University of Antwerp

Chemo-informatics and computational drug design

Prof. Dr. Hans De Winter

University of Antwerp

Campus Drie Eiken, Building A

Universiteitsplein 1, 2610 Wilrijk, Belgium

Gefinancierd door de Europese Unie
NextGenerationEU

Chapter 7. Molecular dynamics

1. Newton's 2nd law of motion

Molecular dynamics (MD) is a method to simulate the physical movement of atoms and molecules using a series of mathematical calculations. Commonly, this is done by numerically solving Newton's equations of motion for the system of interacting atoms and bonds, in which the forces and the potential energies between these atoms are calculated from the standard molecular mechanics force fields (Chapter 6).

A typical MD simulation consists of an iterative process which involves a number of discrete steps:

1. Given a set of atomic positions and a defined molecular mechanics force field, the first step consists of the calculation of the forces on each atom.
2. In the second step, the acceleration on each atom can be deduced from the force that acts on each atom.
3. In the third step, each atom is repositioned according the acceleration that is acting on it. For this purpose, small time steps are required to ensure numerical stability.
4. If enough steps have been performed, the simulation will step. If not, the simulation will continue in step 1.

These steps are described in more detail in the following sections.

1.1. Step 1: force calculation

The mathematical principles behind a MD simulation are quite simple, and involve the calculation of the forces between the atoms from the potential energy function that defines the system:

$$\vec{F} = -\nabla V(\vec{r})$$

with $\vec{F}$ being the forces on the system and $V(\vec{r})$ the potential energy of the system as function of the atomic positions $\vec{r}$.

Using the potential energy function as shown on page 86, calculating the corresponding first derivatives to obtain the forces is actually quite simple:

- Bond forces: $F_b = -2k(b - b_0)$
- Angle forces: $F_a = -2k(\theta - \theta_0)$
- Torsion angle forces: $F_d = kn\sin(n\phi - \delta)$
- Electrostatic forces: $F_e = q_i q_j / Dkr_{ij}^2$
- Van der Waals forces: $F_{vdw} = -12\varepsilon_{ij}R_{min,ij}^6\left(r_{ij}^6 - R_{min,ij}^6\right)/r_{ij}^{13}$

Plots of the potentials and corresponding forces are shown in Figure 60.

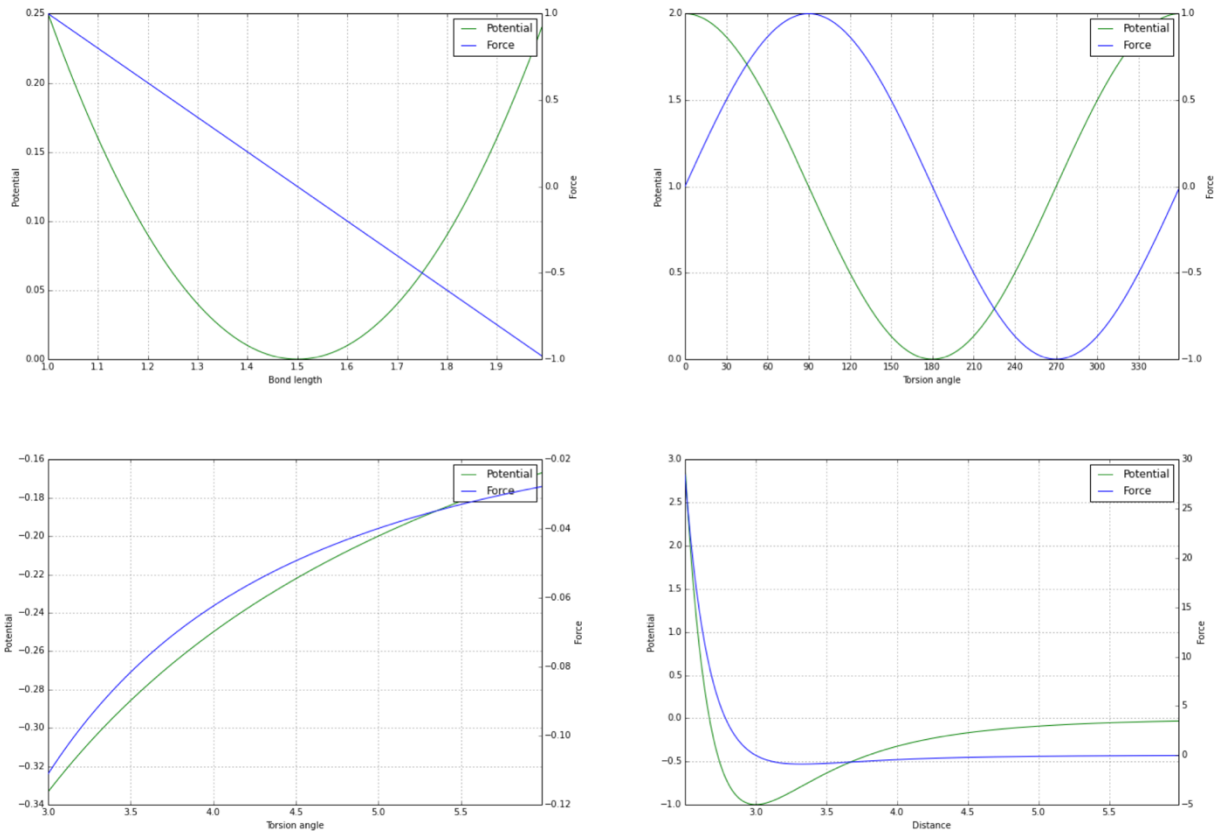


Figure 60. Illustration of the different force field potentials and the corresponding forces that are derived thereof. A) Upper left: bond potential and force with b_0 = 1.5 and k = 1. B) Upper right: torsion angle potential and force with n = 1, k = 1, δ = 0. C) Lower left: electrostatic potential and force with q_i = +1 and q_j = -1, D =1, k = 1. D) Lower right: Van der Waals potential and force with ε_{ij} = 1, $R_{min,ij}$ = 3.

1.2. Step 2: atom acceleration

Once the force acting on each atom is known, it is possible to calculate the acceleration for each atom from Newton's 2nd law of motion:

$$\vec{F} = m\vec{a}$$

Therefore:

$$\vec{a}_i = \frac{\vec{F}_i}{m_i}$$

with *i* being the index of the particular atom *i*.

1.3. Step 3: new atom positions

Several methods have been developed to calculate new atoms positions from the acceleration on each atom. The most important ones are the Verlet and leap-frog algorithms.

Verlet algorithm

The Verlet algorithm is based on the Taylor series expansion:

$$f(x_0 + x) = f(x_0) + \frac{x^1}{1!}f'(x_0) + \frac{x^2}{2!}f''(x_0) + \frac{x^3}{3!}f'''(x_0) + \cdots + \frac{x^n}{n!}f^{(n)}(x_0)$$

Taylor series are often truncated after the term involving the second derivative. Applied to our MD simulation, we can rewrite the new set of coordinates as $r(t + \delta t)$, and expand this as a Taylor series with *t* as x_0 and δt as *x* (or -δt as *x*):

$$r(t+\delta t) = r(t) + \frac{\delta t^1}{1} r'(t) + \frac{\delta t^2}{2} r''(t) + \cdots$$

$$r(t-\delta t) = r(t) - \frac{\delta t^1}{1} r'(t) + \frac{\delta t^2}{2} r''(t) + \cdots$$

which yields the following considering that $r'(t) = v(t)$ and $r''(t) = a(t)$:

$$r(t+\delta t) = r(t) + \delta t\, v(t) + \frac{1}{2}\delta t^2\, a(t)$$

$$r(t-\delta t) = r(t) - \delta t\, v(t) + \frac{1}{2}\delta t^2\, a(t)$$

with *v(t)* being the velocities on the atoms and *a(t)* the accelerations. Adding these two equations gives:

$$r(t+\delta t) + r(t-\delta t) = r(t) + \delta t\, v(t) + \frac{1}{2}\delta t^2\, a(t) + r(t) - \delta t\, v(t) + \frac{1}{2}\delta t^2\, a(t)$$

which can be rewritten as:

$$r(t+\delta t) = 2r(t) + \delta t^2\, a(t) - r(t-\delta t)$$

Implementation of the Verlet algorithm is straightforward. Calculation of a MD trajectory [*r(t+δt)*] requires only a timestep δt, the positions at timestep -δt [*r(t-δt)*], and the accelerations *a(t)* which in turn are derived from the forces.

Leap-frog algorithm

The leap-frog algorithm uses the following relationships:

$$r(t+\delta t) = r(t) + \delta t\, v\left(t + \frac{1}{2}\delta t\right)$$

$$v\left(t + \frac{1}{2}\delta t\right) = v\left(t - \frac{1}{2}\delta t\right) + \delta t\, a(t)$$

The velocities $v\left(t + \frac{1}{2}\delta t\right)$ are first calculated from the velocities at $t - \frac{1}{2}\delta t$ and the accelerations at time *t*. From this, the coordinates $r(t+\delta t)$ are then derived from the velocities at $t - \frac{1}{2}\delta t$ with the positions at the current frame. Hence, the velocities thus 'leap-frog' over the positions, which explains the name of the algorithm (Figure 61).

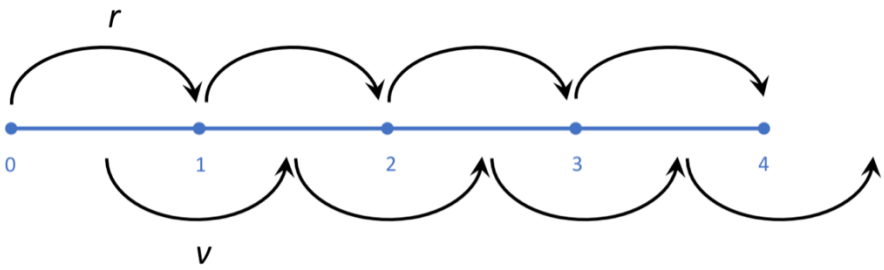


Figure 61. Sketch showing the structure of the leap-frog method. Once the algorithm has started with r(0) and $v(\frac{1}{2}\delta t)$, the algorithms continues by r and v leap-frogging over each other.

Choosing a timestep δt

An important parameter in the equations of motion of each MD simulation is the timestep δt. Using a large timestep would speed up the simulations significantly, but this would lead to algorithmic instabilities as the calculated atom relocations would be too large (Figure 62).

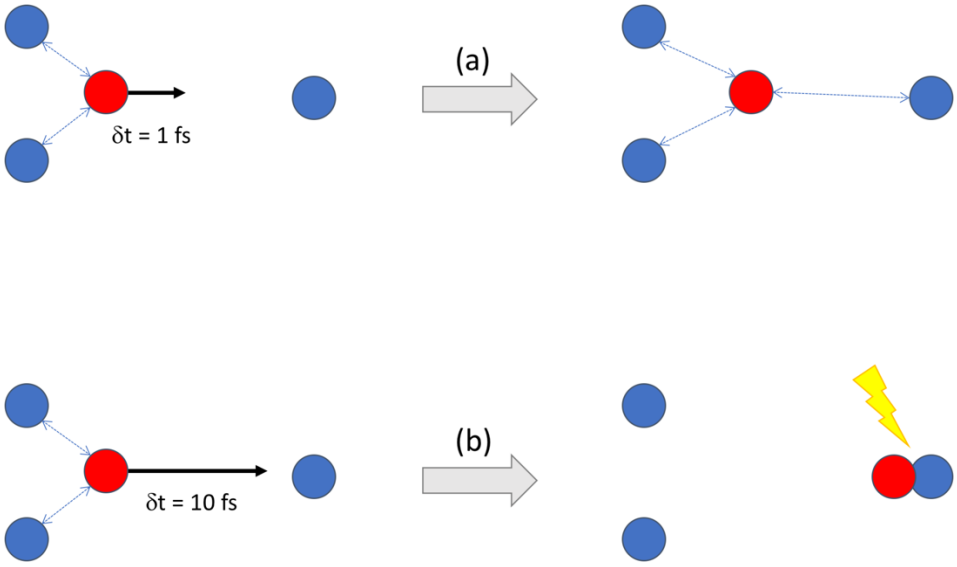


Figure 62. Illustration of the importance of using a suitable timestep. Blue spheres are atoms which are fixed in space, while the red sphere is the atom that is able to relocate due to the forces (blue dotted arrows) imposed by the surrounding blue atoms. (a) Relocation of the red atoms using a timestep of 1 fs. This yields a new situation in which the red atom is moved somewhat closer to the 3rd blue atom, but remains at a distance which is still far enough to avoid sterical clashes. (b) With a timestep of 10 fs, the red atom is relocated too far which results in a sterical clash with the 3rd blue atom.

It has been shown that the most optimal timestep should be 10 times smaller than the period of the highest vibrational frequencies of a molecule. In practice, a timestep of 1 fs is often used, and this number can be doubled to 2 fs in cases where the highest frequencies, *in casu* the C-H bond stretching vibrations, can be freezed out by constraining these bonds to their equilibrium values. Examples of such constraints are the SHAKE [7], LINCS [8] and RATTLE [9] algorithms.

2. Running a simulation

2.1. Setting up the protein and its environment

Obtaining protein coordinates

The basic ingredient to start a molecular dynamics simulations is a structure coordinate file of the molecule one wants to study. In case this is a protein or oligonucleotide structure, these structure coordinates can be downloaded from the Protein Databank (PDB, http://www.rcsb.org/pdb/home/home.do). The PDB contains thousands of high resolution protein structures that are derived by X-ray/neutron scattering and NMR methods. Once the PDB is available and downloaded, it is necessary to pre-format it depending on the presence of ligands or water molecules in the structure. It occurs that ligand coordinates are present in the PDB file and the MD software doesn't recognize the ligand. In this case, ligand chemistry needs to be explicitly defined by extracting the coordinates of the ligand and separated from main PDB file. A separated topology is constructed manually for the ligand and the information is added in the main topology file. It is also advised to remove external water molecules that are present in the PDB file.

Cleaning the protein structure

1) **Missing loops.** Not all the structures of the PDB are complete; some structures are missing some loops in the protein due to extensive motions of these loops. This leads to smearing out the electron density of these regions and therefore lack of visibility in the corresponding electron density maps. Several websites are online that provide tools to model these missing loops:

- ModLoop (https://modbase.compbio.ucsf.edu/modloop/)
- RAPPER (http://mordred.bioc.cam.ac.uk/~rapper/ca_trace.php)

- FALC-Loop (http://falc-loop.seoklab.org/)
- GalaxyWEB (http://galaxy.seoklab.org)
- ArchPRED (http://manaslu.fiserlab.org/loopred/)
- SuperLooper (http://bioinf-applied.charite.de/superlooper/)
- RCD+ (http://rcd.chaconlab.org)
- ROSIE (http://rosie.rosettacommons.org)

2) **Protonation state.** Next to having all missing residues fixed, another thing that should be taken care of is the protonation state of the acidic and basic protein residues, such as histidine (HIS), aspartic acid (ASP), glutamic acid (GLU), lysine (LYS), arginine (ARG), cysteine (CYS) and tyrosine (TYR). Depending on the local pH, these residues can reside in their acidic (deprotonated) or in their basic (protonated) form (Figure 63). The pK_a's of these seven residue are given in Table 12.

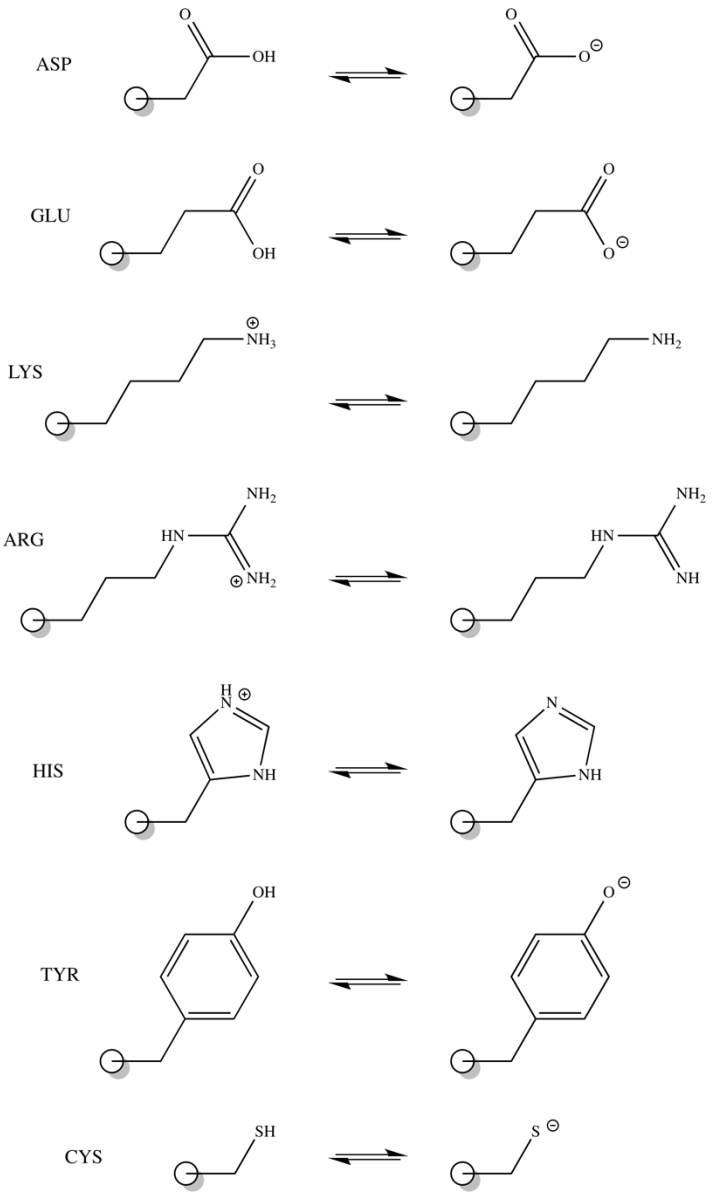


Figure 63. The seven residues which can occur either in their acidic form (left side) or basic form (right side).

Table 12. pK_a values of the acidic form of the seven residues which may occur in either acidic or basic form. Depending on the pH of the local environment (which may differ significantly from the pH of the global environment), residues may be protonated, deprotonated, or reside in both forms. The prevailing form at pH 7.4 can be calculated from the Henderson-Hasselbalch equation $pH = pK_a + log([base]/[acid])$.

Residue	pK_a	Prevailing form at pH 7.4
Aspartic acid	3.65	100% basic (anion)
Glutamic acid	4.25	100% basic (anion)
Lysine	10.53	100% acidic (cation)
Arginine	12.48	100% acidic (cation)
Histidine	6.00	96% basic (neutral), 4% acidic (cation)
Tyrosine	10.07	100% acidic (neutral)
Cysteine	8.33	10% basic (anion), 90% acidic (neutral)

It is important to calculate the pK_a of each ionisable residue in its protein environment, since the local environment of each residue may alter its pK_a significantly. There exist a number of online tools for the calculation of these pK_a's:

- Karlsberg+ (http://agknapp.chemie.fu-berlin.de/karlsberg/)
- H++ (http://biophysics.cs.vt.edu)
- PDB2PQR (http://nbcr-222.ucsd.edu/pdb2pqr_2.0.0/)

3) **Asparagine, glutamine and histidine flips**. Explicit hydrogens are needed for many of the MD force fields to function correctly, and structures determined by crystallography almost never include these hydrogens. As a consequence, hydrogen atoms need to be added prior to starting off a MD simulation, and this process is often automatized by most MD programs. However, a common problem is that the sidechain ends of asparagine (ASN), glutamine (GLN) and HIS residues are easily fitted 180° backwards, since the electron density alone cannot usually distinguish the correct choice of orientation. There are online tools available that can automatically diagnose and correct these types of systematic errors by considering all-atom steric overlaps as well as hydrogen bonding within each local network. The use of such tool is strongly recommended prior to adding any hydrogens:

- MolProbity (http://molprobity.biochem.duke.edu)

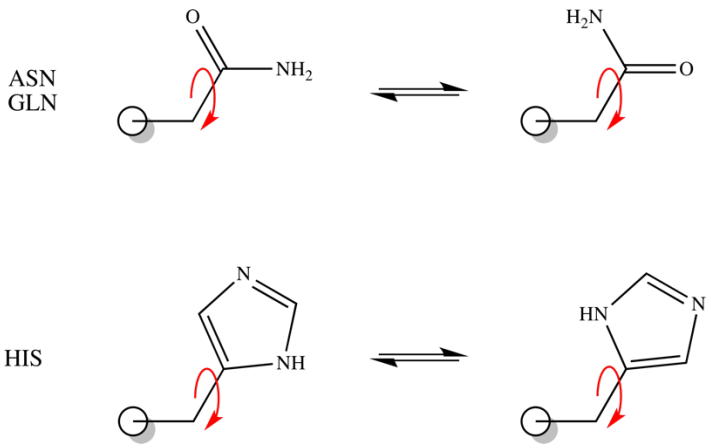


Figure 64. Sidechain flips of ASN, GLN and HIS illustrated. The evidence of such flips can only be deduced from inspection of the local environment around each of these residues.

Setup the box for simulations

1) **Choosing the periodic boundary box**. When simulating biomolecules in a water environment, periodic boundary conditions (PBC) are used to avoid artefacts resulting from boundary effects caused by finite size, and make the system more like an infinite one. During the simulation, only the properties of the original simulation box need to

be recorded and propagated. The minimum-image convention is a common form of PBC particle bookkeeping in which each individual particle in the simulation interacts with the closest image of the remaining particles in the system (Figure 65).

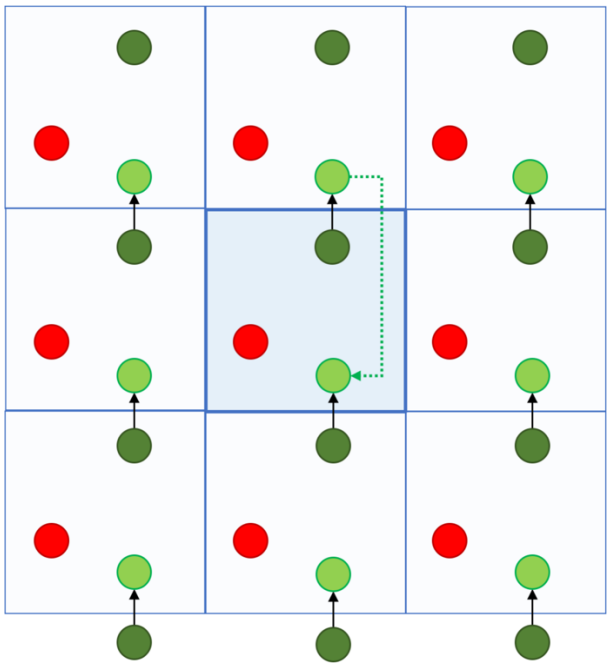


Figure 65. Periodic boundary condition in two dimensions. The blue square denotes the PBC box which contains three particles (red, dark green and light green). In computer simulations, this blue PBC box is the original simulation box, and the others are copies which are called images. A particle which has passed through one face of the simulation box should re-enter through the opposite face, as shown by the dotted arrow.

PBC requires the unit cell to be a shape that will tile perfectly into a three-dimensional crystal. Thus, a spherical or elliptical droplet cannot be used. A cube or rectangular prism is the most intuitive and common choice, but can be computationally expensive due to unnecessary amounts of solvent molecules in the corners, distant from the central macromolecules. An alternative that requires less volume is the truncated octahedron (Figure 66).

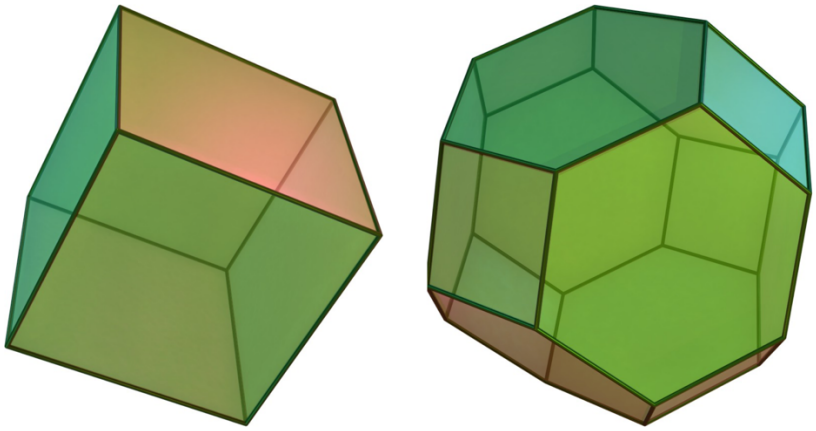
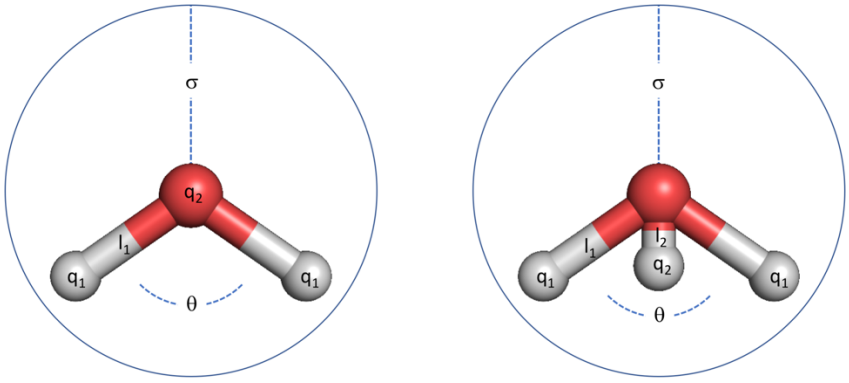


Figure 66. Two commonly used periodic box unit cells. Left: rectangular box. Right: truncated octahedron. This latter shape resembles more a spherical shape and is therefore often more 'economical' in terms of the number of solvent molecules that are needed to fill up the entire volume.

2) **Adding solvent**. MD simulations of biomolecular systems are performed in an environment containing water, which implies that a suitable water model has to be selected. There are a number of water models available (Figure 67), and it is important that the selected water model force field parameters are compatible with the force field parameters of the biomolecular system. In most cases, the TIP3P water model is usually fine.



Model a Model b

Figure 67. Parameters for some water models. 1) ***TIP3P*** *model: topology a, σ = 3.15 Å, θ = 104.5°, q_1 = +0.42, q_2 = -0.84, l_1 = 0.96 Å; 2)* ***SPC*** *model: topology a, σ = 3.17 Å, θ = 109.5°, q_1 = +0.41, q_2 = -0.82, l_1 = 1.00 Å; 3)* ***TIP4P*** *model: topology b, σ = 3.15 Å, θ = 104.5°, q_1 = +0.52, q_2 = -1.04, l_1 = 0.96 Å, l_2 = 0.15 Å.*

3) **Charge-neutralising the system**. The addition of ions is often needed to charge-neutralise the system (total charge of the entire box should be zero) and to reproduce an experimental environment in terms of salt concentration (for example 150 mM NaCl). Most MD software package contain tools to achieve this in a rather automated fashion.

2.2. Maintaining temperature and pressure: ensembles

Canonical ensemble: NVT

In a canonical ensemble, temperature, box volume and the number of particles is kept constant. The box volume is kept constant by keeping the unit cell dimensions unaltered. Temperature is maintained through one of the multiple thermostats that are available. The instantaneous value of the temperature of an unconstrained system is related to the kinetic energy via the particles' linear momenta:

$$T = \frac{2}{Nk_b}\sum_{i=1}^{N}\frac{|\vec{\boldsymbol{p}}_i|^2}{2m_i}$$

where N is the number of atoms, m_i the atomic mass of atom i, k_b the Boltzmann constant, and $\vec{\boldsymbol{p}}_i$ the translational momentum of atom i.

An obvious way to alter the temperature of the system is **velocity scaling**. If the temperature at time t is $T(t)$ and the velocities are multiplied by a factor λ, then the associated temperature change ΔT can be calculated as:

$$\Delta T = \frac{2}{Nk_b}\sum_{i=1}^{N}\frac{(m_i\lambda\vec{v}_i)^2}{2m_i} - \frac{2}{Nk_b}\sum_{i=1}^{N}\frac{(m_i\vec{v}_i)^2}{2m_i}$$

$$\Delta T = \frac{2\lambda^2}{Nk_b}\sum_{i=1}^{N}\frac{(m_i\vec{v}_i)^2}{2m_i} - \frac{2}{Nk_b}\sum_{i=1}^{N}\frac{(m_i\vec{v}_i)^2}{2m_i}$$

$$\Delta T = \lambda^2 T(t) - T(t) \quad \text{hence} \quad \Delta T = (\lambda^2 - 1)T(t)$$

$$T_0 - T(t) = \lambda^2 T(t) - T(t)$$

$$T_0 = \lambda^2 T(t)$$

$$\lambda = \sqrt{\frac{T_0}{T(t)}}$$

The simplest way to control the temperature is thus to multiply the velocities at each time step by the factor $\lambda = \Delta T_0/T(t)$, where $T(t)$ is the current temperature as calculated from the kinetic energy and T_0 being the desired temperature.

A weaker formulation of this approach is the **Berendsen thermostat**. To maintain the temperature the system is coupled to an external heat bath with fixed temperature T_0. The velocities are scaled at each timestep δt, such that the rate of change of temperature is proportional to the difference in temperature:

$$\frac{\delta T(t)}{\delta t} = \frac{1}{\tau}\big(T_0 - T(t)\big)$$

where τ is the coupling parameter which determines how tightly the bath and the system are coupled together. This method gives an exponential decay of the system towards the desired temperature. The change in temperature between successive time steps is:

$$\Delta T = \frac{\delta t}{\tau}\big(T_0 - T(t)\big)$$

and after substituting $(\lambda^2\text{-}1)T(t)$ for ΔT, the scaling factor for the velocities can be deduced :

$$\lambda^2 T(t) - T(t) = \frac{\delta t\big(T_0 - T(t)\big)}{\tau}$$

$$\lambda^2 T(t) = \frac{\delta t\big(T_0 - T(t)\big)}{\tau} + T(t)$$

$$\lambda^2 = \frac{\delta t\big(T_0 - T(t)\big)}{\tau T(t)} + 1$$

$$\lambda^2 = \frac{\delta t}{\tau}\left(\frac{T_0}{T(t)} - 1\right) + 1$$

If τ is large, then the coupling will be weak. If τ is small, then the coupling will be strong. If τ is chosen the same as the timestep δt, the Berendsen thermostat is nothing else than the simple velocity scaling. Values of $\tau \simeq$ 0.1-0.4 ps are typically used in MD simulations of condensed-phase systems, giving $\delta t/\tau \simeq$ 0.01-0.0025.

Velocity scaling artificially prolongs any temperature differences among the components of the system, which can lead to 'hot solvent, cold solute' phenomena, in which the temperature of the solvent is warmer than that of the solute, even though the overall temperature of the system may be at the desired value. One solution is to couple a particular set of particles (for example the solute as one set, and the solvent as a second set) to a separate thermostat. In addition, both the velocity scaling method as well as the Berendsen thermostat do not give rigorous canonical averages. Two methods that do generate rigorous canonical ensembles are the stochastic collisions methods and the extended system method.

In the **stochastic collisions** methods, a particle is randomly chosen at intervals and its velocity is reassigned by random selection from the Maxwell-Boltzmann distribution at the desired temperature. The method is also known as the **Anderson** method. It can be viewed as a method in which the system is in contact with a heat bath that randomly emits 'heat particles' which collide with the atoms in the system.

The **extended system** method was developed by Nose and Hoover, and is therefore also known as the **Nose-Hoover** temperature coupling bath. The mathematical derivatisation will not be covered here.

Isothermal-isobaric ensemble: NPT

In an isothermal-isobaric ensemble, temperature, pressure and the number of particles is kept constant. Many experimental measurements are made under conditions of constant temperature and pressure, and so simulations in the isothermal-isobaric ensemble are most directly relevant to experimental data. Temperature is maintained through one of the multiple thermostats that are available and which are explained in the previous section.

The pressure fluctuates more than other quantities such as the temperature. This can be explained by the fact that the pressure is related to the forces and positions of the particles:

$$p = \frac{k_b T N}{V} + \frac{1}{3V}\overline{\sum_{i<j} \overrightarrow{f_{ij}}\overrightarrow{r_{ij}}}$$

where **p** is the pressure, *T* is the temperature, **V** is the volume and k_b is the Boltzmann constant. The overline is an average, which is a time average in molecular dynamics, $\overrightarrow{f_{ij}}$ is the force on particle *i* exerted by particle *j*, and $\overrightarrow{r_{ij}}$ is the vector going from *i* to *j*: $\overrightarrow{r_{ij}} = \vec{r_j} - \vec{r_j}$. This quantity changes more quickly with the positions than the temperature, hence the greater fluctuations in pressure.

A macroscopic systems maintains constant pressure by changing its volume; hence a simulation in isothermal-isobaric conditions also maintains constant pressure by changing its volume of the PBC unit cell. The amount of volume fluctuation is related to the isothermal compressibility **κ**:

$$\kappa = -\frac{1}{V}\left(\frac{\delta V}{\delta P}\right)_T$$

Larger values of **κ** indicate more easily compressible substances, with larger volume fluctuations that occur at a given pressure than in less compressible substances.

Volume changes in an isobaric ensemble are achieved by changing the size of the unit cell in one, two or all three dimensions (normally in all directions). The relation between the isothermal compressibility **κ** and the box volume is given by:

$$\kappa = \frac{1}{k_b T}\frac{\langle V^2\rangle - \langle V\rangle^2}{\langle V^2\rangle}$$

with $\langle V^2\rangle$ being the time average of the squared volumes, and $\langle V\rangle^2$ the square of the time average of the volumes. The fraction $\frac{\langle V^2\rangle - \langle V\rangle^2}{\langle V^2\rangle}$ corresponds to the mean square volume displacement. Fluctuations in pressure and temperature for a typical MD simulation of a protein in a water box are given in Figure 68.

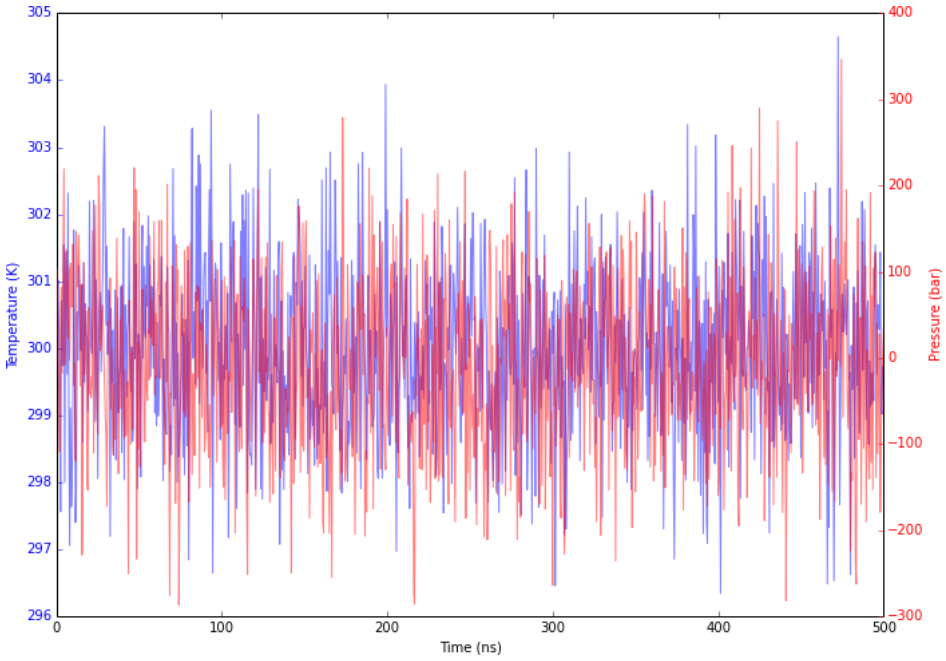


Figure 68. Temperature and pressure fluctuations in a MD simulation of a protein in a water box of dimensions 81.8 x 86.6 x 107.9 Å³. Average simulation temperature is 300.0 K with a standard deviation of 1.2 K, and average simulation pressure is -9.0 bar with a standard deviation of 100.6 bar.

Many of the methods to control pressure are similar to the ones to control temperature. Thus, pressure can be maintained at a constant value by scaling the volume. An alternative is to use a pressure bath in analogy to the temperature bath as found in the Berendsen thermostat. The rate of pressure change is given by:

$$\frac{dP(t)}{dt} = \frac{1}{\tau_p}\Big(P_{ref} - P(t)\Big)$$

with τ_p being the coupling constant, P_{ref} the reference pressure of the bath, and $P(t)$ the actual pressure at time t. To regulate pressure, the volume of the simulation unit cell is scaled by a factor λ, which is achieved by scaling the atomic coordinates towards the centre of the box by a factor $\lambda^{1/3}$. This factor λ can be derived according:

$$\lambda = 1 - \kappa\frac{\delta t}{\tau_p}\big(P - P_{ref}\big)$$

and the new centre of mass coordinates are given by:

$$\vec{r}_i' = \lambda^{1/3}\vec{r}_i$$

2.3. Short- and long-range interactions

Short-range

The most time-consuming part of a MD simulation is the calculation of the non-bonded energies and forces. The number of bond potential, angle potentials and torsion angle potentials in a force field model are all proportional to the number of atoms (order N), but the number of non-bonded terms that need to be evaluated at each step relates to the square of the number of atoms and is thus of order N^2.

Although in principle the non-bonded interactions should be calculated between every pair of atoms in the system, fortunately this is not always justified since the 12-6 Lennard-Jones potential falls off very quickly with distance. At a distance of 3 times $R_{min,ij}$, the van der Waals interaction has fallen to a value which is less than 1% of the corresponding value at a distance of $R_{min,ij}$ (Figure 69).

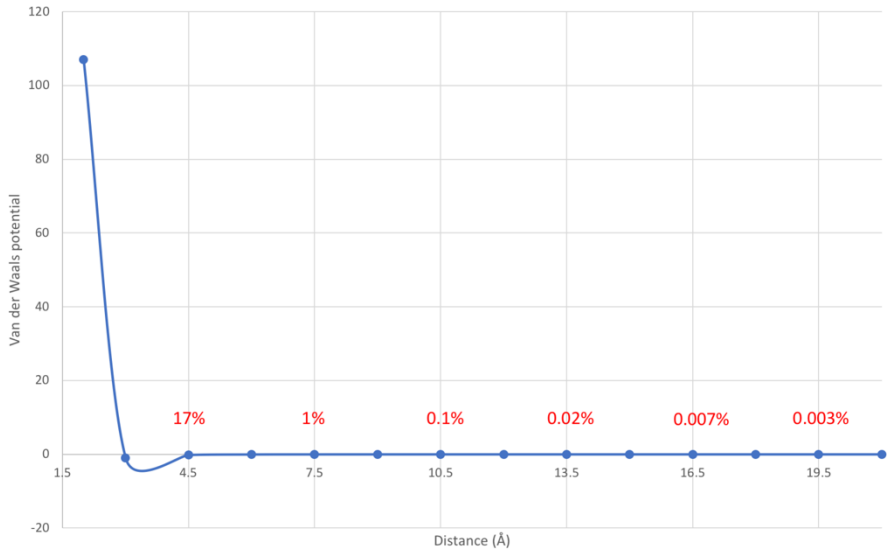


Figure 69. Fall-off of the non-bonded van der Waals interaction between a pair of atoms as a function of distance. $R_{min,ij}$ is set to 3 Å. Values in red is the relative reduction in van der Waals potential compared to the value at 3 Å. At a distance of 9 Å, the van der Waals interaction has fallen to a value which is less than 1% of the corresponding value at a distance of 3 Å.

A common approach to deal with the non-bonded interactions is to use a non-bonded cut-off and to apply the minimum image convention. When a cut-off is applied, the interactions between all pairs of atoms that are further apart than the cut-off value are set to zero. In this minimum image convention, each atom interacts with at most one image of every other atom in the system. When periodic boundary conditions are being used, the cut-off should be less than half the length of the cell, because otherwise a particle could interact with its own image and thus interact with same atom twice (Figure 70). Depending on the size of the biomolecule and the unit cell, a cut-off value of 10 Å or larger is recommend.

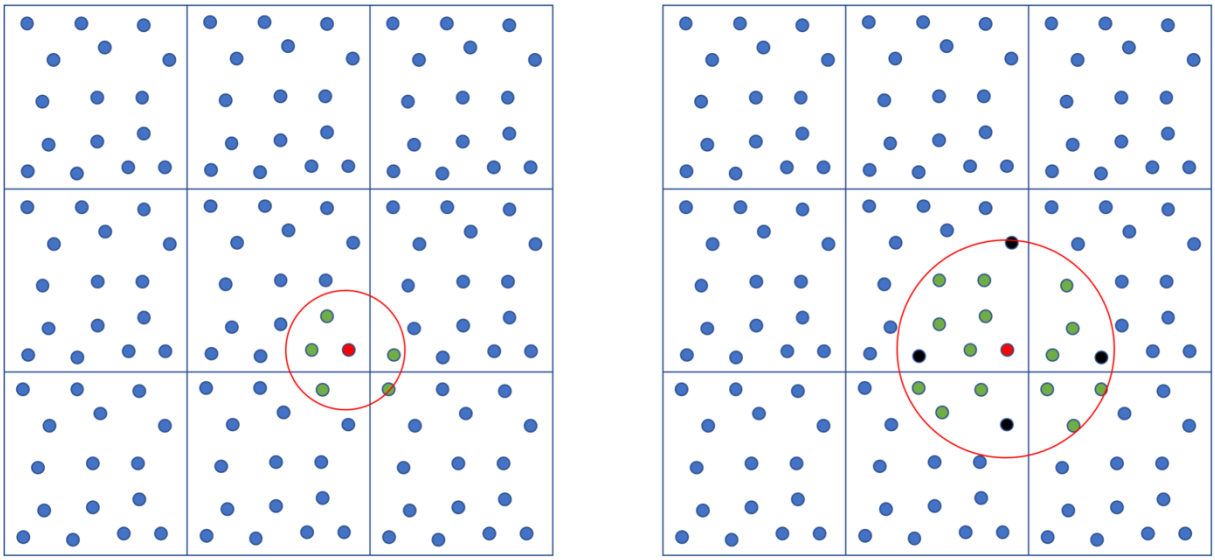


Figure 70. The spherical cutoff and the minimum image convention. Left: spherical cutoff (red circle) that is smaller than half of the unit cell dimension. The red atom is able to interact with all atoms that are positioned within the cutoff radius (shown in green). Right: spherical cutoff with a radius larger than half of the initial cell dimension. This results in interactions with multiple images of the same atoms, as shown in black spheres.

The cut-off distance defines the radius of the sphere containing all atoms for which the interaction with the originating atom is included. Therefore, when implemented as such, it is a hard limit that imposes a discontinuity in the potential energy and forces nearby the cut-off value, and hence may lead to truncation errors and issues with the total energy conservation. To circumvent this, a switching function may be used near the cut-off distance that smooths the van der Waals interaction to zero (Figure 71).

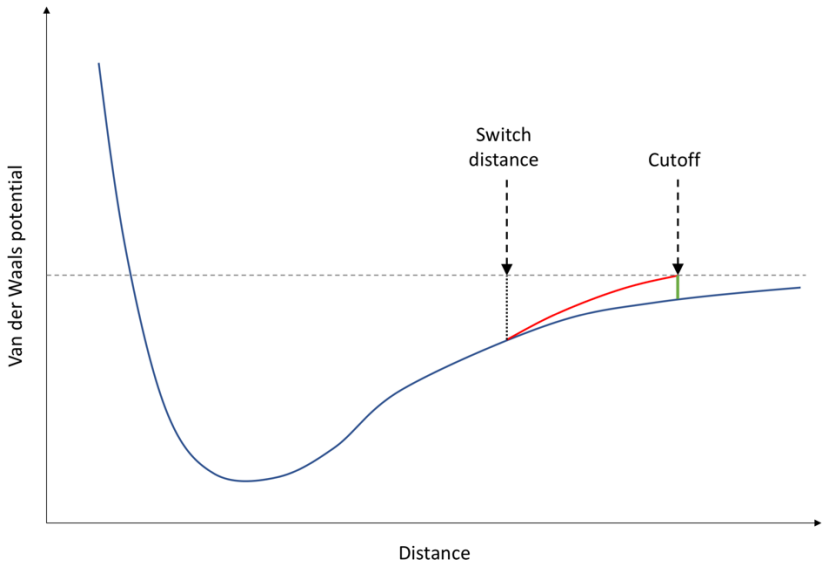


Figure 71. Van der Waals potential (red) and without (green) switching function.

Long-range

The electrostatic potential decays as r^{-1} and can therefore not be truncated at the same cut-off distance that is applicable for the van der Waals interactions. It is important to properly model these long-range non-bonded interactions, and a number of methods have been developed to account for this:

- **Ewald summation** takes advantage of ideas from harmonic analysis to reduce the complexity of the non-bonded interactions the order of $O(N^{3/2})$, which is often still impractical in the case of large systems;
- The **Particle-Mesh Ewald (PME)** method uses Fast Fourier Transform to bring the complexity down to the order of $O(N \log N)$, and this algorithm has been a staple of molecular simulation since its appearance;
- The **Smooth Particle-Mesh Ewald (SPME)** improves on PME by giving a sufficiently smooth energy function whose derivative can be obtained analytically, which results in more realistic simulations due to improved energy conservation.

2.4. Equilibration

In MD simulations, atoms of the macromolecules and of the surrounding solvent have to undergo a relaxation period that usually lasts for tens or hundreds of picoseconds before the system reaches a stationary state. The initial nonstationary segment of the simulated trajectory is typically discarded in the calculation of equilibrium properties. This stage of the MD simulation is called the equilibration stage.

Some equilibration protocols involve gradually increasing the temperature in a stepwise fashion while other more aggressive approaches simply use a linear temperature gradient and heat the system up to the desired temperature. Therefore, equilibration protocols exist in multiple flavours and it is not always straightforward to indicate the most optimal one, although that a gentle equilibration is often better. In Table 13, an example protocol is shown as a matter of illustration.

Table 13. Illustration of a typical equilibration protocol. Position restraints are often placed on the solute heavy atoms to prevent unphysical movements during the initial simulation stages.

Stage	Time (ps)	Temperature (K)	Restraints (kcal/mol/Å²)	Remarks
Minimization	1,000-10,000 steps	0	10	In this stage, steric clashes and close contacts are removed by a steepest descent minimization phase that consists of 1,000-10,000 steps.
Heating	5	0 → 300	10	Atoms are assigned an initial velocity according the desired temperature. To prevent blow-up of the system, temperature (kinetic energy) is added to the system only gradually, for example in steps of 5 K for a period of 5 ps each, resulting in a heating phase that lasts 300 ps (5 ps * 300 ps / 5) in total.
Relax restraints	450	300	10 → 0	The restraints on the heavy atoms are slowly relaxed, going from 10 kcal/mol/Å² to none. This relaxation is achieved over a total time of 450 ps.
Equilibrate	500	300	0	The last stage of the equilibration consists of a normal run at 300 K and without restraints, lasting for at least 500 ps.

3. Analysis

3.1. Production run

The optimal length of a MD production run is hard to define and depends on the size of the system and the timescales of the intrinsic movements one wants to investigate. Some protein folding events may occur on a 1-10 ns timescale, while other folding events may last at least 100-1,000 ns. Also, in order to decide on the optimal length of a MD simulation to study the binding or unbinding event of protein-ligand complex, experimental binding kinetics data of the event might help on deciding.

The output of a typical MD simulation consists of a trajectory file, which is a file containing the time-evolving coordinates of the MD system. Trajectories are sequential snapshots of the simulated molecular system which represents atomic coordinates at specific time periods. Not every timestep is recorded to a trajectory file, but rather at a user-defined interval which depends on the total length of the simulation and the amount of hard disk space that is available. In a typical run, the coordinates are saved to the trajectory file in intervals of 1-100 ps. Each coordinate set in the trajectory file is called a frame, hence the number of frames correspond to the number of coordinate sets. For example, consider a production run of 1,000 ns (which corresponds to 1 μs) with the coordinates saved every 100 ps, this will result in a trajectory file containing 1,000,000 ps / 100 ps = 10,000 coordinate sets or frames.

3.2. Common analysis methods

Visualization

The visualization of the movements of the system is extremely useful, and many software visualization tools have been developed to achieve this goal. Two of these are open-source and provide tools to generate high-quality movies of these movements:

- VMD (Visual Molecular Dynamics) is a product of the Group of Theoretical Biophysics from the University of Illinois (http://www.ks.uiuc.edu/Research/vmd/). It was developed specially for visualization and analysis of such biological systems as proteins, nucleic acids, and molecular systems on the basis of lipids (for example, components of cell membranes). The program compatible with the PDB format and allows user to use various methods of visualization and colouring molecules. VMD is suitable for animation and analysis of phase trajectories obtained from MD simulation.
- PyMol (https://github.com/schrodinger/pymol-open-source) is a molecular visualization system created by Warren DeLano. It is user-sponsored, open-source software, released under the Python License.

PyMOL can produce high-quality 3D images of small molecules and biological macromolecules, such as proteins. The *Py* part of the software's name refers to that it extends, and is extensible by, the programming language Python.

Root-mean square deviation (RMSD)

For each timepoint, the RMSD measures the deviation of the molecule relative to a reference set of coordinates, which is typically the first frame of the trajectory. In order to calculate the deviation, the atom coordinates of all frames are first aligned onto the corresponding coordinates of the first frame, and then the RMSD at each timepoint is calculated:

$$RMSD(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\vec{r}_i(t) - \overrightarrow{r_{i,ref}}\right)^2}$$

in which *N* being the number of atoms that are used in the RMSD calculation at timepoint *t* (this could be all atoms, the heavy atoms, backbone atoms, the Cα atoms, or any other subset of atoms), $\vec{r}_i(t)$ the coordinate of atom *i* at timepoint *t*, and $\overrightarrow{r_{i,ref}}$ the coordinate of the corresponding atom in the reference set.

The RMSD calculation is useful for judging overall motions and equilibration progress, as the RMSD typically first increases and then equilibrates to a steady-state (Figure 72). The *x*-coordinate (abscise) of a RMSD plot is typically the simulation time.

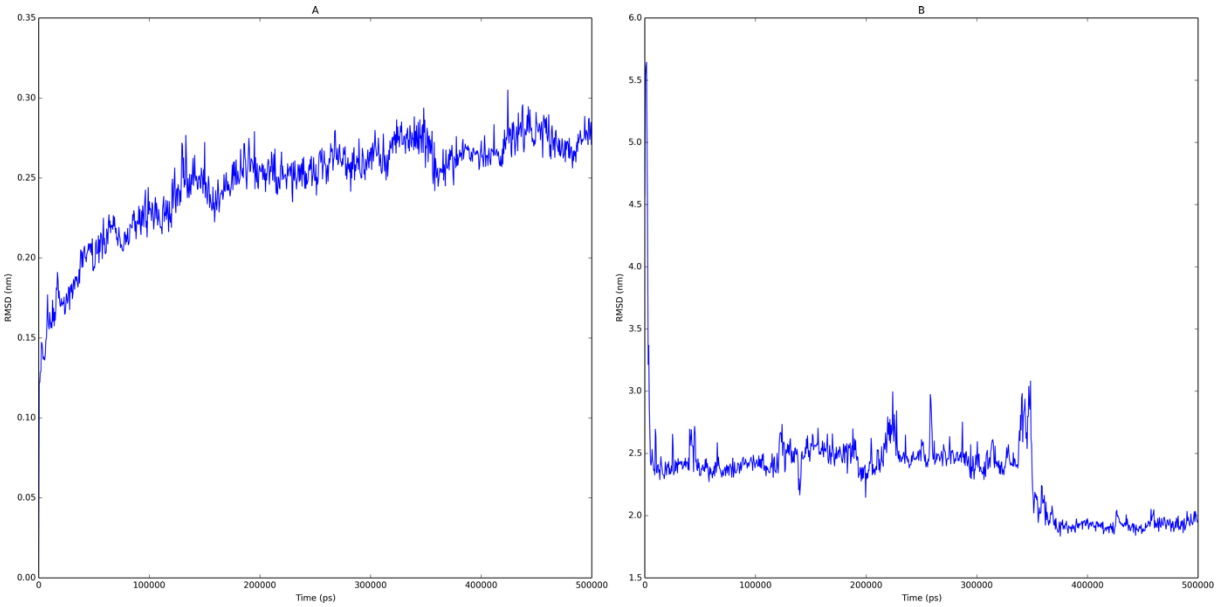


Figure 72. Left: RMSD calculated on the Cα atoms for the PREP protein after fitting the Cα atoms onto those of the first frame. RMSD starts at 0 nm (first frame) and levels out at approximately 0.30 nm. Right: RMSD calculated on atoms of the ligand after fitting the Cα atoms onto those of the first frame. The RMSD of the ligand is much larger (with a maximum of approximately 5.5 nm) compared to the Cα atoms, indicating a larger flexibility and motion.

Root-mean square fluctuation (RMSF)

For each atom, the RMSF measures the fluctuations about a point over an entire simulation, after first aligning the atom coordinates of all frames onto the corresponding coordinates of the first frame:

$$RMSF(i) = \sqrt{\frac{1}{T}\sum_{i=1}^{T}\left(\vec{r}_i(t) - \overrightarrow{r_{i,ref}}\right)^2}$$

with *T* being the total number of frames, $\vec{r}_i(t)$ the coordinate of atom *i* at frame *t*, and $\overrightarrow{r_{i,ref}}$ the coordinate of the corresponding atom in the reference set.

The RMSF is useful for judging which areas of the protein are dynamic or static (Figure 73).

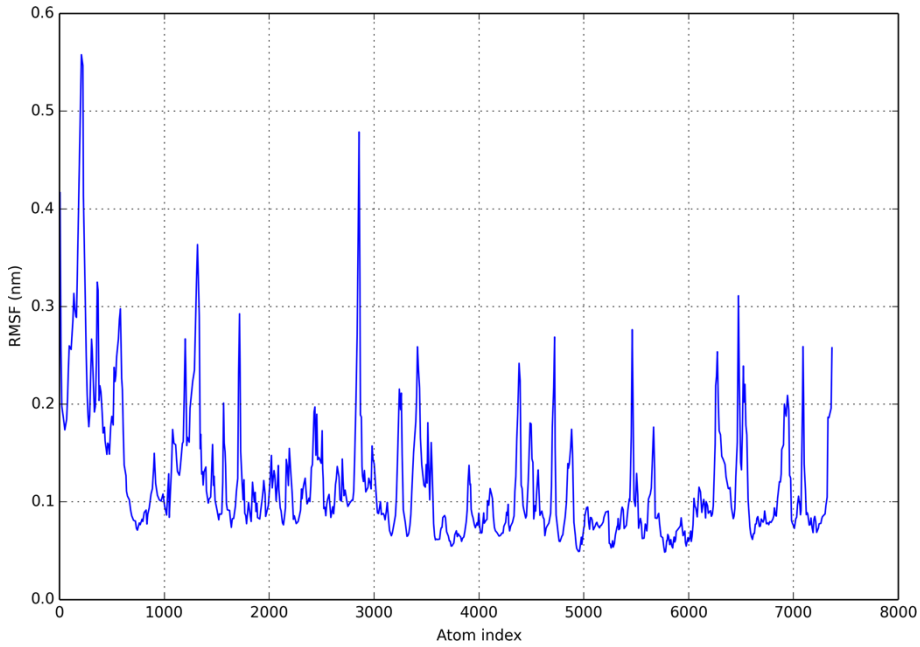


Figure 73. Illustration of a RMSF plot calculated from a 500 ns MD simulation of the PREP protein. The x-axis shows the atom indices of the Cα-carbons of the protein, and the y-axis shows the corresponding calculated RMSF after fitting all frames onto the first frame.

Solvent-accessible surface area (SASA)

The SASA measures the variation of the solvent-accessible surface area of the protein as a function of simulation time, providing information about potential folding or unfolding processes (Figure 74):

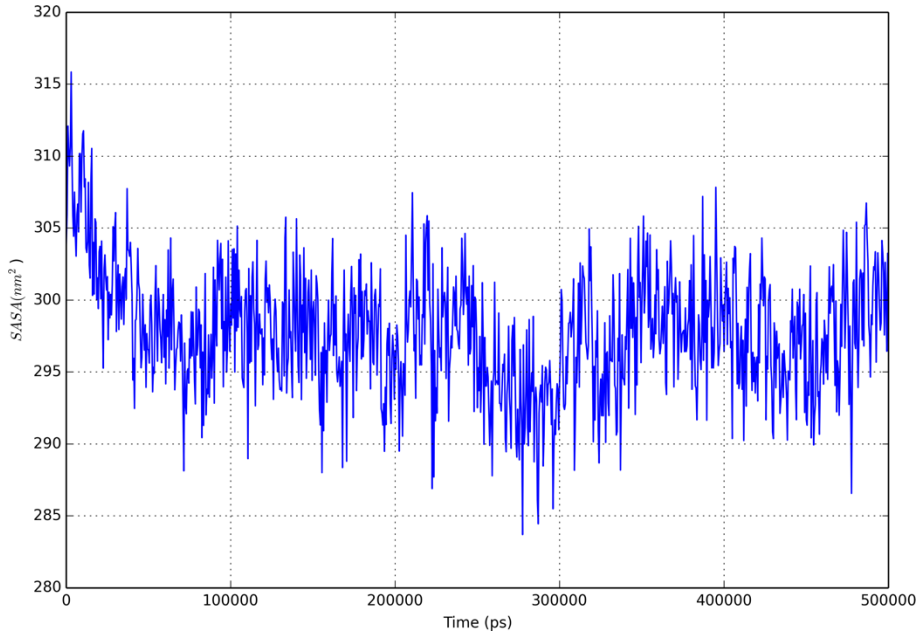


Figure 74. Plot of the SASA as a function of time calculated from a 500 ns MD simulation of the PREP protein. The total SASA (in nm²) is given on the y-axis, and the time (in ps) is given on the x-axis.

Radius of gyration (ROG)

The radius of gyration is the root mean square distance of the protein atoms parts from the molecule's center of mass:

$$ROG(t) = \frac{1}{N}\sum_{i=1}^{N}(\vec{r_i}(t) - \overrightarrow{r_{mean}}(t))^2$$

with *N* being the number of atoms in the protein, $\vec{r_i}(t)$ the coordinate of atom *i* at time *t*, and $\overrightarrow{r_{mean}}(t)$ the mean coordinate of all atoms at time *t*. The larger the value, the 'larger' the molecule. Just like the SASA, the ROG also provides information about potential folding or unfolding processes (Figure 75):

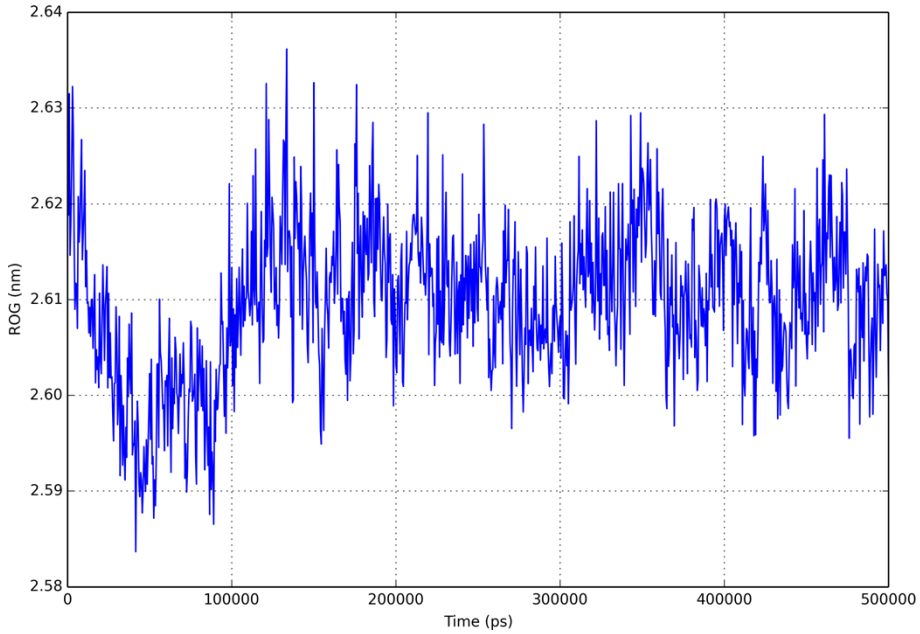


Figure 75. Plot of the ROG as a function of time calculated from a 500 ns MD simulation of the PREP protein. The total ROG (in nm) is given on the y-axis, and the time (in ps) is given on the x-axis. One can see that the protein shows a little 'shrinking around 50,000 ps (50 ns), and then returns to its normal size.