



Chair for System Simulation

Harald Köstler, Dr.-Ing.

Rafael Ravedutti Lucio Machado

# High End Simulation in Practice

Summer Term 2021

## Assignment 2: Molecular Dynamics

### Suggested Work Schedule

- solve the assignment: 10.05.2021 – 23.05.2021
- present the assignment: 24.05.2021 – 28.05.2021

There are no hard deadlines, however, we highly recommend sticking to this schedule as there will be new assignments regularly.

### Mandatory Submission Guidelines

- submit in groups of two to four (you can use the forum to find team members)
- upload your solution as a group to StudOn
- your solution MUST include an appropriate Makefile
- the examination will contain questions about the exercises, so make sure you understood everything
- additional requirements may be stated in the tasks

In this exercise you will write your first simulation application. You are free to use either CUDA or OpenCL (of course you can also code both versions).

## 1 Theory

In molecular dynamics the task is to trace the trajectories of an ensemble of  $N$  given particles based on the physical principles described below. A particle  $i \in \{1, 2, \dots, N\}$  is considered to be a point mass, and is characterized by its position  $\mathbf{x}_i$ , velocity  $\mathbf{v}_i$  and mass  $m_i$ . Additionally we will need to store the acting force  $\mathbf{F}_i$  and the force that was acting in the previous timestep  $\mathbf{F}_i^{old}$ . *Newton's law of motion* states that  $\mathbf{F}_i = m_i \mathbf{a}_i = m_i \ddot{\mathbf{x}}_i$  which can be reformulated as system of ODEs:

$$\begin{aligned}\dot{\mathbf{x}}_i &= \mathbf{v}_i \\ \dot{\mathbf{v}}_i &= \frac{\mathbf{F}_i}{m_i}.\end{aligned}$$

Neglecting how the computation of the force is done for a moment, this system can be integrated numerically by using the so-called *Velocity Verlet* scheme, see Algorithm 1.

---

**Algorithm 1** Velocity Verlet in pseudocode.

---

```

procedure VELOCITY VERLET
   $t \leftarrow t_0$ 
  Calculate initial forces  $\mathbf{F}_i \quad i \in \{1, 2, \dots, N\}$ 
  while  $t \leq t_{end}$  do
     $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \cdot \mathbf{v}_i + \frac{\mathbf{F}_i \cdot \Delta t^2}{2m_i} \quad i \in \{1, 2, \dots, N\}$ 
     $\mathbf{F}_i^{old} \leftarrow \mathbf{F}_i \quad i \in \{1, 2, \dots, N\}$ 
    Calculate new forces  $\mathbf{F}_i \quad i \in \{1, 2, \dots, N\}$ 
     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{(\mathbf{F}_i^{old} + \mathbf{F}_i) \cdot \Delta t}{2m_i} \quad i \in \{1, 2, \dots, N\}$ 
     $t \leftarrow t + \Delta t$ 
  end while
end procedure
```

---

The *Lennard-Jones-Potential* (see figure 1) shall be used for the particle interactions, so that the force between two particles  $i$  and  $j$  with distance vector  $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$  can be computed as:

$$\mathbf{F}_2^{LJ}(\mathbf{x}_i, \mathbf{x}_j) = \frac{24 \varepsilon}{|\mathbf{x}_{ij}|^2} \left( \frac{\sigma}{|\mathbf{x}_{ij}|} \right)^6 \left( 2 \left( \frac{\sigma}{|\mathbf{x}_{ij}|} \right)^6 - 1 \right) \mathbf{x}_{ij} \quad (1)$$

$\varepsilon$  determines how strong the maximal attraction is and  $\sigma$  characterizes the stable distance between two particles.

The total force acting on particle  $i$  calculates as:

$$\mathbf{F}_i(\mathbf{x}) = \sum_{\substack{j=1 \\ i \neq j}}^N \mathbf{F}_2^{LJ}(\mathbf{x}_i, \mathbf{x}_j). \quad (2)$$

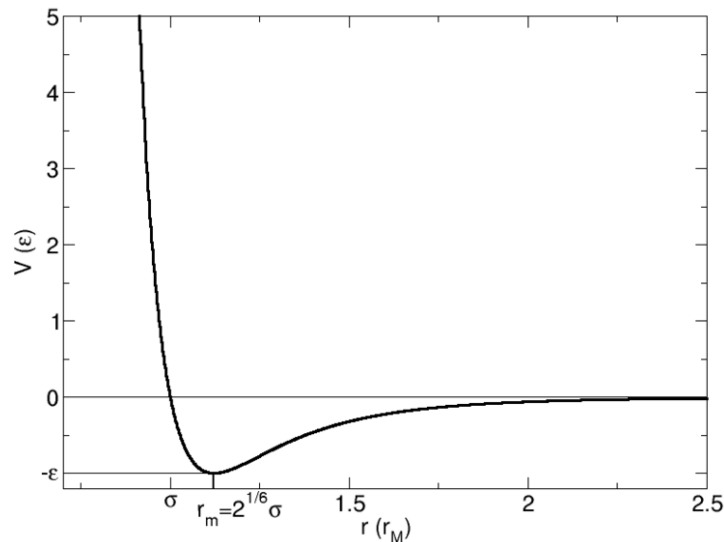


Figure 1: Lennard-Jones potential, source: <http://upload.wikimedia.org/wikipedia/commons/5/5a/12-6-Lennard-Jones-Potential.png>.

Help regarding the algorithm can also be found in [3].

## 2 Implementation

Your task is to implement the *Velocity Verlet* scheme for a molecular dynamics simulation as described above. Your implementation shall have the following functionality:

1. Particles shall be modeled in three-dimensional space, i.e.  $\mathbf{x}_i, \mathbf{v}_i, \mathbf{a}_i, \mathbf{F}_i \in \mathbb{R}^3$ . Of course, setting values of the third dimension to zero must allow to conduct 2D tests!
2. All calculations within the Verlet algorithm, i.e. force calculation, force copying, velocity and position update, shall be implemented as kernels in your chosen language (CUDA / OCL), such that the actual work is executed on the GPU (or the chosen OpenCL device). The data should remain in device buffers on the device and only be copied to the main memory if needed, e.g. for output.
3. Use double precision for all your calculations.
4. The file with simulation parameters shall look like this:

```
part_input_file      parfile.in
timestep_length      0.01
time_end              1.0
epsilon               5.0
sigma                 1.0
part_out_freq         10
part_out_name_base    results_part_
vtk_out_freq          5
```

```

vtk_out_name_base    results_vtk_
cl_workgroup_1dsize  128

```

These parameters shall be parsed by your program. You are free to replace the OpenCL workgroup size with an appropriate equivalent when using CUDA. In this case, you also have to adapt all sample input files.

5. The input particle configuration shall be read in from another ASCII file (`parfile.in` in this example) that has one line per particle and looks like:

```

N
m_1 x_1 y_1 z_1 vx_1 vy_1 vz_1
m_2 x_2 y_2 z_2 vx_2 vy_2 vz_2
...
m_N x_N y_N z_N vx_N vy_N vz_N

```

The first line contains the total number of particles, the following lines one particle each (mass, 3D position vector, 3D velocity vector).

6. For the output use the *VTK File format* [1], that can be visualized in paraview [2]. You shall use two parameters (`part_out_freq` and `vtk_out_freq`) in the simulation parameters file that specify after how many time steps the according output is written. The variables `part_out_name_base` and `vtk_out_name_base` shall be used to specify the file name bases that should be extended by a counting variable (so one file per printed time step). One vtk output file shall contain the positions as unstructured grid, the mass as scalar field and the velocity as vector field. Please format the values of positions, mass and velocity with the C++ format specifier `std::fixed` that enforces a fixed width for floating-point values. This will it make easier to compare your results to the reference solutions.

Note that the 0th time step shall be written to the file with index 0, i.e. before the first position update.

7. We provide some test data on the course's StudOn page. After unpacking the test data you can find different setups in the *input* directory: Two particles that are at a stable distance (i.e. should not move at all), at a distance where they attract each other and on a repelling distance. Furthermore, there is an example of two colliding blocks, each consisting of 16 particles. The *.par* files contain the parameters, the *.in* files contain the particle setup (positions and velocities).

When developing, it might be a good idea to get the *stable* example working first and then one can continue with the other two-particle setups before running the *blocks* example.

8. After your simulation is running, try to generate more elaborate setups with a higher number of particles. How many particles will fit into your device memory? You will most likely note that simulation times increase with a sub-optimal complexity. Determine the algorithmic complexity by measuring the performance of your compute kernels for different numbers of particles. Prepare these results for discussion.

## References

- [1] <http://www.vtk.org/VTK/img/file-formats.pdf>
- [2] <http://www.paraview.org>
- [3] M. Griebel, S. Knapek, G. Zumbusch. *Numerical Simulation in Molecular Dynamics*, Texts in Computational Science and Engineering, Springer, 2007.