Summer Term
2013

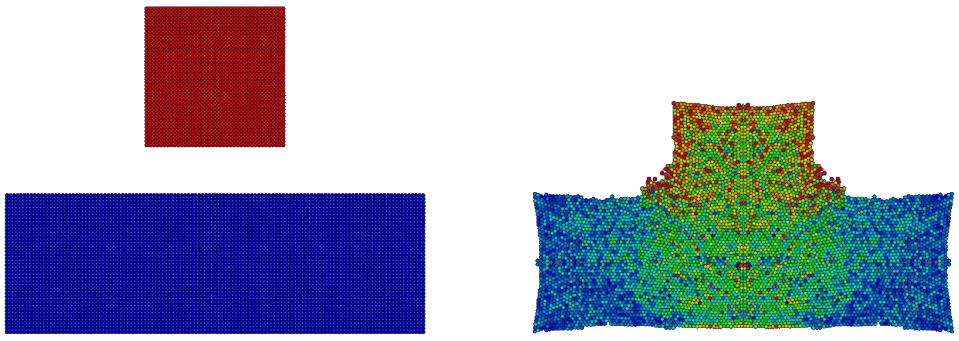
Prof. Dr. U. Rüde
Dominik Bartuschat
Kristina Pickl

# Simulation und wissenschaftliches Rechnen 2
# Assignment 4

Exercise 4 (*Molecular Dynamics*)

## Tasks

1. In this assignment, we will apply the linked cell algorithm to our simulation of two colliding blocks consisting of several thousand particles:



The forces between the particles will act according to the Lennard-Jones potential. The Lennard-Jones potential for a pair of particles $(i, j)$ is given by:

$$U(r_{ij}) = 4 \cdot \epsilon \left( \left( \frac{\sigma}{r_{ij}} \right)^{12} - \left( \frac{\sigma}{r_{ij}} \right)^{6} \right) \tag{1}$$

Depending on the distance $r_{ij}$ between two particles, the change of potential energy results in an attractive or a repulsive force between these two particles. The parameters $\epsilon$ and $\sigma$ change the strength of the potential and the distance at which the interaction changes from attractive to repulsive, respectively. The force on particle $i$ is obtained by taking the negative gradient of the potential function with respect to the particle position $\mathbf{x}_i$:

$$\mathbf{F}_i = -\nabla_{\mathbf{x}_i} \sum_{k=1}^{N} \sum_{l=k+1}^{N} U(r_{kl}) = 24\epsilon \cdot \sum_{j=1, i\neq j}^{N} \frac{1}{r_{ij}^2} \left( \frac{\sigma}{r_{ij}} \right)^{6} \cdot \left( 1 - 2 \cdot \left( \frac{\sigma}{r_{ij}} \right)^{6} \right) \mathbf{r}_{ij} \tag{2}$$

where $N$ is the total number of particles and $\mathbf{r}_{ij}$ denotes the distance vector $\mathbf{x}_j - \mathbf{x}_i$ in contrast to the distance $r_{ij} = \|\mathbf{r}_{ij}\|_2$. To simulate the dynamics of the interacting particles, we will employ the Störmer-Verlet scheme to integrate the particle positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ over small time steps $\Delta t$:

$$\begin{aligned} \mathbf{x}_i^{n+1} &= \mathbf{x}_i^n + \Delta t \mathbf{v}_i^n + \frac{\mathbf{F}_i^n \cdot \Delta t^2}{2m_i} \\ \mathbf{v}_i^{n+1} &= \mathbf{v}_i^n + \frac{(\mathbf{F}_i^n + \mathbf{F}_i^{n+1})\Delta t}{2m_i} \end{aligned} \tag{3}$$

Here $m_i$ denotes the mass of particle $i$ and superscripts $n$ and $n+1$ identify the current and next time step, respectively. This results in the following overall simulation procedure:

**Algorithm 1** Algorithm for the molecular dynamics simulation.

```
1: // Initial force calculation
2: for number of time steps do
3:     // Update of the particle positions
4:     // Calculation of the acting forces
5:     // Update of the particle velocities
6: end for
```

2. Since the potential in Eq. (1) decays with $\frac{1}{r_{ij}^6}$, we can assume that the potential is negligible after a short distance $r_{cut}$. Therefore, we can approximate the forces acting on the particles by summing only over the particles within a prescribed distance $r_{cut}$ (the cut-off radius):

$$\tilde{F}_i = 24 \cdot \epsilon \sum_{\substack{j=1, i \neq j \\ r_{ij} \leq r_{cut}}}^{N} \frac{1}{r_{ij}^2} \left(\frac{\sigma}{r_{ij}}\right)^6 \cdot \left(1 - 2 \cdot \left(\frac{\sigma}{r_{ij}}\right)^6\right) \mathbf{r}_{ij} \tag{4}$$

   The *linked cell* algorithm is a very efficient method for an approximated analysis of forces and energies for rapidly decaying potentials. The idea behind the linked cell method is to divide the physical simulation space into smaller cells. The influence of particles is limited to the own and all neighboring cells. Therefore a lot of computational time is saved without losing much accuracy. Assume a three-dimensional domain $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$. Subdivide the domain into cells which are *at least* of size $r_{cut}$ in each dimension.

3. The boundary conditions are *periodic* in all directions. This means that if particles leave the domain on one side, they will reenter on the opposite side (see [2]). Note that cells near the boundary are also under the influence of the periodic neighbors.

4. We provide you with input *data files* describing the particles' masses, initial positions and velocities. The data is stored in a simple ASCII format. On each line of the input file one particle is specified. For each particle the floating point values for its mass, initial $x$, $y$ and $z$ position and initial $x$, $y$ and $z$ velocity are stated in this order separated by white space.

   You can download several scene descriptions for the example above with a varying number of particles: `blocks-small.dat`, `blocks-medium.dat`, `blocks-large.dat`

5. All simulation parameters except for the particle properties need to be read from a *parameter file*. The parameter file will also be in a simple ASCII format, where on each line there is one key and one value separated by white space. Implement a separate class `ParameterReader` to read in such a file with the following interface:

```
bool readParameters(const std::string& filename);
inline bool IsDefined(const std::string& key) const;
template<typename Type>
inline void GetParameter(const std::string& key, Type &value) const;
```

A parameter file `blocks.par` suitable for the example problem above is listed in the following. You can also download it from the website:

```
    name       blocks
    vis_space  1600
    t_start    0.0
    t_end      8.0
    delta_t    0.00005
    x_min      0.0
    y_min      0.0
    z_min      0.0
    x_max      224.4924
    y_max      224.4924
    z_max      224.4924
    r_cut      2.5
    epsilon    5.0
    sigma      1.0
```

The parameter `name` will be used as the basename of the visualization files and `vis_space` specifies after how many time steps a new visualization output file should be generated. Furthermore `t_start` and `t_end` specify the start and end time of the simulation. All other parameters correspond to variables already introduced above.

6. Visualize your results using the open source visualization tool *Paraview*. In order to execute it on a LSS CIP pool computer, type `/usr/local/bin/paraview` on the command prompt. Write your data as legacy VTK files [1] like in the previous assignment. As an example consider the following annotated VTK file:

```
# vtk DataFile Version 3.0     # file header
SiWiRVisFile                   # this is only a comment
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 4 DOUBLE                # 4 particles
-1.0 -1.0 0.0                  # x, y and z coordinates of first particle
 1.0 -1.0 0.0
-1.0  1.0 0.0
 1.0  1.0 0.0
POINT_DATA 4
SCALARS mass double            # scalar field with identifier “mass”
LOOKUP_TABLE default           # default color table
1                              # mass of first particle
```

```
2
3
4
VECTORS force double             # vector field with identifier “force”
 0.0  1.0 0.0                    # force acting on the first particle
-1.0  0.0 0.0
 1.0  0.0 0.0
 0.0 -1.0 0.0
VECTORS velocity double          # velocity field
...
```

In order to create time series, the VTK files must be tagged with a number: `example0.vtk`, `example1.vtk`, .... *Paraview* automatically detects these time series and offers the functionality to visualize them in an animation. The tag for the time step and the file extension `.vtk` should be appended to the `name` given in the parameter file.

7. Please hand in your solution to this exercise until **Wednesday, July 10, 2013.** Make sure the following requirements are met:

   (a) The program should be compilable with a Makefile.

   (b) The program should compile without errors or warnings with (at least) the following g++ compiler flags:

   ```
   -Wall -ansi -pedantic
   ```

   (c) The program should be callable by `./mdsim [parameter file] [data file]`.

   (d) Use double precision floating-point calculations.

   (e) When submitting the solution, remove all temporary files from your project directory with the name `ex04_groupXX/` and pack it using the following command:

   ```
   tar -cjf ex04_groupXX.tar.bz2 ex04_groupXX/
   ```

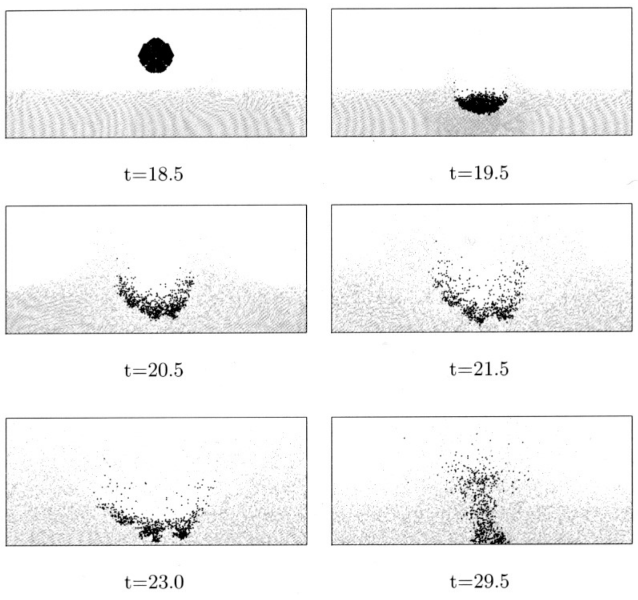
   where `XX` stands for your group number and `ex04_groupXX/` contains your solution. Then send the archive to the course mailing list:

   ```
   siwir2@i10.informatik.uni-erlangen.de
   ```

## Credits

1. Up to 25 credits are awarded if your program correctly performs the above tasks and fulfills all of the above requirements. Submissions with compile errors are marked with 0 points.

2. Up to 5 bonus credits can be gained in the following way:
   Extend your program such that it can simulate a drop falling into a basin filled with a fluid, as shown below (taken from [2]):



   This includes the incorporation of an external gravitational field affecting all particles, as well as initial particle velocies according to a Maxwell-Boltzmann distribution. In this scenario, reflecting boundary conditions are applied at all walls. Thus, particles can not leave the simulation domain. More details can be found in [2].

## References

[1] http://www.vtk.org/VTK/img/file-formats.pdf

[2] M. Griebel, S. Knapek, and G. Zumbusch. *Numerical Simulation in Molecular Dynamics*. 2007.