

Contents

1	Langevin Equation	2
2	Integrator	2
3	Reduced Units	2
4	Lennard-Jones Potential	3
5	Implementation details	4
5.1	General algorithm	4
5.2	Periodic Boundary Conditions	4
5.3	Force calculation	5
5.4	OpenMP	6
6	Measurements	7
7	Results for the Lennard-Jones-Fluid	8
8	Polymer Physics	10
8.1	Normal Coordinates	11
8.2	Excluded Volume	12

1 Langevin Equation

The Langevin Equation for the i -th particle is:

$$m_i \ddot{\vec{x}}_i = \vec{F}(\vec{x}_i(t)) - \gamma_i m_i \dot{\vec{x}}_i + \vec{R}_i \quad (1)$$

where $\vec{F}(\vec{x}_i(t)) = \vec{F}_i$ is the interaction force, γ_i is the collision frequency and \vec{R}_i represents a random force which is related to the mass m_i , the Temperature T and the collision frequency γ_i by:

$$\langle \vec{R}_i(t) \vec{R}_j(t') \rangle = 2m_i k_B T \gamma_i \delta_{i,j} \delta(t - t') \quad (2)$$

and

$$\langle \vec{R}_i(t) \rangle = 0 \quad (3)$$

This force can be calculated using a Gaussian distribution

$$\rho(\vec{R}_i) = \frac{1}{\sqrt{4\pi\gamma_i m_i k_B T}} \cdot e^{\frac{-\vec{R}_i^2}{4\gamma_i m_i k_B T}} \quad (4)$$

or using

$$\vec{R}_i(t) = \sqrt{2k_B T \gamma_i m_i} \eta(t) \quad (5)$$

where $\eta(t) = \dot{W}(t)$ is a white-noise ($W(t)$ being a Wiener process). In order to solve the differential equation one needs to integrate the equation of motion. A second-order integrator for Langevin equations was found by Eric Vanden-Eijnden and Giovanni Ciccotti [1] and is a generalization of the BBK integrator. The algorithm reduces to the velocity-Verlet algorithm when $\gamma_i = 0$.

2 Integrator

The Integrator found is

$$\begin{cases} \vec{v}^{(n+1/2)} = \vec{v}^{(n)} + \frac{1}{2} \Delta t \vec{f}(\vec{x}^{(n)}) - \frac{1}{2} \Delta t \gamma \vec{v}^{(n)} + \frac{1}{2} \sqrt{\Delta t \sigma} \vec{\xi}^{(n)} \\ \quad - \frac{1}{8} \Delta t^2 \gamma (\vec{f}(\vec{x}^{(n)}) - \gamma \vec{v}^{(n)}) - \frac{1}{4} \Delta t^{3/2} \gamma \sigma \left(\frac{1}{2} \vec{\xi}^{(n)} + \frac{1}{\sqrt{3}} \vec{\eta}^{(n)} \right) \\ \vec{x}^{(n+1)} = \vec{x}^{(n)} + \Delta t \vec{v}^{(n+1/2)} + \Delta t^{3/2} \sigma \frac{1}{\sqrt{12}} \vec{\eta}^{(n)} \\ \vec{v}^{(n+1)} = \vec{v}^{(n+1/2)} + \frac{1}{2} \Delta t \vec{f}(\vec{x}^{(n+1)}) - \frac{1}{2} \Delta t \gamma \vec{v}^{(n+1/2)} + \frac{1}{2} \sqrt{\Delta t \sigma} \vec{\xi}^{(n)} \\ \quad - \frac{1}{8} \Delta t^2 \gamma (\vec{f}(\vec{x}^{(n+1)}) - \gamma \vec{v}^{(n+1/2)}) - \frac{1}{4} \Delta t^{3/2} \gamma \sigma \left(\frac{1}{2} \vec{\xi}^{(n)} + \frac{1}{\sqrt{3}} \vec{\eta}^{(n)} \right) \end{cases} \quad (6)$$

where $(\vec{\xi}^{(n)}, \vec{\eta}^{(n)})$ are independent Gaussian variables with mean zero and covariance

$$\langle \xi_i^{(n)} \xi_j^{(n)} \rangle = \langle \eta_i^{(n)} \eta_j^{(n)} \rangle = \delta_{i,j} \quad \langle \xi_i^{(n)} \eta_j^{(n)} \rangle = 0 \quad (7)$$

3 Reduced Units

As molecular dynamics actions take place on a small time- and spacescale it's convenient to change the unit system to the system of reduced units (MD units) [4]. Following changes are made

$$\tilde{r} \rightarrow r\sigma \quad (8)$$

$$\tilde{E} \rightarrow E\epsilon \quad (9)$$

$$\tilde{m} \rightarrow mm_a \quad (10)$$

where σ and ϵ are paramter of the Lennard-Jones potential (ϵ governs the strength of the interaction and σ defines a length scale, both just numbers) and m_a is the atomic mass. In order to derive the factor for the conversion factor of time one can use the fact that for example the formula for the kinetic energy shouldn't change under unit system transformations. This leads to

$$\tilde{t} \rightarrow t\sqrt{\frac{m_a\sigma^2}{\epsilon}} \quad (11)$$

By setting $k_B = 1$ the MD unit of temperature is now also defined. The following table contains the most used quantities, as well as values for argon:

Physical quantity	Unit	Value for Ar
Length	σ	$3.4 \cdot 10^{-10} \text{ m}$
Energy	ϵ	$1.65 \cdot 10^{-25} \text{ J}$
Mass	m	$6.69 \cdot 10^{-26} \text{ kg}$
Time	$\sigma(m/\epsilon)^{1/2}$	$2.17 \cdot 10^{-12} \text{ s}$
Velocity	$(\epsilon/m)^{1/2}$	$1.57 \cdot 10^2 \text{ m/s}$
Force	ϵ/σ	$4.85 \cdot 10^{-12} \text{ N}$
Pressure	ϵ/σ^3	$4.20 \cdot 10^7 \text{ Nm}^{-2}$
Temperature	ϵ/k_B	120 K

Table 1: Conversation factors for MD units and specific values for argon

For the mass of argon the reduced timesunit corresponds to $2.161 \cdot 10^{-12} \text{ s}$. Thus a simulation timestep of $\Delta t = 0.005$ would correspond to approx. 10^{-14} s . For a liquid density of 0.942 g/cm^3 the reduced lenght of a box sized simulation region is $L = 1.218N^{1/3}$.

4 Lennard-Jones Potential

The Lennard-Jones Potential approximates the interaction between neutral atoms or molecules. A common expression is

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

where $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ and $r_{ij} = |\vec{r}_{ij}|$ is the distance between the particle i and j , σ is the finite distance at which the inter-particle potential is zero and ϵ is the depth of the potential well.

For simulations it is necessary to cut off the potential at a given radius r_c in order to reduce the simulationtime, and the potential becomes

$$U_{LJ} = \begin{cases} 4\epsilon \left(\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right) & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases} \quad (13)$$

To avoid a discontinuity at the cutoff it is commom to use a truncated and shifted potential. The following potential satisfies this condition

$$U = \begin{cases} U_{LJ} - U_{LJ}(r_c) & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases} \quad (14)$$

The force is

$$-\vec{\nabla}U = \vec{F}_{ij} = \begin{cases} \frac{48\epsilon}{\sigma^2} \left(\left(\frac{\sigma}{r_{ij}} \right)^{14} - \frac{1}{2} \left(\frac{\sigma}{r_{ij}} \right)^8 \right) \vec{r}_{ij} & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases} \quad (15)$$

When reduced units are used, the force reduces to the following form

$$-\vec{\nabla}U = \vec{F}_{ij} = \begin{cases} 48 \left(r_{ij}^{-14} - \frac{1}{2} r_{ij}^{-8} \right) \vec{r}_{ij} & r_{ij} < r_c \\ 0 & r_{ij} \geq r_c \end{cases} \quad (16)$$

5 Implementation details

5.1 General algorithm

In the case where the force depends on all other particles (e.g. Lennard-Jones-Potential) one is forced to split the algorithm in two different parts. One for the calculation of the half-step for the velocity and the new position. And the other part for the calculation of the new velocity. A pseudocode for the implementation could be like Algorithm 1 and a expiration chart of this algorithm may look like Figure 1.

Data: Initial positions and velocities of m particles $(x_1^1, x_2^1, \dots, x_m^1), (v_1^1, v_2^1, \dots, v_m^1)$

Result: Final positions $(x_1^n, x_2^n, \dots, x_m^n)$ and velocities $(v_1^n, v_2^n, \dots, v_m^n)$

initialization;

for $k \leftarrow 1$ **to** n **do**

for $i \leftarrow 1$ **to** m **do**

 calculate force for all m particles;

 update positions of all particles;

 calculate velocity half step;

 calculate force for new positions;

 update velocity;

end

end

Algorithm 1: Main algorithm

5.2 Periodic Boundary Conditions

When simulating liquids the problem is the huge amount of particles. To reduce this problem one can introduce periodic boundary conditions, where the simulation takes place in a container of some kind and considering an infinite, space-filling array of identical copies of the simulation region. This means, the particles near the boundaries effect the particles on the other side of the simulation space. This leads to a wraparound effect. If a particle moves outside the region, it has to be set on the other side. The following pseudocode takes care of this behaviour

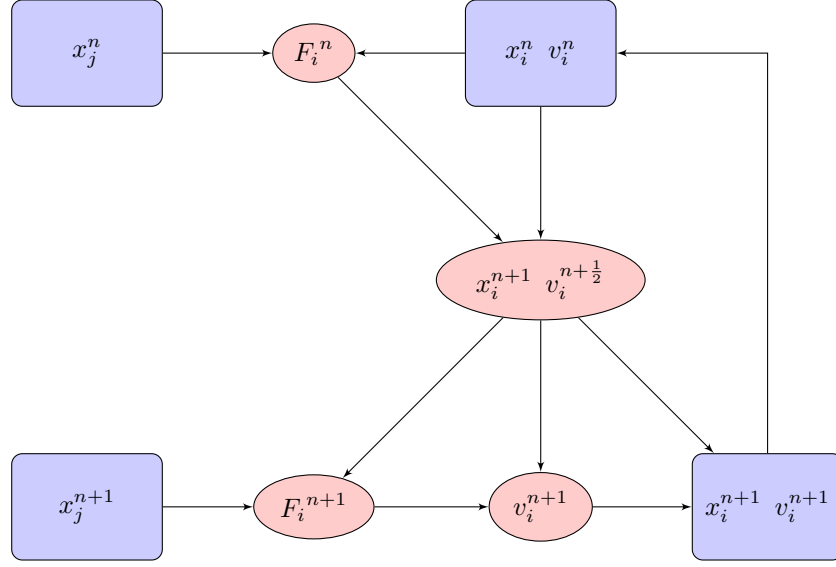


Figure 1: Expiration chart for the implementation. Red: calculation, Blue: storage

Data: New position x_i of particle, Simulationspace size L

Result: If particle leaves simulation region, update position based on periodic boundary conditions

initialization;

if $x_i > L$ **then**

$x_i = x_i - L$;

end

if $x_i < 0$ **then**

$x_i = x_i + L$;

end

Algorithm 2: Particle wraparound

5.3 Force calculation

For a system of N particles, most of the simulation time is spend in calculating the force between different particles. A way to reduce this calculation time is to cut off after a specific radius r_c and only include particle within this radius. For the Lennard-Jones potential typical values are $r_c = 2.5\sigma - 3.5\sigma$. A way to improve this technique was introduced by Loup Verlet [7], the so-called Neighbor list or Verlet list. The idea is to use a second radius $r_v > r_c$ and calculate the force for all particles within this radius, but only at every n -th step. In order to estimate r_v one can use

$$r_v = r_c + \Delta r \quad (17)$$

with

$$\Delta r \leq n\tilde{v}\Delta t \quad (18)$$

where Δt is the time step, \tilde{v} the root-mean-square. See 5.3 for a graphic representation of this method. This means that within the following $n - 1$ steps no particles, other the ones in the list, will move into the cut-off radius. A typical number for n is 10 - 20. The computation time is reduced by a factor of 10.

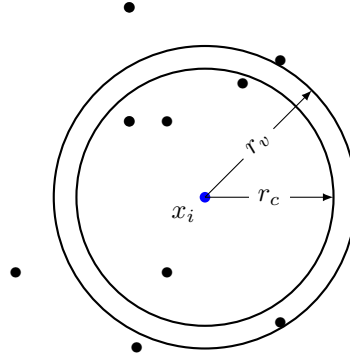


Figure 2: Neighbor list

For periodic boundary conditions the distance between two particles depends on the simulation region, as two particles near the boundaries can affect each other. A pseudocode for the calculation of the force F_i for the i -th particle is listed below (see algorithm 3).

Data: Position of all N particles at timestep m ($x_1^m, x_2^m, \dots, x_N^m$), boolean periodic

Result: Force F_i^m for i -th particle

initialization;

if $m \bmod n == 0$ **then**

for $j \leftarrow 1$ **to** N **do**

 calculate distance d from particle i to other particles;

if $d \leq r_v$ **then**

 add particle position to neighbor list;

end

end

end

calculate force on i -th particle based on neighbor list;

Algorithm 3: Neighbor list with boundary conditions

5.4 OpenMP

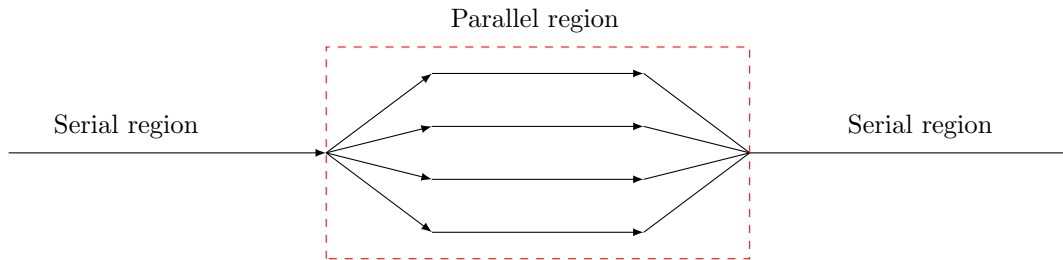


Figure 3: OpenMP

OpenMP [3] is application programming interface (API) which supports parallel programming in C/C++ and Fortran. The easiest way to use OpenMP is by splitting up independent loops in a given amount of threads, so that each thread can calculate parallel. To use OpenMP one needs to import the header `omp.h`. The following function call specifies the amount of threads which want to be used in the following code segment

```
omp_set_num_threads(NUM_THREADS);
```

It is not guaranteed that the specified amount of threads are created. A parallel region is introduced by

```
#pragma omp parallel for
```

and closed by

```
#pragma omp barrier
```

The last statement ensures that the computation continues only if every thread has finished it's task. It is possible to keep variables private inside of threads. The following example splits the loop over i in a specified amount of threads, but keeps j and k private for each thread to prevent unexpected changes of these variables. Another way of keeping variables private is to declare them inside the loops.

```
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel for private(j,k)
  for (i=0; i<amount; i++){
    int l;
    for (j=0; j<amount; j++){
      for (k=0; k<3; k++){
        // Computations
      }
    }
  }
#pragma omp barrier
```

Listing 1: Example: OpenMP

If two or more threads want to change a shared value, it causes problems if this happens at the same time. There are two ways to prevent this. First by using the *reduction* directive. This statement creates a copy of the argument for each thread and uses the unity element for the specified operation to combine them after the parallel region.

```
#pragma omp parallel for reduction(+:variable)
```

This method doesn't work for class members. In this case the *atomic* directive can be used.

```
#pragma omp atomic
```

The statement after this directive is protected and can only be written if no other thread is currently writing. If a compiler doesn't support OpenMP, the directives cause errors on this systems. To avoid this, the following statement is used

```
#ifdef _OPENMP
  #include <omp.h>
#endif
```

Listing 2: To avoid error messages where OpenMp is not installed

6 Measurements

The kinetic and potential energies are

$$E_K = \frac{1}{2} \sum_{i=1}^N \vec{v}_i^2 \quad (19)$$

$$E_U = 4 \sum_{i \leq i < j \leq N} (r_{ij}^{-12} - r_{ij}^{-6}) \quad (20)$$

The temperature of the system is

$$T = \frac{1}{3N} \sum_{i=1}^N \vec{v}_i^2 \quad (21)$$

The pressure is defined in terms of the virial expression

$$PV = NT + \frac{1}{3} \left\langle \sum_{i=1}^N \vec{x}_i \cdot \vec{F}_i \right\rangle \quad (22)$$

7 Results for the Lennard-Jones-Fluid

If the friction constant γ is set to zero, the integrator reduces to the velocity verlet integrator and therefore energy should be conserved. To qualify this, the shift with respect to the initial value is calculated. This is done in the following way

$$\Delta E = \frac{E(t) - E(0)}{E(t)} \quad (23)$$

The energy evolution over time is plotted in figure 4.

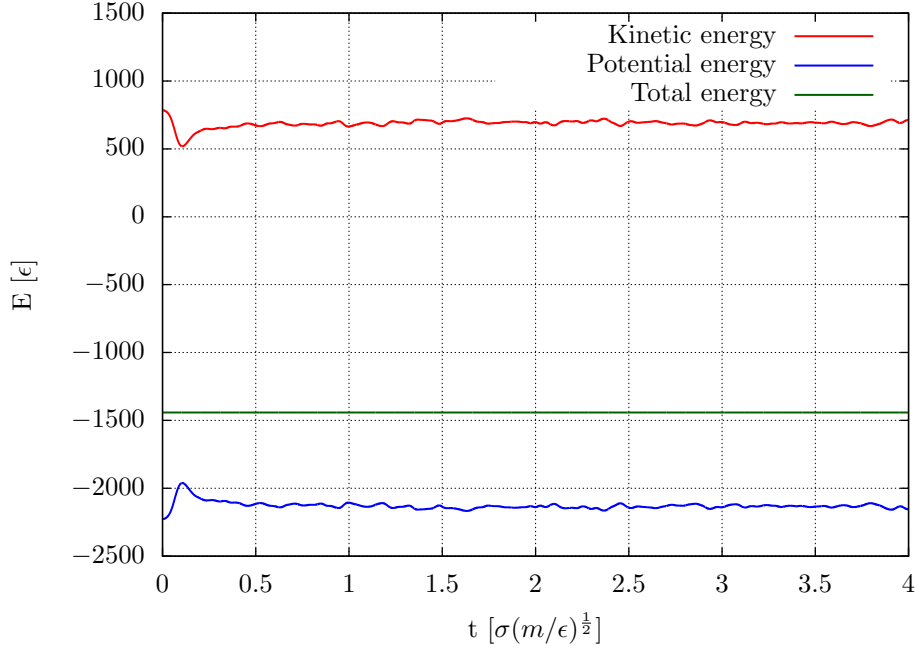


Figure 4: Energy evolution ($\gamma = 0$, $N = 512$, $\rho = 0.6$, $\Delta t = 0.001$, $T = 1$)

The energyshift can be seen in figure 5.

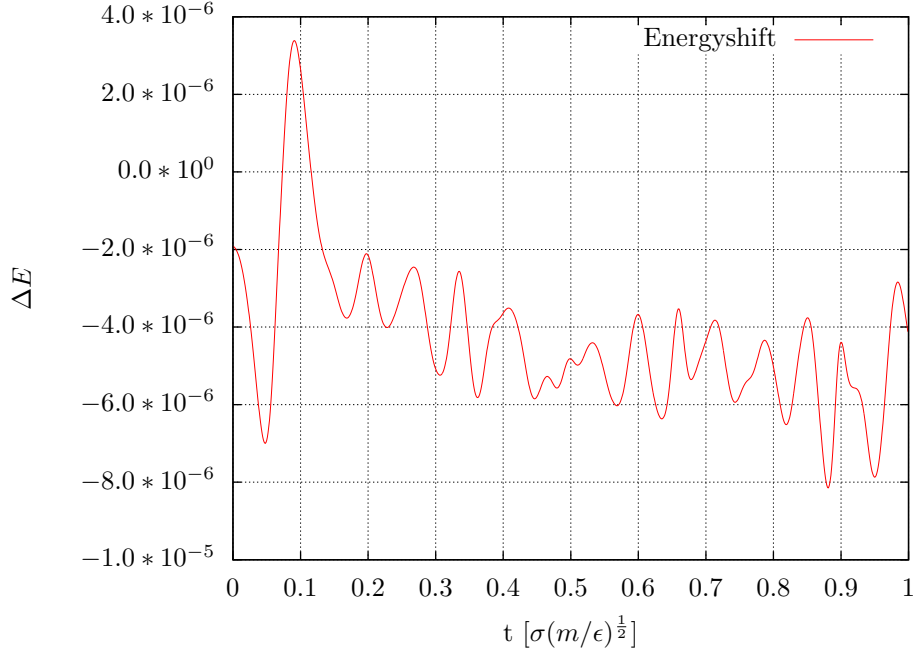


Figure 5: Energyshift ($\gamma = 0$, $N = 512$, $\rho = 0.6$, $\Delta t = 0.001$, $T = 1$)

The Langevin-Integrator works as a thermostat, i.e. the system is coupled to a heat bath. Like the kinetic energy and the pressure, the temperature fluctuates. The difference between a system with and without a heat bath can be seen in figure 6.

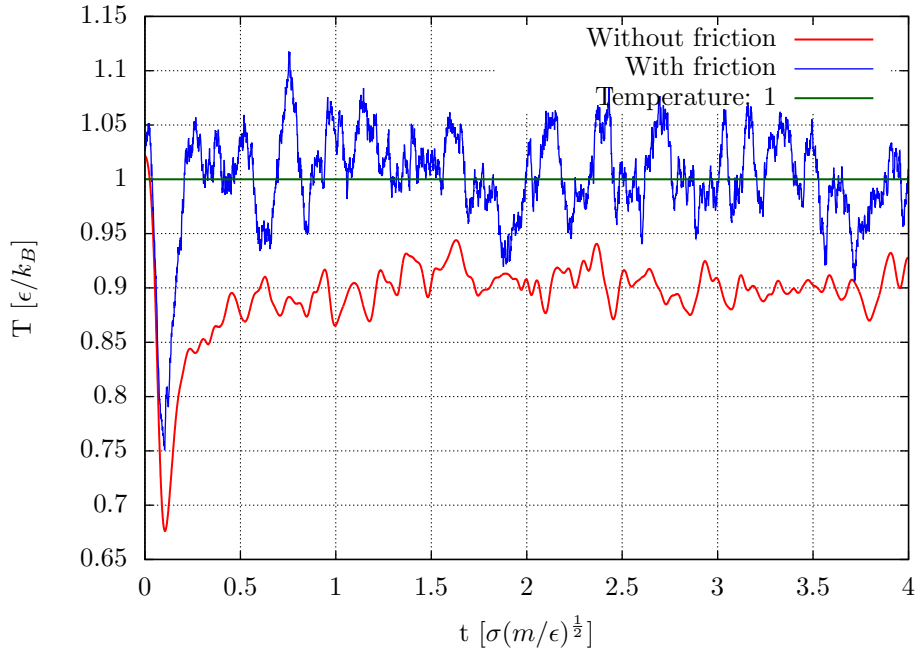


Figure 6: Temperature ($N = 512$, $\rho = 0.6$, $\Delta t = 0.001$, $T = 1$) Red: $\gamma = 0$, Blue: $\gamma = 5$, Green: Initial temperature 1

The pressure can be compared to the value for the pressure of the ideal gas.

$$PV = Nk_B T \quad (24)$$

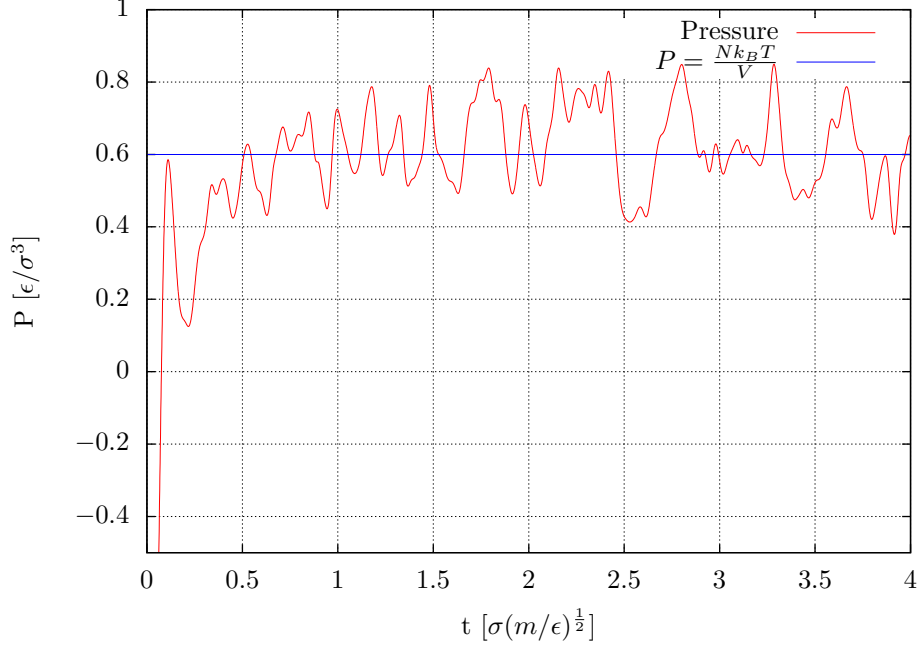


Figure 7: Pressure ($\gamma = 0$, $N = 512$, $\rho = 0.6$, $\Delta t = 0.001$, $T = 1$)

8 Polymer Physics

A simple way to represent a polymer is by using springs between the atoms. This model is called the Rouse model [5]. A schematic representation of this model can be seen in figure 8.

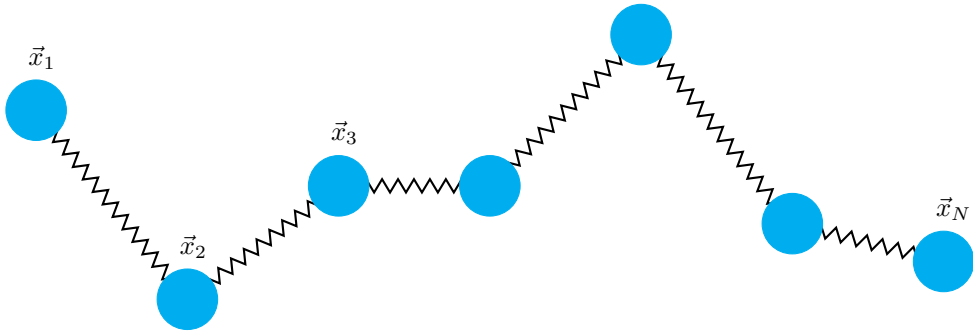


Figure 8: Rouse model

The potential energy of one bond is

$$U_{ij} = \frac{1}{2}k(\vec{x}_i - \vec{x}_j)^2 \quad (25)$$

with spring constant $k = \frac{3k_B T}{b^2}$ where b is the statistical segment length and $b^2 = \frac{\langle R_{ee}^2 \rangle}{N-1}$ with $\langle R_{ee}^2 \rangle$ the second moment of end-to-end vector.

The force on the i -th bond is

$$\vec{F}_i = k(\vec{x}_{i+1} - 2\vec{x}_i + \vec{x}_{i-1}) \quad (26)$$

In the cases where $i = 1$ set $\vec{x}_0 = \vec{x}_1$ and where $i = N$ set $\vec{x}_{N+1} = \vec{x}_N$.

This model is also called the Gaussian chain, because every bond length satisfies the distribution [2]

$$\psi(\vec{x}) = \left(\frac{3}{2\pi b^2}\right)^{\frac{3}{2}} \exp\left(-\frac{3\vec{x}^2}{2b^2}\right) \quad (27)$$

For given m, n the distribution of the vector $\vec{x}_n - \vec{x}_m = \sum_{i=m}^{n-1} (\vec{x}_{i+1} - \vec{x}_i)$ is given by

$$\phi(\vec{x}_n - \vec{x}_m, n - m) = \left(\frac{3}{2\pi b^2 |n - m|}\right)^{\frac{3}{2}} \exp\left(-\frac{3(\vec{x}_n - \vec{x}_m)^2}{2|n - m|b^2}\right) \quad (28)$$

A plot of the distribution due to the simulation and the analytic one can be found in figure 9.

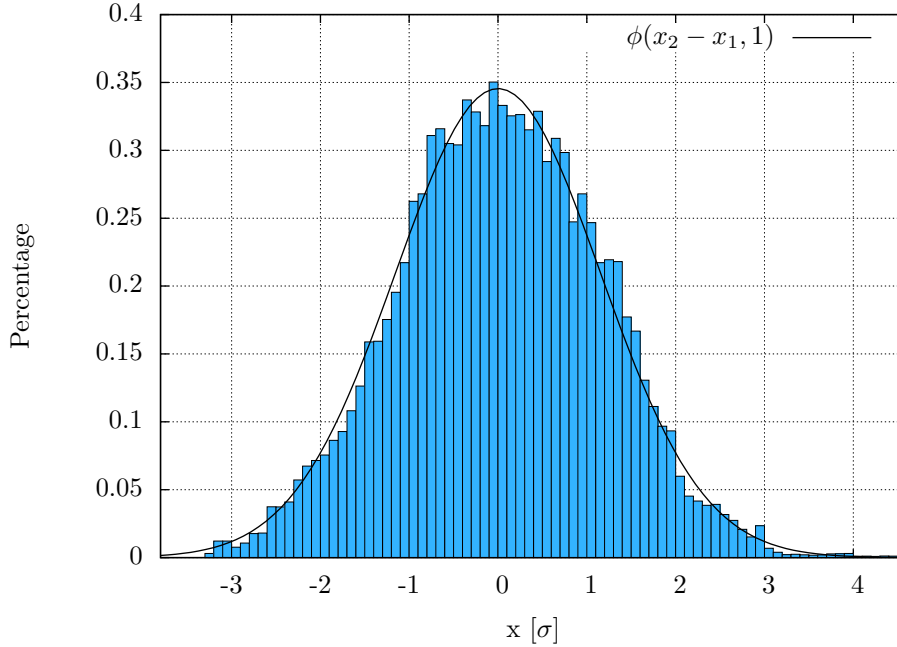


Figure 9: End-to-end distribution between the first and second bead ($\gamma = 1$, $N = 10$, $\Delta t = 0.001$, $T = 1$)

8.1 Normal Coordinates

Measurements in a molecular dynamics simulation can only be performed when the system is equilibrated. A way to give an estimate of the relaxation time needed can be done by introducing normal coordinates [6]. The equation of motion is

$$m_i \ddot{\vec{x}}_i = k(\vec{x}_{i+1} - 2\vec{x}_i + \vec{x}_{i-1}) - \gamma m_i \dot{\vec{x}}_i + \vec{R}_i \quad (29)$$

The term $m_i \ddot{\vec{x}}_i$ is important for vibrational motion, but is negligible for long time scales (μs to s). The mass m_i is dropped for simplicity. The equation of motion becomes

$$\gamma \dot{\vec{x}}_i = k(\vec{x}_{i+1} - 2\vec{x}_i + \vec{x}_{i-1}) + \vec{R}_i \quad (30)$$

The i -th normal coordinate is defined by

$$\vec{q}_i = \frac{1}{N} \sum_{n=1}^N \cos\left(\frac{in\pi}{N}\right) \vec{x}_n \quad (31)$$

The 0-th normal coordinate describes the center of mass motion

$$\vec{q}_0 = \frac{1}{N} \sum_{n=1}^N \vec{x}_n \quad (32)$$

The first normal coordinate is a vector from the center of mass of the first half pointing to the center of mass of the second half. Therefore \vec{q}_1 describes the rotation of the chain.

The equation of motion in normal coordinates is

$$\xi_i \dot{\vec{q}}_i = -k_i \vec{q}_i + \vec{g}_i \quad (33)$$

with

$$\xi_i = \begin{cases} N\gamma & \text{if } i = 0 \\ 2N\gamma & \text{if } i \neq 0 \end{cases} \quad (34)$$

The force constant k_i of the i -th mode is given as

$$k_i = k \frac{\xi_i}{\gamma} \left(\frac{i\pi}{N}\right)^2 = \frac{6\pi^2 k_B T}{Nb^2} i^2 \quad (35)$$

The random force \vec{g}_i is defined by

$$\vec{g}_i = \frac{\xi_i}{\gamma} \frac{1}{N} \sum_{n=1}^N \cos\left(\frac{in\pi}{N}\right) \vec{R}_n \quad (36)$$

The equation for \vec{q}_i does not depend on other $\vec{q}_j (j \neq i)$ and is therefore decoupled. The relaxation time is defined as

$$\tau_i = \frac{\xi_i}{k_i} \quad (37)$$

Expressed with the relaxation time the equation of motion is

$$\dot{\vec{q}}_i = -\frac{1}{\tau_i} \vec{q}_i + \frac{1}{\xi_i} \vec{g}_i \quad (38)$$

For the Rouse-Model:

$$\tau_i = \frac{\gamma N^2 b^2}{3\pi^2 k_B T} \frac{1}{i^2} \quad (39)$$

As $\frac{1}{\tau_0} = 0$, τ_1 can be used as an estimate when equilibrium is achieved.

8.2 Excluded Volume

If only the harmonic potential is used, two particles can be at the same place. If the Lennard-Jones-Potential is included as well, two particles cannot occupy the same place and the equation of motion becomes

$$m_i \ddot{\vec{x}}_i = k(\vec{x}_{i+1} - 2\vec{x}_i + \vec{x}_{i-1}) + 48 \left(r_{ij}^{-14} - \frac{1}{2} r_{ij}^{-8} \right) \vec{r}_{ij} - \gamma_i m_i \dot{\vec{x}}_i + \vec{R}_i \quad (40)$$

$\langle R_{ee}^2 \rangle$ is now not longer proportional to $N - 1$, but

$$\langle R_{ee}^2 \rangle \propto (N - 1)^{2\nu} \quad (41)$$

where ν is about $3/5$.

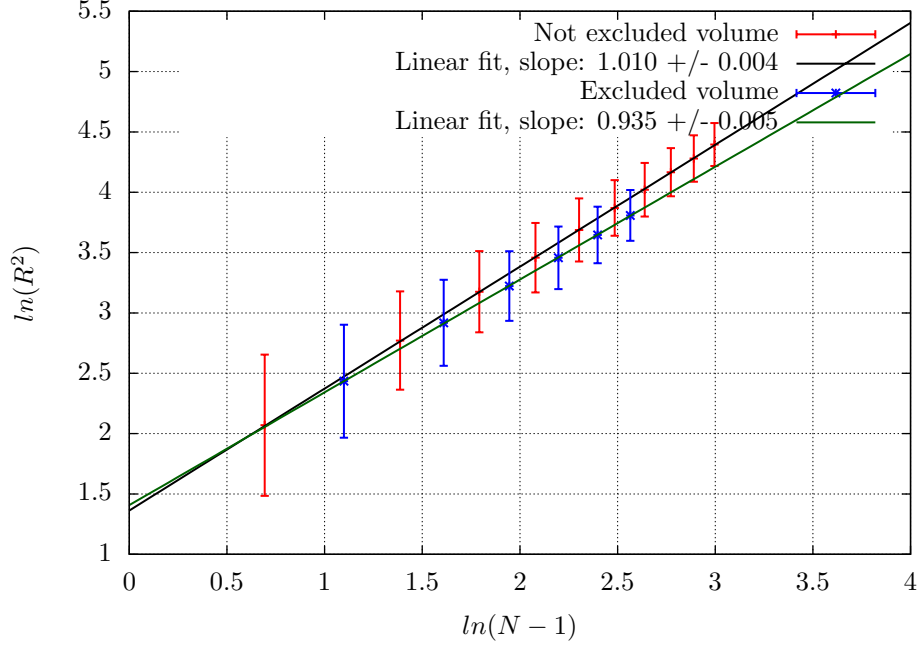


Figure 10: Log-Log plot for the end-to-end distance as a function of the amount of springs without excluded volume (red) and with excluded volume (blue) ($\gamma = 1$)

References

- [1] Giovanni Ciccotti Eric Vanden-Eijnden. Second-order integrators for langevin equations with holonomic constraints. *Chemical Physics Letters*, 429, 2006.
- [2] S. F. Edwards M. Doi. *The Theory of Polymer Dynamics*. Clarendon Press, 1988.
- [3] OpenMP API for parallel programming, version 3.1.
- [4] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge, 2004.
- [5] Prince E. Rouse. A theory of the linear viscoelastic properties of dilute solutions of coiling polymers. *The Journal of Chemical Physics*, 1953.
- [6] Iwao Teraoka. *Polymer Solutions: An Introduction to Physical Properties*. John Wiley & Sons, 2002.
- [7] Loup Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical Review*, 159, 1967.