

PSE Molekulardynamik

Sheet 4: Thermostats, Rayleigh-Taylor instability and “falling drops”

Exercise on 04th December 2015

Task 1 “Thermostats”

In order to keep the simulated material at a constant temperature, as well as to be able to simulate heating or cooling processes, it is necessary to have a thermostat. The temperature of the phase is determined by the velocity of the particles. The dependency is given by

$$E_{\text{kin}} = \frac{\text{\#dimensions} \cdot \text{\#particles}}{2} k_B T$$

or dimensionless ($k_B = 1$):

$$E_{\text{kin}} = \frac{\text{\#dimensions} \cdot \text{\#particles}}{2} T$$

where E_{kin} can be calculated as the sum of the kinetic energy of all particles:

$$E_{\text{kin}} = \sum_{i=1}^{\text{\#particles}} \frac{m_i \langle v_i, v_i \rangle}{2}$$

As seen in the tutorial, we can scale the velocities $v_i \leftarrow \beta v_i$. The scaling factor can be calculated as:

$$\beta := \sqrt{\frac{T^D}{T}}$$

with desired temperature T^D to be imposed on the MD system.

- Implement such a thermostat.
- The input parameters are

- initial temperature:
the initial scaling value for the velocity for the Maxwell-Boltzmann distribution is calculated as:

$$v = \sqrt{\frac{2E_{\text{kin}}}{\text{\#dimensions} \cdot \text{\#particles} \cdot m}}$$

The initialization of the phase to a given temperature may be moved to the thermostat and be called after all particles have been instantiated.

For our applications (at least for the moment) it is ok to set $m = 1$ and hardcode it.

However, the initialization with the Brownian Motion should be optional.

- the number of timesteps $n_{\text{thermostat}}$ after which the thermostat is applied
- Furthermore for simulations with a different target temperature:
 - target temperature
 - temperature difference, i.e. the step size in which the temperature should be changed
 - the number of timesteps after which the temperature has to be changed
- Adapt the XML and tests accordingly.

Task 2 “Simulation of the Rayleigh-Taylor instability”

The Rayleigh-Taylor instability occurs at the interface of two fluids when a fluid with higher density resides on a fluid with lower density and is subject to gravity.

In this task we will perform the simulation of a Rayleigh-Taylor instability. We will impose reflecting boundary conditions on the horizontal boundaries of the domain and periodic boundary conditions on the vertical boundaries.

The two different fluids are represented by two molecular cuboids.

- Implement periodic boundaries:
 - particles which left the domain at a boundary have to be inserted at the opposite boundary
 - particles from the boundary layer of the opposite boundary have to be inserted into the halo layer of the boundary.
- The fluids are exposed to gravity. Implement a force calculation routine which adds a gravitational force G (along the y-axis) according to

$$G = m \cdot g_{\text{grav}}$$

It should be possible to specify the factor g_{grav} .

- We have to model two different fluids. Extend the molecules so they can be of different type. Especially they may have different
 - mass
 - Lennard-Jones Parameters ϵ and σ

For two molecules of different type the Lennard-Jones parameters have to be determined according to the Lorentz-Berthelot mixing rule:

$$\sigma_{12} = \sigma_{21} = \frac{\sigma_{11} + \sigma_{22}}{2} \quad \epsilon_{12} = \epsilon_{21} = \sqrt{\epsilon_{11}\epsilon_{22}}$$

- For testing purpose, perform a small experiment:

$$\begin{array}{lll} g_{grav} = -12.44 & \text{delta.t} = 0.0005 & t_{end} = 25 \\ \text{Size of domain } L = 63, 36 & \text{initial temperature } T_{init} = 40 & n_{thermostat} = 1000 \\ r_{cutoff} = 2.5\sigma & & \end{array}$$

Liquid 1:

$$\begin{array}{lll} m_1 = 1.0 & \epsilon_1 = 1.0 & \sigma_1 = 1.0 \\ x_1 = \{0.6; 2\} & v_1 = \{0; 0\} & n_1 = \{50; 14\} \\ h_1 = 1.2 & & \end{array}$$

Liquid 2:

$$\begin{array}{lll} m_2 = 2.0 & \epsilon_2 = 1.0 & \sigma_2 = 0.9412 \\ x_2 = \{0.6; 19\} & v_2 = \{0; 0\} & n_2 = \{50; 14\} \\ h_2 = 1.2 & & \end{array}$$

- Perform the bigger experiment:

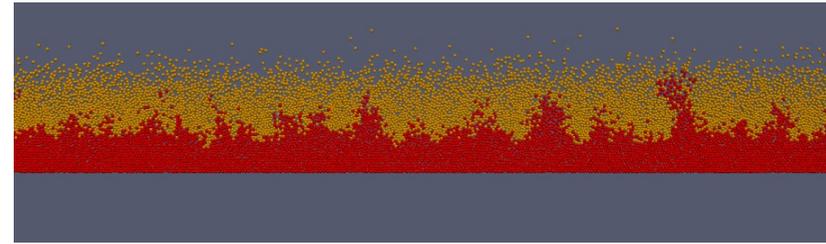
$$\begin{array}{lll} g_{grav} = -12.44 & \text{delta.t} = 0.0005 & t_{end} = 50 \\ \text{Size of domain } L = 300, 54 & \text{initial temperature } T_{init} = 40 & n_{thermostat} = 1000 \\ r_{cutoff} = 2.5\sigma & & \end{array}$$

Liquid 1:

$$\begin{array}{lll} m_1 = 1.0 & \epsilon_1 = 1.0 & \sigma_1 = 1.2 \\ x_1 = \{0.6; 2\} & v_1 = \{0; 0\} & n_1 = \{250; 20\} \\ h_1 = 1.2 & & \end{array}$$

Liquid 2:

$$\begin{array}{lll} m_2 = 2.0 & \epsilon_2 = 1.0 & \sigma_2 = 1.1 \\ x_2 = \{0.6; 27\} & v_2 = \{0; 0\} & n_2 = \{250; 20\} \\ h_2 = 1.2 & & \end{array}$$



Task 3 “Simulation of a falling drop”

In this task we want to simulate a drop falling into a bassin filled with liquid.

- Write an output component which saves the phase space to a text file, in a similar format like the input file for the simulation of the solar system. Similarly, the output file will be used as the input for a new simulation.

Note: Here you should save the complete state of the molecules to the file (including old force, type, and so on...)

- Initially, we let gravity act on the fluid in the bassin until $t = 15$ so that the molecules arrange in a natural way (equilibration). Then the phase space has to be saved to a file which we use as input for the further simulation.

Use the following parameters:

$$\begin{array}{lll} g_{grav} = -12.44 & \text{delta.t} = 0.0005 & t_{end} = 15 \\ \text{Size of domain } L = \{303, 180\} & \text{initial temperature } T_{init} = 0.5 & n_{thermostat} = 1000 \\ r_{cutoff} = 2.5\sigma & & \end{array}$$

Liquid:

$$\begin{array}{lll} m_1 = 1.0 & \epsilon_1 = 1.0 & \sigma_1 = 1.2 \\ x_1 = \{1.5; 2\} & v_1 = \{0; 0\} & n_1 = \{250; 50\} \\ h_1 = 1.2 & & \end{array}$$

- Now perform the actual simulation. Basically, we use the equilibrated fluid, a drop (represented by a sphere), and let gravity accelerate the drop. The thermostat is now switched off.

Use the parameters and the output of the above simulation and additionally specify a sphere with: $x_1 = \{150; 150\}$ $n_1 = 20$ $t_{end} = 40$

Task 4 “Performance Measurement”

As simulations are pretty long-running, we have to

- Change the application so that you can measure the runtime of an iteration (take the average over several iterations).
Note: Use the function `gettimeofday()`
- Create a profile of your application with GNU `gprof` and Intel®VTune™ Amplifier XE. Use a large, representative scenario (≥ 10000 molecules), with io switched off. Which parts of the code consume the most runtime?
- Hand in measurements of your code with both tools. Use the batch system of the MAC-cluster (queues: `snb` and `bdz`) to obtain these, http://www.mac.tum.de/wiki/index.php/MAC_Cluster, <http://www.lrz.de/services/compute/linux-cluster/>.
- Think about how you could improve the performance of your code, especially of the compute intensive parts, but also with respect to memory consumption. Consider features of the c++-language, e.g. inlining, templates, as well as algorithmic features (e.g. reuse calculated values, avoid unnecessary square-roots).

Task 5 “Tuning the sequential Performance”

For the last task, you had to think about ways to tune the code. Perform each of the steps (you may sum up similar ones into one step), and measure performance after each one.

Compile the code with the GNU Compiler (`gcc`) and the Intel C++ Compiler (`icpc`). Compare both after major optimisation steps!

- algorithmic
- code related
- compiler (Intel: `-O3 -ip -ipo -fast`); Profile-Guided Optimization.
Refer to the **Intel C++ Compiler User and Reference Guide**

Hand in measurements of your improvements profiled with GNU `gprof` and Intel®VTune™ Amplifier XE. Use the batch system of the MAC-Cluster again.

Contest: Report the runtime of the initial 1,000 iterations of the large experiment of task 2. Which group will deliver the best performance?

Have fun!

Deliver by 16th December 2015, 3 PM per mail to neumanph@in.tum.de!