bond:          measure=off,   meanmeasure=off,   \
↪    vis=off;
  bondDistances {
                 bondDistance:    \
                 ↪    particle_type1=CT,   \
                 ↪    particle_type2=CT,   \
                 ↪    distance=2.700000e+00;
                 };
velocity:      measure=off, meanmeasure=off,   \
↪    vis=off,   min=0,   max=0.006382352, n_bin=50;
msd:           msdusegroups=off;
msd:           particle_type=Argon, groupno=0, \
↪    starttime=100, resetinterval=1000, \
↪    measureflag=1, convection_correction=on/off;
local_stress: localstress=on/off;
};
```

Out{-vis,-m,-mm,-data}

See section 10.1 for details. Note that measure data is also written, if visual output is created, regardless of the state of the measure output interval. This interval is not touched, so if the intervals do not naturally coincide both are observed. Regardless of whether a measure event has been caused by creation of visual output or expiration of the measure interval, both are written to the same file.

The data stream consists of the current state of particles in terms of the input (`.data`-file). This information is written to `$PROJECTNAME.data.9999` in the case of serial computation or `$PROJECTNAME.data.9999.####` in the parallel case, where `####` are replaced by a four digit representation of the job number on which the particles where currently placed. (Scripts for merging this are described in *TODO: Describe scripts.*)

energy

Both option take `on` and `off` as arguments. They decide whether measure and/or mean measure are carried out.

radial

As a rule, in addition to switching the measurement of the radial distribution on, the user must specify for which pairs of particles the radial distribution shall be computed. However, the parameters set apply to all pairs of measurements. The measure/meanmeasure setting has the same functionality as for the other measurement types. The r_cut determines the distance from any central atom, up to which atoms are considered for the radial distribution function. The parameter n_bin sets the number of bins, into which the the distance is segmented and into which the atoms are counted.

The data is added to the `$PROJECTNAME.histogram` files. By reading the first line of the file, you find in which columns the data of a particluar measurement is stored. *TODO: What does the vis parameter do?* For the evaluation of the output see section 10.4 or the Tremolo_Tool_Documentation.

bond

TODO: complete output

velocity

TODO: complete output

msd

TODO: complete output

local_stress

When `localstress` is turned on, stress values will be computed for each individual particle. The values are then stored in the "beta" column in the .pdb files.

Read more about the output settings (specifically intervals) in the respective chapter above.

13.6 $PROJECTNAME.external

This file is optional, it is only needed when its functionality is desired. Then it also requires a $PROJECTNAME.exttypes file (see below).

In this file outer forces/constraints are specified. First a type is specified to which the following forces belong, then starting times are matched to forces/constraints. This also allows to switch outer influences on/off, while the simulation is running.

Type 0 must always be off.

Since this file is not required, there will be no warning when it is missing and it will not be looked for in the default path.

```
#EXTERNAL EXTTYPE number <extspec> ... END ...
# <extspec> : time ftype ...
# ftype     : FORCE | FREEZE | OFF ...
# FORCE Fscale Fx Fy Fz
# FREEZE
# OFF
```

```
# Example:
# no forces (this is the default and can be left out)
# Attention: do not change type 0!  Type 0 has to be OFF, \
↪   since it is the default!
#EXTERNAL EXTTYPE 0
#0.0      OFF

# pull to the left
EXTERNAL EXTTYPE 1
0.0      FORCE 3.5 -1. 0. 0.
10.0     OFF
END
# pull to the right
EXTERNAL EXTTYPE 2
2.0      FORCE 3.5  1. 0. 0.
10.0     OFF
END
# freeze (particles with this type will not move at all)
EXTERNAL EXTTYPE 3
0.0      FREEZE
END
```

TODO: Tether force not implemented yet.

In addition to applying forces to particletypes, it is possible to apply forces to regions of the domain. Currently implemented regions are tubes and spheres:

```
EXTERNAL EXTPOTENTIAL TUBE 500.0 0.5 0.5 0.0 1.5 0.0 0.0 3.5
```

- First three keywords indicate that a potential for a tube region (a zylinder) is defined.
- The force constant *TODO: units, formula?*
- 3 coordinates for the point of origin for the tube (zylinder), in other words the center of its base area. Note that those coordinates are transformed coordinates, they mus always be in the intervall [0,1].
- the radius of the tube. These are supplied in real units.
- 3 coordinates for the direction/length vector of the tube. This vector does not only define orientation, but also the length of the tube. It is also supplied in real coordinates.

```
EXTERNAL EXTPOTENTIAL SPHERIC 500 0.5 0.5 0.5 1.5
```

- First three keywords indicate that a potential for a spheric region is defined.

- The force constant *TODO: units, formula?*
- 3 coordinates for the center point of the sphere. Note that those coordinates are transformed coordinates, they mus always be in the intervall [0,1].
- the radius of the sphere. These are supplied in real units.

13.7 $PROJECTNAME.exttypes

This file is optional, it is only needed in combination with the use of a $PROJECTNAME.external file.

It contains a list of particle Id's matched with a corresponding number specifying the type of external force applied `ExtType`.

Since this file is not required, there will be no warning when it is missing and it will not be looked for in the default path.

```
# particles to which force specified as EXTTYPE 1 is applied
1 1
2 1

# particles to which force specified as EXTTYPE 2 is applied
11 2
12 2
```

Chapter 14

Tutorial

This tutorial will introduce you to the most common features of Tremolo-X by guiding you through a variety of example simulations. It is assumed that you read the chapter "First Steps" prior to starting with this tutorial.

All files mentioned here already exit in your Tremolo-X home directory in the folder `tutorial`. The leading number in the folder name corresponds to the section number in this chapter.

14.1 Optimizing an initial particle setup

Very often the setup of a simulation requires the assembly of a data file (with the particle positions) by hand or script. As a result the particle distribution is almost always non-optimal in the sense that the relative positions create local energy spikes which adversely affect the stability and equilibrium property of the simulation.

To counter these effects a simulation is usually preceded by an optimization phase, in which particle positions are slightly modified towards a (static) energy minimum.

We demonstrate the optimization procedure for an argon gas with reduced units.

We start by writing up the file `argon.tremolo`

```
global: defaultpath=".";
global: projectname="argon";
global: comment="Reduced argon example";
global: systemofunits=custom;
```

First we set the `defaultpath`. Even though we do not intend to use an external path to files in this example, this must always be set. In the second line we provide the `projectname`, all files for this simulation *must* carry this name (input) or respectively *will* carry this name (output). The `comment` can be freely used to add information about the simulation. In the last line we specify that we will use a custom system of units. This means that we

have to specify a few base magnitudes, from which any others are derived. Those magnitudes are specified by line pairs, the first fixing the physical unit to be used, whereas the second specifies the amount:

```
custom: lengthunit=angstrom;
custom: lengthscalingfactor=3.4;
custom: timeunit=ps;
custom: timescalingfactor=2.17;
custom: massunit=u;
custom: massscalingfactor=39.948;
custom: currentunit="e/s";
custom: currentscalingfactor=1;
custom: temperatureunit=K;
custom: temperaturescalingfactor=120;
```

In the above example, a length unit corresponds to 3.4Å, whereas mass is measured in 39.948 atomic mass units. (Thus a length of 2 in Tremolo-X units is 6.8Å, whereas a Tremolo-X mass of 0.5 corresponds to 19.974u.

Now we turn to the particles and their potentials, those are specified in the `argon.potentials` file. First we specify the particle types present in the simulation. In this case we use only one particle, which we name Argon, with chemical symbol `Ar`. The element name appears in some of the output files. Then we fix the `sigma, epsilon, sigma14 and epsilon14` values [42]. (Generally you should keep to the convention `sigma = sigma14` and `epsilon = epsilon14`, unless you have parameter sets for complex molecules which tell you otherwise.) Those are used in the computation of various potentials, namely the Lennard Jones potential which we will use here.

```
particles       {
        particle:       particle_type=Argon,    \
        ↪   element_name=Ar,        sigma=1,        \
        ↪   epsilon=1,      sigma14=1,      epsilon14=1, \
        ↪       mass=1, free=3, charge=0;
        };
```

Due to our custom choice of units, all those values are set to one. The same holds true for the particle mass. Degrees of freedom are three (this is the norm), the particles hold no charge. In the same file we enter the actual potentials to be used in between particles. Since only one type of particle is present there are not many mixtures to keep track of.

```
nonbonded_2body_potentials      {
        lennardjones:   particle_type1=Argon,   \
        ↪   particle_type2=Argon,   r_cut=12.0;
        ljspline:       particle_type1=Argon,   \
        ↪   particle_type2=Argon,   r_cut=12.0,     \
        ↪   r_l=10.0;
        };
```

We denote the pair of particle types which are affected by the potential. For the first Lennard Jones potential we specify a cut of radius r_{cut} of 12.0*TODO: Why?* (custom units), outside of which the interaction between two particles is set to 0. Since that introduces a discontinuity, we also second potential, which is a splined Lennard Jones Potential. It gets an additional parameter r_l, after which a spline is used to interpolate from the current value to 0 a r_{cut}. Note that both types of potentials are full LJ potentials in their own right. Using both would cause approximately twice the normal potential (and twice the normal forces) to be calculated between particles. Nevertheless, it does not hurt to supply any applicable potentials in this file, as they can be switched on and off at a different location, determining those which contribute to the particle interaction.

This location is the `argon.validates` file.

```
validate: particle_type=Argon, state=on;

validate: force_type=lennardjones, state=off;
validate: force_type=ljspline, state=on;
```

Here we validate the use of the particle type "Argon" and decide which of the two potentials shall remain in effect, while making sure that the other one is switched off.

The majority of parameter choices are made in the `argon.parameters` file. Here we will determine the domain, the number of optimization steps and make choices about the output created by the program. First we state that this simulation is to be an optimization, not a dynamic simulation.:

```
integration: type=optimization;
```

Next we determine the domain as cubic shaped with a side length of 81.05. The borders are to be periodic, so a particle leaving the cube over one side will reenter it via the other.

```
domain {
    size: type=cube, size=81.05;
    border: bt_xlow=periodic, bt_xhigh=periodic, \
    ↪   bt_ylow=periodic, bt_yhigh=periodic,
        bt_zlow=periodic, bt_zhigh=periodic;
};
lcs: cellrcut=12.0;
```

The choice of the cell size (`cellrcut`) is more algorithmic that geometric, but we have to ensure that `cellrcut < size`.
Remark: In the parallel case this condition changes to

$$\texttt{cellrcut} < \frac{\texttt{size}}{\#\ \texttt{of processors per dimension}}.$$

In the next block we set the options for parameterization. In this example we want to optimize particle positions only. Optimization is to be done by conjugate gradient (cg) method with 2001 steps (for algorithmic reasons the step number must be divisible by 3, otherwise it would be adjusted upwards).

```
optimization {
    particle: state=on, alpha=0.2, beta=0.8, lambda1=0, \
    ↪   lambda2=1;
    common: algorithm=cg, maxcg=2001, RT=periodical, \
    ↪   maxresetcg=6, LS=strongwolfe, maxlinesearch=6, \
    ↪   mean_force_eps=1e-6, mean_force_eps_rel=1e-10, \
    ↪   prefactor=1e-4;
};
```

The CG method is reset periodically to speed up convergence and furthermore we use strong Wolfe conditions on the cg line search. `mean_force_eps` and `mean_force_eps_rel` are the cutoff values for the absolute and relative mean force values respectively. The last parameter in line `prefactor`, is possibly the most important, as it directly controls the size of changes. Depending on the energy surface of the sample, a small value might be imperative to prevent the optimization from failing. On the other hand a small value prevents significant changes and slows the method down.

What is meant by visuals?

In the last block of this file we specify the type and intervals of output created. Visuals shall be created every 5 time units or 10 iteration steps, whereas the particle data shall be written every 500 time units or 10 iteration steps. Any other measured quantities are written every 0.5 time units or after 1 iteration step. In this example we measure only energy.

```
output {
    Outvis: T_Start=0, T_Delta=5.0, Step_Delta=10;
    Outdata: T_Start=0, T_Delta=500, Step_Delta=10;

    Outm: T_Start=0, T_Delta=0.5, Step_Delta=1;

    energy: measure=on;
};
```

The last item missing before we can start the simulation is the `argon.data` file supplying the initial particle positions. This file may not contain any free comments, all lines must match a particular format. Here we display only the first few lines of the file:

```
# ATOMDATA Id x=3 u=3 type
# INPUTCONV temp 2.7
1       80.10823        14.25174        47.74411        \
↪   0.0     0.0     0.0     Argon
2       70.29545        7.599451        58.29292        \
↪   0.0     0.0     0.0     Argon
```

```
3          66.10589          2.939586          69.22116          \
↪    0.0          0.0          0.0          Argon
4          77.18604          63.33052          73.03413          \
↪    0.0          0.0          0.0          Argon
5          76.02857          54.41654          24.62265          \
↪    0.0          0.0          0.0          Argon
6          16.96877          11.43903          21.40026          \
↪    0.0          0.0          0.0          Argon
7          53.09684          7.723396          47.48021          \
↪    0.0          0.0          0.0          Argon
8          77.50526          70.68508          49.97302          \
↪    0.0          0.0          0.0          Argon
⋮
```

The first line sets the layout of the particle list. It always begins with `# ATOMDATA` followed by the attributes set in the file. In this case this is the particle id, the particles position (3 columns), the particles velocity (3 columns) and the particle type. As you can see we already supply spatial coordinates, however we set all velocities to zero. Those will be set by the other `#` preceded line in the file, which performs some manipulation on the data provided. Such lines always begin with `# INPUTCONV`. In this instance we set the temperature of the sample. Using a Maxwell-Boltzmann distribution each particle is then assigned a random velocity.[1] The temperature is measured in the unit system we provided in the `argon.tremolo` file.

Now all pieces are in place to start the optimization on our particles. You can use the commands you learned in the "Quickstartguide" (chapter 3.1) to run the optimization.

The optimized particle positions are written to `argon.data.999`. The file looks very much like the original data file, though we find that the `INPUTCONV` line has been removed and two lines have been added at the top (Their use is explained in section 14.5). Furthermore the velocity columns now show non-zero entries - particle speeds corresponding to the specified temperature.

a '9' is missing in the file name

14.1.1 Exercises

- Increase the prefactor to $1.0e^{-2}$. What happens? The message tells you which particle causes the problem - have a look at it and the surrounding particles in the data file.

- Have a look at the potential energy curve, how does it behave?

[1] Note that if you plot these distributions, you see the shape of a Maxwell-Boltzmann distribution, but you seldomly find the parameters to be matching exactly. This is due to finite size effects, even very large simulations contain few particles compared to the real world, where you might have particle numbers by the mol.

14.2 Setting up a basic simulation

Now we are ready to start the actual simulation. Fortunately we did most of the work required setting up the optimization, so we now only have to amend very few lines in the `argon.parameters` file and make sure that we use the optimized data instead of the original. First we need to change the line specifying the integration type from "optimization" to "dynamics":

```
integration: type=dynamics;
```

The `cellrcut` and `domain` blocks remain untouched. We could remove the block with the optimization parameters, but since we changed the integration type to dynamic (removing optimization), we can also leave it in place without harm.

However we need to add a new block, setting the parameters for the dynamics:

```
dynamics {
    ensemble: ensemble=NVE;
    propagator, verlet: delta_T=0.5e-3, endtime=1000;
};
```

Here we set particle **n**umber, domain **v**olume and total **e**nergy to be constant (NVE ensemble). For the integration of the particle trajectories we choose a standard `verlet` algorithm with a time step of 0.005 custom time units. The total simulation time will be 1000 custom time units.

Do you mean 0.0005 or is this a mistake in the listing above?

At last we make a small addition to the **output** block; we would like to gain some insight into the velocity distribution of the particles:

```
 :
energy: measure=on;

    analyze {
        velocity: measure=on, meanmeasure=off, vis=off, \
        ↪    min=0.0, max=25.0, n_bin=50;
    };
};
```

After the `energy` line we add a sub-block for analysis (within the output block). We consider velocities between 0 and 25 (custom units) and use bins of 0.5 (custom units) each.

Now we have to make sure that we use the optimized data instead of the original. (As a general rule one should keep copies of the original at all times.) So, after having copied the original `argon.data` file somewhere safe, we rename the `argon.data.9999` file as `argon.data`. For this example we can still ignore the extra lines.

Which extra lines?

Now you can start the simulation as we did with the optimization before.

14.2.1 Exercises

- Compare the values of the `argon.etot, argon.ekin` and `argon.epot` files (plot them in the same graph). What do you notice?
- Take a look at the velocity distribution in the `argon.histogram` file.
- Try starting the simulation with the original data instead of the optimized.
- In order to smooth measurement curves and remove static, one often uses mean measurements over intervals. Switch mean measurements on for the energy measurements and compare the curves of the mean measurement with those of the regular measurement (they are written to separate files.)
- Have a look at the different energy curves. Then, in the `argon.data` file, change the temperature value to 3.0. (Use the `#INPUTCONV temp` line from the previous lesson.) The individual velocity values are then overwritten. How does this affect the different energies?

Is there something particular one have to look at?

14.3 Using the Berendsen thermostat

In this section we will introduce the first of two different thermostats. Given the previous preparations we require even less changes to the parameter file. All changes are within the `dynamics` block.

First we have to change the ensemble type:

```
dynamics {
    ensemble: ensemble=NVT;
    propagator, verlet: delta_T=0.5e-3, endtime=1000;
```

Instead of the total energy, we now hold the temperature constant. For this type of thermostat the propagator remains untouched. Then we open a new sub-block for the details of the thermostat:

```
    thermostat {
        berendsen: state=on, T_Interval=0.01;
        constanttargettemp: state=on, T_Temp=2.5;
    };
};
```

We declare that we use the `berendsen` thermostat and choose to enforce it every second time step by our choice of **T_Interval**. Furthermore we declare that we wish to hold the temperature constant and at which value. Note that we chose a value lower than the originally set temperature (regardless of whether you did the exercise or not.) Those are all the changes required, so go ahead and run the simulation.

which data file has to be used? The one with '9999' after running the last simulation?

In difference to the previous example, where the temperature varied around its original value, you will observe a very sharp drop from the original to the designated temperature. Afterwards all temperature values hit the value almost *exactly*. In fact, every second one is exactly at 2.5, whereas every other varies ever so slightly.

14.3.1 Exercises

- Compare the values of the `argon.etot, argon.ekin` and `argon.epot` files (plot them in the same graph). What changed compared to the previous lesson?

- Play with the time interval for the thermostat. Make it match the propagator timestep or make it a thousand times as long and observe the effects. In addition to the kinetic energy, observe the potential and total energy as well.

14.4 An alternative: The Nose-Hoover-thermostat

Shouldn't it be a é?

We also introduce a second type of thermostat. Three lines require a change in order to switch to the alternative, all in the `dynamics` block of the `argon.parameter` file.

```
dynamics {
    ensemble: ensemble=NVT;
    propagator, beeman2v: delta_T=0.5e-3, endtime=1000;
```

Here we have to make a change to the propagator, in order for the Nose-Hoover thermostat to work, a velocity integrator is required.

What is a velocity integrator?

```
    thermostat {
        berendsen: state=off, T_Interval=0.01;
        nosehoover: state=on, F_Mass=1.0;
        constanttargettemp: state=on, T_Temp=2.5;
    };
};
```

In the tutorials-file F_Mass is set to 5.0...

Instead of deleting the `berendsen` line we can also switch it off. In the added `nosehoover` line we do not have to specify an interval for the thermostat, but a virtual mass. This constant determines the strength of the coupling of the particles in the simulation with a virtual heat bath.

Start the simulation as before and observe the temperature behavior.

I am not sure what this is supposed to mean...

You will note that the temperature oscillates, first significantly reducing its amplitude. While the amplitude increases again after some time it does not gain the same value as before. Thus, when using the Nose-Hoover thermostat considerations with respect to equilibration are imperative.

I do not understand the last sentence.

14.4.1 Exercises

- Again, compare the values of the `argon.etot, argon.ekin` and `argon.epot` files (plot them in the same graph). What changed compared to the previous two lessons?

- Play around with different virtual masses (0.01 - 100.0) and different starting temperatures. You will note some different behaviors.

14.5 Optimizing the domain

Sometimes it is not possible to determine the optimal size of the domain prior to the simulation. In addition to the use of the barostat which we handle in the next lesson of this tutorial, we will now take a look at the initial optimization

We begin by modifying the the optimization of the sample prior to the actual simulation. In particular we allow that, in addition to the positions of the atoms, also the box may be scaled to minimize the potential energy.

Duplicate 'the'.

In the `argon.parameters` file from the first lesson we add two lines to the optimization block

```
optimization {
    particle: state=on, alpha=0.2, beta=0.8, lambda1=0, \
    ↪   lambda2=1;
    simucell: state=on, alpha=0.2, beta=0.8, lambda1=0, \
    ↪   lambda2=1;
    simucell: constraint=isotropic , XX=1, YY=1, ZZ=1, \
    ↪   XY=0, XZ=0, YX=0, YZ=0, ZX=0, ZY=0;
```

where we set the line search parameters for the cell optimization (generally we can use the same parameters as for the particle positions) and may also choose some constraints on which entries of the box matrix are allowed to vary.

Finally, we make one addition to the `common` parameter set by adding an external pressure value:

```
    common: algorithm=cg, maxcg=2001, RT=periodical , \
    ↪   maxresetcg=6, LS=strongwolfe , maxlinesearch=6, \
    ↪   mean_force_eps=1e-6, mean_force_eps_rel=1e-10, \
    ↪   prefactor=1e-4, extpressure=0.0024455185;
};
```

After running the optimization, take a look at the file `argon.data.9999`. In addition to the new coordinates of the particles you find a time stamp and a box matrix entry. This is the domain shape after the optimization has finished. To make use of these values, you need to transfer them to the parameter file, as described in the next lesson.

14.5.1 Exercises

- Change the `extpressure` value. Check and compare the new box values[2]. (Changes by order of magnitude are advised for clearly visible results.)
- In the constraint matrix, change one of the main axis entries (`XX, YY, ZZ`) to 0. Check and compare the new box values.
- Change one of the secondary axis entries to 1. Check and compare the new box values. Now rename the constraint to `constraint=standard` and check again. Repeat the process for `constraint=symmetric`.

14.6 Introducing barostats

In some cases it is desired to run simulations not only with isothermic, but also isobaric conditions. For this to be possible we allow the volume of the box to be variable and set a barostat similarly to the thermostat.

Again we work in the `argon.parameter` file and enter the thermostat after (or in place of) the thermostat:

```
dynamics {
    ensemble: ensemble=NPE;
    propagator, beeman2v: delta_T=0.5e-3, endtime=1000;

    thermostat {
        berendsen: state=off, T_Interval=0.01;
        nosehoover: state=off, F_Mass=0.01;
        constanttargettemp: state=off, T_Temp=2.5;
    };

    barostat {
        parinello: state=on, f_mass=1;
        constantpressure: state=on, Pressure=0.0024455185;

        constraint: type=isotropic;
        constraintmap: xx=1, yy=1, zz=1, xy=0, xz=0, yx=0, \
        ↪   yz=0, zx=0, zy=0;
    };
};
```

In the first line we switch the barostat on and specify a virtual mass. In the second line we can choose whether the pressure aimed at shall be constant and if so, at what value. Note that as usually this value is in reduced units. Furthermore, we make our choice of constraints on the allowed changes to the box. First we specify a specific type of constraint, and afterwards we can

[2] If you run tremolo with increased verbosity (e.g. `-v` the new box matrix - among other things - will be written to the standard output)

apply additional modification by specifying entries in the box-matrix which may be affected.

Finally we have to deal with the particle data. If we start with a file where we did not use the box-optimization there is nothing else to worry about, we can start the simulation immediately. But lets take the `argon.data.9999` file from the previous lesson. (Preferably the one which was optimized with the originally supplied parameters.) Should you attempt to start it right away, you will receive an error message. This is due to the fact that the box-matrix in the data file and the one in the parameter file are different. So go ahead and change the domain in the appropriate line. (Note that the value shown here may be different from your value. If you wish to use a different result (e.g. the one won from `symmetric` constraints) you need to change the domain type as well. For the appropriate syntax see 13.5.1.

```
domain {
    size: type=cube, size=7.935811e+01;
    border: bt_xlow=periodic, bt_xhigh=periodic, \
    ↪   bt_ylow=periodic, bt_yhigh=periodic,
        bt_zlow=periodic, bt_zhigh=periodic;
```

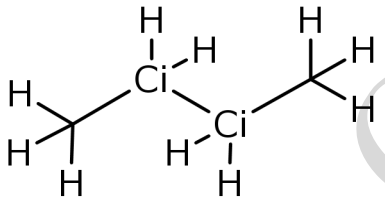
14.6.1 Exercises

- Change the `extpressure` value. Check and compare the new box values. (changes by order of magnitude are advised for clearly visible results)
- In the `constraintmap`, change one of the main axis entries (`XX, YY, ZZ`) to 0. Check and compare the new box values.
- In the `constraintmap`, change one of the secondary axis entries. Check and compare the new box values.
- Change the `f_mass` value to 1000. Check and compare the of the box values.
- Go back to the previous lesson and use one of the other constraints types (such as standard). Now transfer this `argon.data.9999` file and insert its domain shape in the `argon.parameter` file. For the appropriate syntax see 13.5.1.

14.7 Bonded potentials and measuring bonds

Hitherto only the nonbonded Lennard Jones interaction has been covered in this Tutorial. In most cases the connectivity of atoms is known and it is desired that it stays in its initial configuration. In these cases the indices of the neighbor atoms are set in the appropriate column in the data file and bonded

potentials are specified in the `.potentials` file. The bond type covered here, named `bond` is a harmonic potential – computationally inexpensive and with unbreakable bonds. As per definition bonded-type bonds can't be broken, however with harmonic potentials there will be a restoring force proportional to the deflection from the minimal energy distance r_0. While unphysical for large displacements, this behaviour can also be desired if simulations are carried out under very high kinetic energies to speed processes up, because the general structure of the molecules will persist.



We will set up a butane example and measure the bond distances in the CH3- and CH2-groups. There are three atom types: Methyl-carbon (`C` in CH3), methylene-carbon (`Ci` in CH2, i stands for "inner") and Hydrogen (`H`). The butane data file is build accordingly and the field `neighbors=4` is added. The ensemble provided with the tutorial files consists of $5 \times 5 \times 5$ butane molecules with a density of $2.71\,\mathrm{kg\,m^{-3}}$. The unit system used is kcalpermole.

Listing 14.1: Header of the .data file

```
# ATOMDATA Id type x=3 u=3 neighbors=4
```

When setting up the `butane.potentials` file, begin with Lennard Jones interaction, which should additionally work in between molecules. Tremolo-X handles Lennard Jones in bonded molecules in a way, that the potential is *not* calculated among direct neighbors. However for large molecules, like proteins, intramolecular interaction should be considered: These are controlled via the settings `sigma14` and `epsilon14`, as they take effect from the fourth bond on, hence the naming.

As for the bonded potentials we use `bonds`, `angles` and `torsions`. The angle potential, like the bonds, is harmonic (linear restoring force for angle displacement from an optimal value) and the torsional potential is expressed as cosine series expansion. All parameters are taken from the AMBER94 force field. For the literal definition of the potential terms see the potentials-section in this manual (11.4.1).

Listing 14.2: Excerpt from the .potentials file

```
bonds {
  bond: particle_type1=C, particle_type2=Ci, \
  ↪    bond_type=harmonic, k_b=310, r_0=1.526;
```

```
  [...]
};

angles {
  angle: particle_type1=C, particle_type2=Ci, \
  ↪    particle_type3=Ci, angle_type=harmonic, k_th=40, \
  ↪    theta_0=109.5;
  [...]

torsions {
  torsion: particle_type1=H, particle_type2=C, \
  ↪    particle_type3=Ci, particle_type4=H, \
  ↪    torsion_type=cosine, k_1=1.4, delta_1=0, n_1=3, \
  ↪    mult=1;
  [...]

};
```

The simulation `$PROJECTNAME.parameters` are a simple `NVE` ensemble with verlet propagator and initial temperature of 0 °C. In the `analyze` section bond distance measurement is set up. It doesn't matter whether the measured pair is bonded in means of the `neighbors` field: Every pair with specified types undershooting the specified threshold (in this particular case 180 pm for C–C and 140 pm for H–C) are considered bonded and their Ids written to the `$PROJECTNAME.info.bonds` (vis) file. The mean value of bond lengths of any pair of types requested in the `$PROJECTNAME.parameters` file is written to the `$PROJECTNAME.generalmeas` file.

Listing 14.3: Excerpt from the .parameters file

```
    analyze {
        bond: measure=on, meanmeasure=off, vis=off;
        bondDistances {
            bondDistance: particle_type1=Ci, \
            ↪    particle_type2=Ci, distance=1.8;
            bondDistance: particle_type1=C, \
            ↪    particle_type2=Ci, distance=1.8;
            bondDistance: particle_type1=Ci, \
            ↪    particle_type2=H, distance=1.4;
            bondDistance: particle_type1=C, \
            ↪    particle_type2=H, distance=1.4;
        };
```

As we can see the mean values oscillate around a constant, so there is no time dependant development. We assume the ensemble is equilibrated after 20 time units and compare the mean of the mean values for any available pair and get following results:

Bond	Distance/pm	r_0/Å
C–Ci	152.7(1)	1.526
Ci–Ci	152.7(2)	1.526
C–H	109.1(1)	1.09
Ci–H	109.08(9)	1.09

There are no differences in the bond length within statistical error and the values match the r_0-Parameter because the same parameters were used for both, inner and outer, carbon atom types. The nearest neighbors of the atoms (spatial configuration) are not taken into account thus leading to an identical bond distance.

14.7.1 Exercises

- Increase the temperature or lower the bond strength and observe the magnitude of the oscillation.
- Alter the equilibrium distance r_0.

14.8 Tersoff potential and stress

After we successfully introduced basic bonded and non-bonded force fields, this chapter shows how to use a bond order potential to determine Young's Modulus of a single graphene sheet. Instead of defining fixed individual neighbors, the potential function will determine the spatial configuration of surrounding carbon atoms by itself. This way the `graphene.data` file looks rather trivial, the force field parameters for `graphene.potentials` are taken directly from [94]:

```
tersoff {
    tersoffparticle: particle_type=C0, A=32115.7, \
    ↪   B=7990.67, lambda=3.4879, mu=2.2119, \
    ↪   beta=1.5724e-07, n=0.72751, c=38049, d=4.3484, \
    ↪   h=-0.57058, R=1.95, S=2.1;
};
```

Just like in 14.6 an `NPT`-ensemble is used, but this time additionally to the external pressure we also support a custom stress tensor, which stretches the domain in `xx`-direction with linearly increasing strength, starting from 0 in the beginning up to $1e5\ \frac{[F][V]}{[A]}$[3] (`kcalpermole` units). At this time note that the stress value is not given in units of pressure, as one would expect, but contains a surplus V-Term. The volume can usually be derived from the domain dimensions, but particularly in this tutorial we have to *decide*

[3] **F**orce, **V**olume and **A**rea. For a detailed explanation of the volume term see chapter 10.8

how much volume a single graphene sheet has. Also the box vectors need to be changed individually, so we will choose `standard` constraints (`isotropic` constraints would be infeasible due to the coupling of the constraints).

```
barostat {
    parinello: state=on, f_mass=500;
    constraint: type=standard;
    constraintmap: xx=1, xy=0, xz=0, yx=0, yy=1, yz=0, \
    ↪   zx=0, zy=0, zz=0;
    constantpressure: state=on, Pressure=1.45742e-05;

    stresstensor: [time, stress, interpolation, xx, xy, \
    ↪   xz, yy, yz, zz =
        (0, 0, linear, 1, 0, 0, 0, 0, 0),
        (200, 1e5, linear, 1, 0, 0, 0, 0, 0)];
};
```

Stress and strain are measured automatically if `outm` is set. However if you are interested in the stress distribution along individual particles you need to use the `local_stress`-feature:

```
analyze {
    local_stress: localstress=on;
};
```

If activated, the beta column in the visual .pdb output contains per particle stress, which can be visualized with an external tool like VMD-Viewer (not covered by this tutorial). Due to limitations of the PDB-format the numerical value is clipped at $99.99\frac{[F][V]}{[A]}$.

After the simulation has been finished, the output is analyzed by plotting a stress-strain diagram. The strain is defined as the length change relative to the initial box `xx`-length L_0: $\epsilon(L) = \frac{L-L_0}{L0}$. The argument will be the 43^{rd} column in the file `graphene.mbox`, which is the domain length in `xx` direction. The stress is read from column 31 in the same file and divided by the volume of the graphene sheet, which is the product of `xx` length (column 43), `yy` length (column 44) and *height*, which has to be chosen by the user. We set $height = 3.7\mathring{A}$, the inter layer distance in graphite. Note that since we pull, the stress value from the file is negative. With this information we can now use gnuplot to create the diagram and fit a slope against the initial section, representing the elastic zone where Hook's Law applies. This slope is the Young's Modulus.

```
gnuplot> L0 = 48.19683
gnuplot> epsilon(L) = (L-L0)/L0
gnuplot> plot 'graphene.mbox' using \
↪   (epsilon($43)):(-$31/($43*$44*3.7))

gnuplot> f(x) = E*x
gnuplot> fit [0:0.04] f(x) 'graphene.mbox' using \
↪   (epsilon($43)):(-$31/($43*$44*3.7)) via E
```

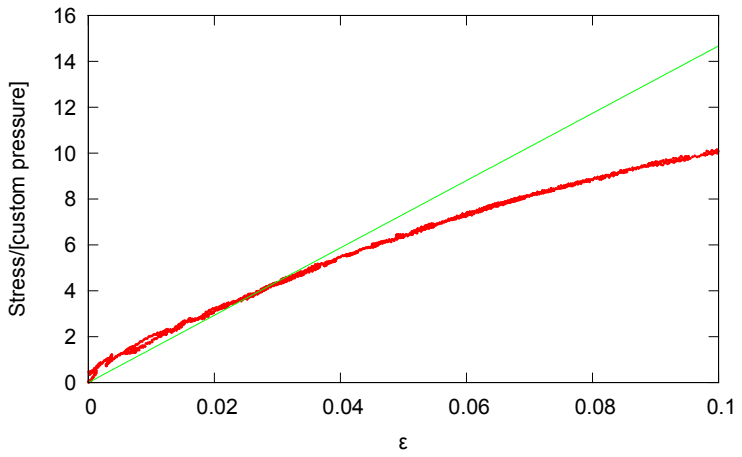


Figure 14.1: Stress-strain diagram of monolayer graphene (red) with green slope fit.

We obtain $E = 146.7(5)\,\frac{[F]}{[A]}$, in SI-units $1019(3)\,\mathrm{GPa}$, which sufficiently reproduces the measurement of 1 TPa in [50].

14.8.1 Exercises

- Change the pull direction to `yy`. Are there differences?
- Alter the force progression:
  - Increase the simulation length by factor ten.
  - Add a $100\,[t]$ relaxation time at the beginning ($\sigma = 0$).
  - Increase the maximum stress at the end of the simulation by factor five.

  Create a new stain-stress plot – is there a difference in the fit quality ($\chi^2_{\mathrm{d.o.f.}}$ and parameter error as supplied by gnuplot)?

14.9 Long ranged potentials 1 - Halley's Comet with N^2

This tutorial covers how to set up simulation to use long ranged potentials like gravity or the coulomb potential. In either case the potential's distance characteristic is $\propto\ 1/x$ and any kind of cutoff introduces significant errors on the forces and with them error in macroscopical observables.

The simpelest way to deal with these kind of potentials is using an ordinary pair potential. As stated earlier the potential should not be cut off but has to to fit the linked cell structure of the domain. With only a few

particles it is adequate to run the whole simulation in a single cell, the performance drops quadratically with the particle number. In this example we simulate our solar system and calculate when Halley's Comet runs through its perihelion point next time (as of time of writing, year 2013).

To obtain handy numbers we set up following unit system in the `.tremolo` file:

Quantity	Unit	Equals
Length	1.496×10^{11} m	ua
Time	86 400 s	d
Mass	1×10^{23} kg	$\approx m_{\mathrm{Earth}}/100$

For the `.data` file we use positions (relative to the Barycenter) and velocities obtained from JPL HORIZONS for sixty astronomical objects of our solar system including the most massive ones. The data represents the state of the solar system on December 19th, 2012.

Tremolo-X doesn't support "gravity" literally, but since the potential is identic to the coulomb potential disregarding the force constant we set up a coulomb simulation with adapted `epsilon0inv`. It can easily be calculated with Gnu Units:

```
> units -t G "(m*1.496e11)**3 / (kg*1e23 * (s*86400)**2)"
1.4881216e-11
```

Note that the value needs to be multiplied by -1 so equally named "charges" attract each other.

We use an NVE ensemble with `verlet` propagator, 0.05 d `timestep` and 100 a `endtime`, "temperature" and "pressure" can't be applied on the experimental conditions. Also it would not be accurate to wrap around the gravitational forces at the borders of the solar system as it is preferred when homogenous systems are observed, so we choose the domain to three times larger than the solar system, place it in the center (done with a `INPUTCONV SHIFT` directive in the data file) and set `leaving` boundary conditions (particles passing the border are removed):

```
lcs: cellrcut = 80.0;

domain {
    size: type=cube, size=240;
        border: bt_xlow=leaving, bt_xhigh=leaving, \
        ↪   bt_ylow=leaving,
    bt_yhigh=leaving, bt_zlow=leaving, bt_zhigh=leaving;
};
```

A single linked cell will be large enough to contain the whole ensemble. This is not practical for large scale molecular dynamics because it can't be paralellized, however unless a more sophisticated method like SPME (covered

in the next section) is used, it is the only way to obtain accurate results using long ranged potentials.

The `coulomb` section is set up thusly:

```
coulomb {
    permittivity: epsilon0inv=-1.4881216e-11;
    n2spline: state=on, r_cut=80, r_l=70, i_degree=5;
};
```

The `n2spline` solver calculates the force for every pair and cuts off beyond 70 ua with a spline taper (like `ljspline`). The cutoff is chosen large enough to contain the whole solar system so the forces will be exact. If `n2spline` is used in molecular dynamics with small cutoff the spline interpolation guarantees conservation of energy but the forces will be off their exact result due to the characterstic of the potential.

The bond distance measurement covered earlier is used to measure the distance between Halley's Comet/Earth and the Barycenter:

```
output {
[...]
    analyze {
        bond: measure=on, meanmeasure=off, vis=off;
        bondDistances {
            bondDistance: particle_type1=Barycenter, \
            ↪   particle_type2=Earth, distance=80;
            bondDistance: particle_type1=Barycenter, \
            ↪   particle_type2=Halley, distance=80;
        };
    };
};
```

Finally we set up the objects in the `.potentials` file. To calculate the gravity potential with the coulomb solver we need to set the "charge" of the particle to its mass. The Barycenter is included as pseudo particle with mass 1 and no charge – this way it won't be affected by any force and stay central.

After running the simulation we extract the time to the minimum of $|\vec{r}_{\text{Barycenter}} - \vec{r}_{\text{Halley}}|$ by plotting the first and fourth column of the `.generalmeas` file. It's 17750 time units (days), so the next perihelion is 25th July 2061, slightly deviant to the value computed with HORIZONS: 28th July. This relative difference of 0.02% is caused by the simplification of the solar system to sixty objects and the resulting difference in the local density. The minima of the Earth–Barycenter distance are 365.24 d apart which matches the definition of the tropical year.

14.9.1 Excercises

- Remove any particle but Sun, Jupiter, Barycenter and Halley and compare the relative error of the perihelion.

- Only remove Jupiter from the original setup and compare the relative error, observe planetary trajectories and the length of an earth year.

14.10 Long ranged potentials 2 - Sodium chloride with SPME

This part covers a typical usage scenario of coulomb forces in molecular dynamics with more than just a few particles. To maintain a good performance with a large N the potential is seperated into a short ranged part, which is calculated in a linked cell fashion as before, and a long ranged part, which is calculated by Ewald summation in fourier space, to take into account farther particles. In comparison to the Fast Multipole Method, which abstracts groups of far particles into a single one, this way is espacially suitable for periodic systems, like an ionic crystal: In this example we are going to simulate solid NaCl and measure its radial distribution functions. The system of units used is `kcalpermole`.

In the `.potentials`-file we set up the short ranged interactions using the Tosi Fumi[12] Potential, which has shown to produce accurate results with this kind of system. The starting configuration in the data file is an NaCl-structure with small random offset for each atom at 20 °C.

We set up an `NPT`-ensemble in the `.parameters` file with 1000 hPa pressure maintained by the Parrinello-Barostat with isotropic constraint (the crystal is cubic) and Nose-Hoover-Thermostat for fixed temperature. In the `coulomb`-section we specify the parameters for the `spme`-method and a force constant using the vacuum permittiviy:

```
coulomb {
    permittivity: epsilon0inv=332;
    spme: state=on, r_cut=9.0, G=0.32, i_degree=5, \
    ↪    cellratio=4;
};
```

Up to `r_cut`, which is the short ranged part of the potential, the force is evaluated locally (restricted to neighbored cells) and pair wise, as if `n2spline` was used. From there it is approximated by bell curves with splitting coefficient `G`, which is inverse to the standard deviation σ, and applied on a mesh, which is created by splitting the linked cell `cellratio` times (rounded upward to the next power of two). `G` should be kept at 0.24 Å^{-1} to 0.35 Å^{-1}.

In the `analyze`-section we set up the measurement of the radial distributions of all atom types:

```
output {
[...]
    analyze {
        radial: measure=on, meanmeasure=off, vis=off, \
        ↪    r_cut=9.0, n_bin=50;
```

```
        radialdistribution {
            radialdist: particle_type1=Na, \
            ↪   particle_type2=Na;
            radialdist: particle_type1=Na, \
            ↪   particle_type2=Cl;
            radialdist: particle_type1=Cl, \
            ↪   particle_type2=Cl;
        };
    };
};
```

The value of `r_cut` has to be within the `lcs: cellrcut` limits, just like the `r_cut` of the potentials.

Since we now use the SPME method, we have to use a parellel version of Tremolo-X, since the SPME method is not implemented sequentially. Nevertheless you can unse the SPME method and start the parallel version with a single process.

If you do not know how to run the parallel version of Tremolo-X, please check chapter 3.2 and 4.

14.10.1 Excercises

- Compare the radial distribution histograms from first (ideal NaCl structure) and last timestep.
- Decrease/Increase the temperature and observe the differences in the radial distribution.

14.11 Melting point of Sodium Chloride

In the previous Tutorial the basics for simulating an ionic crystal using the Coulomb- and Tosi-Fumi-Potential have been covered while this one shows how to determine the melting point of NaCl, which is a common application of molecular dynamics. From the different methods described in [102] we are going to use the *Voids method*: A series of NaCl lattices with increasing defect concentration (removed atoms) is simulated using an NPT-ensemble with temperature timeline. Starting with an ideal lattice, which is a hypothetical, defect free state at 0 K, the temperature at which the lattice breaks ("melting point") is significantly overestimated because the activation energy for this transition is very high. By removing atoms from the crystal it becomes labile and the lattice breaks easily if the temperature is high enough, lowering the activation energy to start the melting process. If the defect concentration is too high the crystal rearranges back into a more stable configuration, increasing the observed melting point. This process leads to a reduced volume which has to be compensated for, using a barostat. Therefore if the measured melting point is plotted against defect concentration one

can see that the measured melting point decreases quickly with increasing defect concentration at first and then oscillates around the actual melting point, which can be obtained by calculating the mean value.

A tricky aspect is how to observe the melting point: Liquid and solid state can be discriminated using the MSD-measurement, potential energy, density, radial distribution or bond length, usually indicated by a rapid slope change which can be seen if these measurements are plotted. Preliminary studies have shown that observing the bond lenght is the easiest to interpret while very accurate option in this particular example: As the crystal heats up the bond lenght increases linearily until the crystal breaks and the bond distances relaxes rapidly. The obsverved melting point T_m is the temperature at maximum bond lenght.

The simulation setup is similar to the previous tutorial apart from the thermostat settings and more measurement options in the `nacl.parameters` file:

```
thermostat {
    timeline: state=on, [time, temperature, interpolation =
    (0, 0.5822, linear), # 20°C
    (100, 0.5822, linear),
    (1000, 2.9255, linear)]; # 1200°C

    nosehoover: state=on, F_Mass=800;
};
```

TODO: Buchstabendreher beim Gradzeichen im PDF/Text fixen (Encoding-problem mit lstlisting).

With these settings the temperature is held constant at 20 °C for 100 time units and then lineary increased to 1200 °C at the end of the simulation (1000 time units).

```
bond: measure=on;
bondDistances {
    bondDistance: particle_type1=Na, particle_type2=Cl, \
    ↪    distance=4.0;
};
```

Every Na–Cl-pair with a distance less than or equal to 4 Å is considered bonded and contributes to the mean distance written to `nacl.generalmeas`.

When carrying out a series of simulations it is handy to make use of the `defaultpath`-option in `nacl.tremolo`. The simulation is organized into a root directory which contains any file but `.data` and `.tremolo` and subdirectories containing only these, whith individual `nacl.data` containing an increasingly more defective crystal. The individual `nacl.tremolo` files look like this

```
global: defaultpath="../nacl";
global: projectname="nacl";
global: comment="NaCl";
```

```
global: systemofunits=kcalpermole;
```

with the `defaultpath` set to the parent directory and nacl as basename, so tremolo looks for `nacl.potentials` and so forth. The subdirectories are named after the relative count of cells containing a (single) pair defect. A single cell contains 8 atoms, so the resulting defect concentration is *dirname*/4.

After the simulations are done the times at which the bond lenght peaks are read from `nacl.generalmeas` and the temperature at this time is looked up in `nacl.ekin`, fourth column.

Plotting T_m against defect concentration results in following graph:

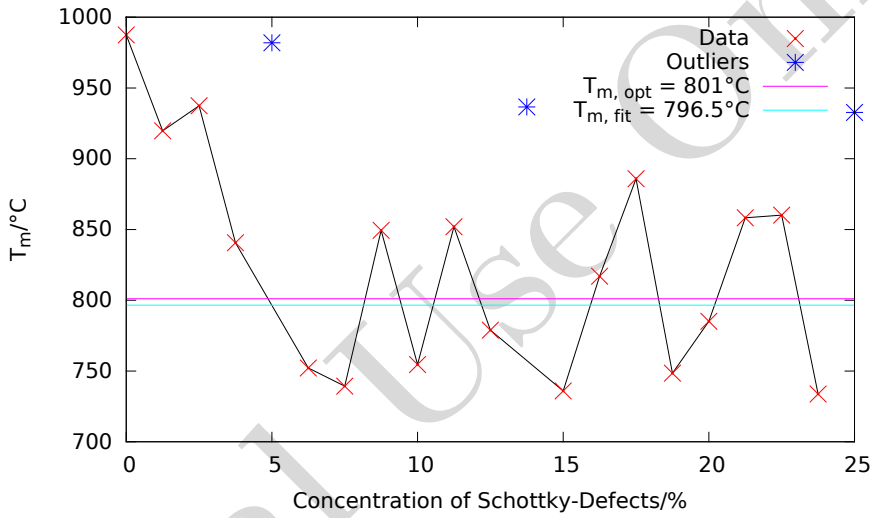


Figure 14.2: Melting points at different defect concentrations. $T_{\mathrm{m,\ opt}}$ is the melting point according to literature, $T_{\mathrm{m,\ fit}}$ is the mean value of the oscillating region. The black dot-connecting line and outliers are for visualization purposes only.

The oscillating region starts after 7.5 %, the resulting mean value differs by 4.5 K from the exact value of 801 °C, which is a relative error as low as 0.6 %.

14.12 The EAM potential - Observing phase transition in Metall

The "embedded atom method" (EAM) is a standard potential used in the analysis of metalls and alloys. They are used quite successfully in the investigation of fractures, surface reactions martensite-austenite transitions and phasechanges in condensed matter, nano particles and thin films.

In this tutorial lesson we will demonstrate the use of the EAM potential and how a phase transition can be analyzed with Tremolo-X. We will heat a

Fe-Ni nanoparticle from 100 K to 800 K and observe it changing its lattice structure from bcc to fcc/hcp.

In order to use EAM potentials, the user must have a file with EAM parameters in the format "eam/fs" (a generalized EAM type by Finnis-Sinclaire) or in the "eam/alloy" format, which is slightly modified from DYNAMO setfl file formats. For a detailled description of the EAM formats supported, please check the appropriate section 11.3.7

The unit system of the eam parameters file determines the units which need to be used throughout the simulation. As a result in this example SI units will be used. The respective entries for iron and nickel particle values in the potential file are consequently:

```
particles       {
        particle:       particle_type=Fe,        \
        ↪   element_name=Fe,
                        sigma=1.0,               epsilon=0.0,
                        sigma14=1.0,             \
                        ↪   epsilon14=0.0,
                        mass= 9.2732785e-26,
                        free=3,
                        charge=0;
        particle:       particle_type=Ni,        \
        ↪   element_name=Ni,
                        sigma=1.0,               epsilon=0.0,
                        sigma14=1.0,             \
                        ↪   epsilon14=0.0,
                        mass=9.7462664e-26,
                        free=3,
                        charge=0;
        };
```

and subsequently we specify the format ("alloy" format) and the filename in the following way:

```
eam     {
          setfl:      file="Fe-Ni-MeyerEntel-1995.eam.alloy";
        };
```

In the parameter file we need to specify the following:

```
integration: type=dynamics;
############ Section Domain:
domain {
       size:    type=cube, size=60.0e-10;
       border:  bt_xlow=periodic,
                bt_xhigh=periodic,
                bt_ylow=periodic,
                bt_yhigh=periodic,
                bt_zlow=periodic,
                bt_zhigh=periodic;
```

```
                };
########## Section Ensemble and Propagator:
dynamics {
        ensemble:       ensemble=NVT;
        propagator,     verlet: delta_T=1.0e-15,        \
        ↪   endtime=8.0e-11,        \
        ↪   timeinteps=1e-07,       maxiteration=100;
        thermostat      {
                berendsen:  state=on,
                            T_Interval=1.0e-15;
                timeline:   state=on,
                        [time, temperature, interpolation=
                        (0,         1.3806503e-21, linear), # \
                        ↪   100K / 7.2429638e+22 K
                        (1e-14,     1.3806503e-21, linear),
                        (7.56e-11, 1.1045202e-20, linear), # \
                        ↪   800K
                        (8.0e-11,  1.1045202e-20, linear)];
                          };
         };
########## Section Solver and Parallelization:
lcs:    cellrcut=5.6001e-10;
########## Section Output Measurement:
output  {
        Outvis: T_Start=0, T_Delta=5.0e-13, Step_Delta=100;
        Outm:   T_Start=0, T_Delta=1.0e-15, Step_Delta=100;
        Outmm:  T_Start=0, T_Delta=5.0e-14, \
        ↪   T_Deltam=4.0e-14, Step_Delta=20, \
        ↪   Step_Deltam=10;
        Outdata: T_Start=0, T_Delta=10.0e-13, Step_Delta=50;
        energy: measure=on,     meanmeasure=off;
        analyze {
          radial: measure=on, meanmeasure=off, vis=on, \
          ↪   r_cut=5.6001e-10, n_bin=56;
                radialdistribution      {
                    radialdist: particle_type1=Ni, \
                    ↪   particle_type2=Ni;
                    radialdist: particle_type1=Ni, \
                    ↪   particle_type2=Fe;
                    radialdist: particle_type1=Fe, \
                    ↪   particle_type2=Fe;
                     };
                   };
        };
```

Note that while we use SI units, the temperature has a scaling prefactor, which needs to be accounted for when setting the thermostat. While we use periodic boundaries, those will not be relevant to our simulation, as we set the simulation domain to a cube of 60Å, whereas the nanoparticle has a

radius of 20 Å. Thus no interaction with or across the domain boundaries is present.

We now can run the simulation.After having run the simulation, we can take a look at the potential energy curve and notice that its slope changes around 4.64e-11s into the simulation. It is at this time that the transformation takes place.

We will now analyze the radial distribution of the sample at the beginning of the simulation, during the melting phase and after the transition phase. For this we will use the tool "CalcRadialHist" delivered with Tremolo-X.

By calling

```
CalcRadialHisto eam.histogram 5e-15 15.3e-15 0.0 5.6e-10 \
↪    3.0e-10 1 56 545 2196 216000 3.0e-10 > RadialHist
```

we average the radial distribution over the time interval from 5 to 15 fs and write the results to the file `RadialHist`. The same can be done for the time intervals 4e-11s to 4.3e-11s and 5e-11s to 5.3-11s. When those three distributions are compared, we observe, that around 4e-11s the original configuration has been severly melted, but that around 5e-11s the atoms have rearanged themselves, which can be seen from the clearly shifted peaks in the radial distribution function.

Bibliography

[1] http://www.hyper.com/.

[2] http://www.pharmacy.umaryland.edu/~alex/research.html.

[3] *PDB guide*. http://rutgers.rcsb.org/pdb/docs/format/pdbguide2.2.

[4] *The RCSB Protein Data Bank*. http://rutgers.rcsb.org/pdb.

[5] GJ Ackland, DJ Bacon, AF Calder, and T Harry. Computer simulation of point defect properties in dilute fe—cu alloy using a many-body interatomic potential. *Philosophical Magazine A*, 75(3):713–732, 1997.

[6] GJ Ackland, MI Mendelev, DJ Srolovitz, S Han, and AV Barashev. Development of an interatomic potential for phosphorus impurities in α-iron. *Journal of Physics: Condensed Matter*, 16(27):S2629, 2004.

[7] GJ Ackland, G Tichy, V Vitek, and MW Finnis. Simple n-body potentials for the noble metals and nickel. *Philosophical Magazine A*, 56(6):735–756, 1987.

[8] Graeme J Ackland. Theoretical study of titanium surfaces and defects with a new many-body potential. *Philosophical Magazine A*, 66(6):917–932, 1992.

[9] Karsten Albe, Kai Nordlund, and Robert S Averback. Modeling the metal-semiconductor interaction: Analytical bond-order potential for platinum-carbon. *Physical Review B*, 65(19):195124, 2002.

[10] Karsten Albe, Kai Nordlund, Janne Nord, and Antti Kuronen. Modeling of compound semiconductors: Analytical bond-order potential for Ga, As, and GaAs. *Physical Review B*, 66(3):035205, 2002.

[11] James E Angelo, Neville R Moody, and Michael I Baskes. Trapping of hydrogen to lattice defects in nickel. *Modelling and Simulation in Materials Science and Engineering*, 3(3):289, 1995.

[12] Jamshed Anwar, Daan Frenkel, and Massimo G. Noro. Calculation of the melting point of nacl by molecular simulation. *The Journal of Chemical Physics*, 118(2):728–735, 2003.

[13] Steven M Arnold and Terry T Wong. *Models, Databases and Simulation Tools Needed for Realization of Integrated Computational Mat. Eng.(ICME 2010)*. ASM International, 2011.

[14] M Backman, N Juslin, and K Nordlund. Bond order potential for gold. *The European Physical Journal B*, 85(9):1–5, 2012.

[15] Anatoly B Belonoshko, R Ahuja, and Börje Johansson. Quasi–ab initio molecular dynamic study of fe melting. *Physical Review Letters*, 84(16):3638, 2000.

[16] C Björkas, KOE Henriksson, M Probst, and K Nordlund. A be–w interatomic potential. *Journal of Physics: Condensed Matter*, 22(35):352206, 2010.

[17] C Björkas, N Juslin, H Timko, K Vörtler, K Nordlund, K Henriksson, and P Erhart. Interatomic potentials for the Be-C-H system. *Journal of Physics: Condensed Matter*, 21(44):445002, 2009.

[18] Giovanni Bonny, RC Pasianot, and Lorenzo Malerba. Fe–ni many-body potential for metallurgical applications. *Modelling and Simulation in Materials Science and Engineering*, 17(2):025010, 2009.

[19] Giovanni Bonny, Roberto C Pasianot, Nicolas Castin, and Lorenzo Malerba. Ternary fe–cu–ni many-body potential to model reactor pressure vessel steels: First validation by simulated thermal annealing. *Philosophical Magazine*, 89(34-36):3531–3546, 2009.

[20] B. Brooks, R. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus. CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Caclulations. *J. Comp. Chem.*, 4(2):187–217, 1983.

[21] brooks@scripps.edu. *CHARMM c27b1 Documentation*. `http://www.scripps.edu/brooks/c27docs/Charmm27.Html`.

[22] T Çağın, G Dereli, M Uludoğan, and M Tomak. Thermal and mechanical properties of some fcc transition metals. *Physical Review B*, 59(5):3468, 1999.

[23] Murray S Daw and MI Baskes. Semiempirical, quantum mechanical calculation of hydrogen embrittlement in metals. *Physical Review Letters*, 50(17):1285–1288, 1983.

[24] Murray S Daw and Mi I Baskes. Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Physical Review B*, 29(12):6443, 1984.

[25] F de Brito Mota, JF Justo, and A Fazzio. Hydrogen role on the properties of amorphous silicon nitride. *Journal of applied physics*, 86(4):1843–1847, 1999.

[26] R Devanathan, T Diaz de la Rubia, and WJ Weber. Displacement threshold energies in β-sic. *Journal of nuclear materials*, 253(1):47–52, 1998.

[27] P. Erhart and K. Albe. Analytical potential for atomistic simulations of silicon, carbon, and silicon carbide. *Physical Review B*, 71(3):035211, 2005.

[28] Paul Erhart, Niklas Juslin, Oliver Goy, Kai Nordlund, Ralf Müller, and Karsten Albe. Analytic bond-order potential for atomistic simulations of zinc oxide. *Journal of Physics: Condensed Matter*, 18(29):6585, 2006.

[29] Michael R Fellinger, Hyoungki Park, and John W Wilkins. Force-matched embedded-atom method potential for niobium. *Physical Review B*, 81(14):144119, 2010.

[30] Kristen A Fichthorn, Yogesh Tiwary, Thomas Hammerschmidt, Peter Kratzer, and Matthias Scheffler. Analytic many-body potential for gaas (001) homoepitaxy: Bulk and surface properties. *Physical Review B*, 83(19):195328, 2011.

[31] MW Finnis and JE Sinclair. A simple empirical n-body potential for transition metals. *Philosophical Magazine A*, 50(1):45–55, 1984.

[32] Andrea Fortini, Mikhail I Mendelev, Sergey Buldyrev, and David Srolovitz. Asperity contacts at the nanoscale: Comparison of ru and au. *Journal of Applied Physics*, 104(7):074320–074320, 2008.

[33] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation*. Academic Press, 2. edition, 2002.

[34] Marcus Gastreich, Julian D Gale, and Christel M Marian. Charged-particle potential for boron nitrides, silicon nitrides, and borosilazane ceramics: Derivation of parameters and probing of capabilities. *Physical Review B*, 68(9):094110, 2003.

[35] M. Griebel and J. Hamaekers. Molecular dynamics simulations of the mechanical properties of polyethylene-carbon nanotube composites. In M. Rieth and W. Schommers, editors, *Handbook of Theoretical and*

Computational Nanotechnology, volume 9, chapter 8, pages 409–454. American Scientific Publishers, 2006. Also as INS Preprint No. 0502.

[36] M. Griebel, S. Knapek, and G. Zumbusch. *Numerical Simulation in Molecular Dynamics*. Springer, Berlin, Heidelberg, 2007.

[37] Gregory Grochola, Salvy P Russo, and Ian K Snook. On fitting a gold embedded atom method potential using the force matching method. *The Journal of chemical physics*, 123:204719, 2005.

[38] R. Haberlandt, S. Fritzsche, G. Peinel, and K. Heinziger. *Molekular-dynamik*. vieweg, 1995.

[39] Thomas Hammerschmidt, P Kratzer, and M Scheffler. Analytic many-body potential for inas/gaas surfaces and nanostructures: Formation energy of inas quantum dots. *Physical Review B*, 77(23):235303, 2008.

[40] Krister OE Henriksson and K Nordlund. Simulations of cementite: An analytical potential for the fe-c system. *Physical Review B*, 79(14):144107, 2009.

[41] Derek J Hepburn and Graeme J Ackland. Metallic-covalent interatomic potential for carbon in iron. *Physical Review B*, 78(16):165115, 2008.

[42] J. Honeycutt and H. Andersen. Molecular dynamics study of melting and freezing of small lennard-jones clusters. *J. Phys. Chem.*, 91:4950–4963, 1987.

[43] Huai Yu Hou, Rong Shan Wang, Jing Tao Wang, Xiang Bing Liu, Guang Chen, and Ping Huang. An analytic bond-order potential for the fe–cu system. *Modelling and Simulation in Materials Science and Engineering*, 20(4):045016, 2012.

[44] Jin-Wu Jiang, Harold S Park, and Timon Rabczuk. Molecular dynamics simulations of single-layer molybdenum disulphide (mos2): Stillinger-weber parametrization, mechanical properties, and thermal conductivity. *Journal of Applied Physics*, 114(6):064307, 2013.

[45] N Juslin, P Erhart, P Traskelin, J Nord, Krister OE Henriksson, K Nordlund, E Salonen, and K Albe. Analytical interatomic potential for modeling nonequilibrium processes in the w–c–h system. *Journal of applied physics*, 98(12):123520–123520, 2005.

[46] Sefa Kazanc and Cengiz Tatar. Investigation of the effect of pressure on some physical parameters and thermoelastic phase transformation of NiAl alloy. *International Journal of Solids and Structures*, 45(11):3282–3289, 2008.

[47] Yoshitaka Kimura, Yue Qi, Tahir Cagin, and WA Goddard. The quantum sutton–chen many-body potential for properties of fcc metals. *Phys. Rev., to be submitted*, 1998.

[48] J Kioseoglou, Ph Komninou, and Th Karakostas. Interatomic potential calculations of iii (al, in)–n planar defects with a iii-species environment approach. *physica status solidi (b)*, 245(6):1118–1124, 2008.

[49] A Landa, P Wynblatt, DJ Siegel, JB Adams, ON Mryasov, and X-Y Liu. Development of glue-type potentials for the al–pb system: phase diagram calculation. *Acta materialia*, 48(8):1753–1761, 2000.

[50] C. Lee, X. Wei, J. W. Kysar, and J. Hone. Measurement of the Elastic Properties and Intrinsic Strength of Monolayer Graphene. *Science*, 321:385–, July 2008.

[51] Xiao-Chun Li, Xiaolin Shu, Yi-Nan Liu, Fei Gao, and Guang-Hong Lu. Modified analytical interatomic potential for a w–h system with defects. *Journal of Nuclear Materials*, 408(1):12–17, 2011.

[52] Youhong Li, Donald J Siegel, James B Adams, and Xiang-Yang Liu. Embedded-atom-method tantalum potential developed by the force-matching method. *Physical Review B*, 67(12):125101, 2003.

[53] L Lindsay and DA Broido. Optimized tersoff and brenner empirical potential parameters for lattice dynamics and phonon thermal transport in carbon nanotubes and graphene. *Physical Review B*, 81(20):205441, 2010.

[54] Xiang-Yang Liu, Furio Ercolessi, and James B Adams. Aluminium interatomic potential from density functional theory calculations with improved stacking fault energy. *Modelling and Simulation in Materials Science and Engineering*, 12(4):665, 2004.

[55] Xiang-Yang Liu, PP Ohotnicky, JB Adams, C Lane Rohrer, and RW Hyland Jr. Anisotropic surface segregation in al mg alloys. *Surface science*, 373(2):357–370, 1997.

[56] Christel M Marian and Marcus Gastreich. A systematic theoretical study of molecular si/n, b/n, and si/b/n (h) compounds and parameterisation of a force-field for molecules and solids. *Journal of Molecular Structure: THEOCHEM*, 506(1):107–129, 2000.

[57] Katsuyuki Matsunaga, Craig Fisher, and Hideaki Matsubara. Tersoff potential parameters for simulating cubic boron carbonitrides. *JAPANESE JOURNAL OF APPLIED PHYSICS PART 2 LETTERS*, 39(1A/B):L48–L51, 2000.

[58] Katsuyuki Matsunaga and Yuji Iwamoto. Molecular dynamics study of atomic structure and diffusion behavior in amorphous silicon nitride containing boron. *Journal of the American Ceramic Society*, 84(10):2213–2219, 2001.

[59] B. Mehlig, D. W. Heermann, and B. M. Forrest. Hybrid monte carlo method for condensed-matter systems. *Phys. Rev. B*, 45:679–685, Jan 1992.

[60] M. I. Mendelev, D. J. Srolovitz, G. J. Ackland, and S. Han. Effect of fe segregation on the migration of a non-symmetric sigma 5 tilt grain boundary in al. *J. Mater. Res.*, 20(1):208–218, January 2005.

[61] MI Mendelev, M Asta, MJ Rahman, and JJ Hoyt. Development of interatomic potentials appropriate for simulation of solid–liquid interface properties in al–mg alloys. *Philosophical Magazine*, 89(34-36):3269–3285, 2009.

[62] MI Mendelev, S Han, DJ Srolovitz, GJ Ackland, DY Sun, and M Asta. Development of new interatomic potentials appropriate for crystalline and liquid iron. *Philosophical magazine*, 83(35):3977–3994, 2003.

[63] MI Mendelev, MJ Kramer, CA Becker, and M Asta. Analysis of semi-empirical interatomic potentials appropriate for simulation of crystalline and liquid al and cu. *Philosophical Magazine*, 88(12):1723–1750, 2008.

[64] MI Mendelev, MJ Kramer, SG Hao, KM Ho, and CZ Wang. Development of interatomic potentials appropriate for simulation of liquid and glass properties of nizr2 alloy. *Philosophical Magazine*, 92(35):4454–4469, 2012.

[65] MI Mendelev, MJ Kramer, RT Ott, DJ Sordelet, D Yagodin, and P Popel. Development of suitable interatomic potentials for simulation of liquid and amorphous cu–zr alloys. *Philosophical Magazine*, 89(11):967–987, 2009.

[66] MI Mendelev, DJ Sordelet, and MJ Kramer. Using atomistic computer simulations to analyze x-ray diffraction data from metallic glasses. *Journal of Applied Physics*, 102(4):043501–043501, 2007.

[67] Mikhail I Mendelev, Seungwu Han, Won-joon Son, Graeme J Ackland, and David J Srolovitz. Simulation of the interaction between fe impurities and point defects in v. *Physical Review B*, 76(21):214105, 2007.

[68] R Meyer and P Entel. Molecular dynamics study of iron-nickel alloys. *Le Journal de Physique IV*, 5(C2):C2–123, 1995.

[69] Y Mishin, D Farkas, MJ Mehl, and DA Papaconstantopoulos. Interatomic potentials for monoatomic metals from experimental data and ab initio calculations. *Physical Review B*, 59(5):3393, 1999.

[70] Y Mishin, MJ Mehl, and DA Papaconstantopoulos. Embedded-atom potential for b2-nial. *Physical Review B*, 65(22):224114, 2002.

[71] Yu Mishin, MJ Mehl, DA Papaconstantopoulos, AF Voter, and JD Kress. Structural stability and lattice defects in copper: Ab initio, tight-binding, and embedded-atom calculations. *Physical Review B*, 63(22):224106, 2001.

[72] Yuri Mishin. Atomistic modeling of the γ and $\gamma<$ sup$>'<$/sup$>$-phases of the ni–al system. *Acta materialia*, 52(6):1451–1467, 2004.

[73] Won Ha Moon, Myung Sik Son, and Ho Jung Hwang. Molecular-dynamics simulation of structural properties of cubic boron nitride. *Physica B: Condensed Matter*, 336(3):329–334, 2003.

[74] M. Müller, P. Erhart, and K. Albe. Analytic bond-order potential for bcc and fcc iron—comparison with established embedded-atom method potentials. *Journal of Physics: Condensed Matter*, 19(32):326220, 2007.

[75] M. Müller, P. Erhart K., and Albe. Thermodynamics of l1_ {0} ordering in FePt nanoparticles studied by monte carlo simulations based on an analytic bond-order potential. *Physical Review B*, 76(15):155412, 2007.

[76] S. Munetoh, T. Motooka, K. Moriguchi, and A. Shintani. Interatomic potential for si–o systems using tersoff parameterization. *Computational materials science*, 39(2):334–339, 2007.

[77] Christian Neuen. Ein multiskalenansatz zur poisson-nernst-planck gleichung - a multiscale approach to the poisson-nernst-planck equation. Diplomarbeit Universität Bonn, 2010.

[78] J. Nord, K. Albe, P. Erhart, and K. Nordlund. Modelling of compound semiconductors: analytical bond-order potential for gallium, nitrogen and gallium nitride. *Journal of Physics: Condensed Matter*, 15(32):5649, 2003.

[79] K Nordlund, J Nord, J Frantz, and J Keinonen. Strain-induced kirkendall mixing at semiconductor interfaces. *Computational materials science*, 18(3):283–294, 2000.

[80] Onyekwelu U Okeke and JE Lowther. Molecular dynamics of binary metal nitrides and ternary oxynitrides. *Physica B: Condensed Matter*, 404(20):3577–3581, 2009.

[81] SM Peng, Li Yang, XG Long, HH Shen, Qing-Qiang Sun, XT Zu, and Fei Gao. Bond-order potential for erbium-hydride system. *The Journal of Physical Chemistry C*, 115(50):25097–25104, 2011.

[82] GP Purja Pun and Y Mishin. Embedded-atom potential for hcp and fcc cobalt. *Physical Review B*, 86(13):134116, 2012.

[83] GP Purja Pun and Y Mishin. Development of an interatomic potential for the ni-al system. *Philosophical Magazine*, 89(34-36):3245–3267, 2009.

[84] H Rafii-Tabar and AP Sulton. Long-range finnis-sinclair potentials for fcc metallic alloys. *Philosophical Magazine Letters*, 63(4):217–224, 1991.

[85] Daniel Schopf, Peter Brommer, Benjamin Frigan, and Hans-Rainer Trebin. Embedded atom method potentials for al-pd-mn phases. *Physical Review B*, 85(5):054201, 2012.

[86] DE Smirnova, A Yu Kuksin, SV Starikov, VV Stegailov, Z Insepov, J Rest, and AM Yacout. A ternary eam interatomic potential for u–mo alloys with xenon. *Modelling and Simulation in Materials Science and Engineering*, 21(3):35011–35034, 2013.

[87] DE Smirnova, SV Starikov, and VV Stegailov. Interatomic potential for uranium in a wide range of pressures and temperatures. *Journal of Physics: Condensed Matter*, 24(1):015702, 2012.

[88] F. H. Stillinger and T. A. Weber. Computer simulation of local order in condensed phases of silicon. *Phys. Rev. B*, 31(8):5262–5271, 1985.

[89] Jess B Sturgeon and Brian B Laird. Adjusting the melting point of a model system via gibbs-duhem integration: Application to a model of aluminum. *Physical Review B*, 62(22):14720, 2000.

[90] Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314(1–2):141 – 151, 1999.

[91] DY Sun, MI Mendelev, CA Becker, K Kudin, Tomorr Haxhimali, M Asta, JJ Hoyt, A Karma, and DJ Srolovitz. Crystal-melt interfacial free energies in hcp metals: A molecular dynamics study of mg. *Physical Review B*, 73(2):024116, 2006.

[92] AP Sutton and J Chen. Long-range finnis–sinclair potentials. *Philosophical Magazine Letters*, 61(3):139–146, 1990.

[93] J. Tersoff. Empirical interatomic potential for silicon with improved elastic properties. *Phys. Rev. B*, 38(14):9902–9905, 1988.

[94] J. Tersoff. New empirical approach for structure and energy of covalent systems. *Phys. Rev. B*, 37(12):6991–7000, 1988.

[95] J. Tersoff. New empirical approach for the structure and energy of covalent systems. *Physical Review B*, 37(12):6991, 1988.

[96] J. Tersoff. Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Phys. Rev. B*, 39(8):5566–5568, 1989.

[97] J. Tersoff. Erratum: Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Physical Review B*, 41(5):3248–3248, 1990.

[98] J. Tersoff. Chemical order in amorphous silicon carbide. *Physical Review B*, 49(23):16349, 1994.

[99] PL Williams, Y Mishin, and JC Hamilton. An embedded-atom potential for the cu–ag system. *Modelling and Simulation in Materials Science and Engineering*, 14(5):817, 2006.

[100] JM Winey, Alison Kubota, and YM Gupta. A thermodynamic approach to determine accurate potentials for molecular dynamics simulations: thermoelastic response of aluminum. *Modelling and Simulation in Materials Science and Engineering*, 17(5):055004, 2009.

[101] Henry H Wu and Dallas R Trinkle. Cu/ag eam potential optimized for heteroepitaxial diffusion from ab initio data. *Computational Materials Science*, 47(2):577–583, 2009.

[102] Yong Zhang and Edward J. Maginn. A comparison of methods for melting point calculation using molecular dynamics simulations. *The Journal of Chemical Physics*, 136(14):144116, 2012.

[103] XW Zhou, RA Johnson, and HNG Wadley. Misfit-energy-increasing dislocations in vapor-deposited cofe/nife multilayers. *Physical Review B*, 69(14):144113, 2004.

[104] XW Zhou, JA Zimmerman, BM Wong, and JJ Hoyt. An embedded-atom method interatomic potential for pd–h alloys. *Journal of Materials Research*, 23(03):704–718, 2008.

[105] James F Ziegler and Jochen P Biersack. *The stopping and range of ions in matter*. Springer, 1985.

[106] Rajendra R Zope and Yu Mishin. Interatomic potentials for atomistic simulations of the ti-al system. *Physical Review B*, 68(2):024102, 2003.