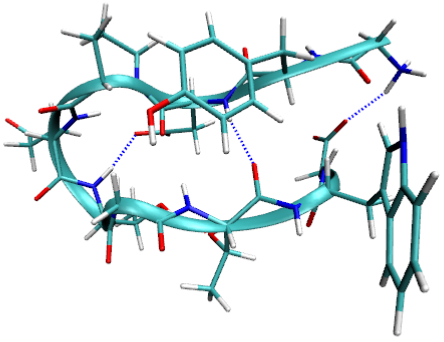
T 38
Lehrstuhl für theoretische
Physik - Molekulardynamik

Technische Universität München

TU München
Physics Department
Chair of Theoretical Biophysics (T38) - Biomolecular Dynamics
Prof. Dr. Martin Zacharias

Lab Course Biophysics

Experiment 74 - Molecucar Dynamics



About this Lab Course

In the first section of this manual a short introduction of proteins and protein properties for this lab course is given, followed by an introduction of how molecular dynamics as a non-quantum model of molecular interaction works.
The practical part, covers two examples of molecular dynamics simulations. In the first example you will set up a small simulation, execute it yourself and get to know the analysis tools by applying them under supervision. The analysis is done visually and more quantitatively with a program that makes it possible to view three dimensional animated "movies" of the simulated molecules. In the second, more realistic example you will apply the analysis skills you have learned on your own and try to get a detailed picture of the folding process of a small protein called chignolin (pictured above), what hinders and enhances it, as seen through a molecular dynamics simulation.
At the end you will have a more detailed understanding of how the vast complexity of molecular behavior is created one small physical interaction at a time.

1 Biophysics

Biomolecules carry out complex tasks in all organisms. From metabolic functions like splitting long sugar chains, to tasks like DNA synthesis which comprises a complex organization of many different very large molecular machines, that unwind, split, fuse together, wind back up again and much more.
Structure and dynamics of biomolecules determine their function. A biomolecule adopts very specific three dimensional structure to perform a specific task. And often the structure changes dynamically as part of the performed functions.

1.1 Protein Structure

Proteins are the most diverse class of biomolecules with active functions, basically involved in every biological process. Their building plan is read directly form the DNA, and in a complex process called protein biosynthesis that can comprise different steps depending on the kind of protein being created, results in long chains of small molecules called amino acids, that can arrange in complex three dimensional structures. Proteins consist normally of more than 20 amino acids and can contain tens of thousand of amino acids.

Amino Acids

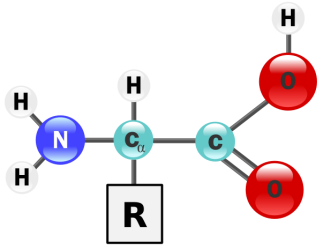


Figure 1.1: *The basic chemical structure of an amino acid.*

The amino acids all share the the same basic structure (fig 1.1), that forms the so called **backbone** of a chain of amino acids that are are bonded together (fig. 1.2). The bonds between amino acids also always follow the same pattern, which means a covalent bond (a chemical bond that is formed by sharing an electron pair) forms between the outer carbon (the non α carbon) of one amino acid with the nitrogen of another amino acid, creating an additional water molecule in the process.
The resulting bond is what is called a **partial double bond**, meaning it shares some of the properties of a double bond, specifically the property that a double bond can not be rotated around. While for a partial double bond rotations are possible, they are very rare in proteins and normally must be induced by something like UV radiation, or only happen on very long timescales which are irrelevant for molecular dynamics.

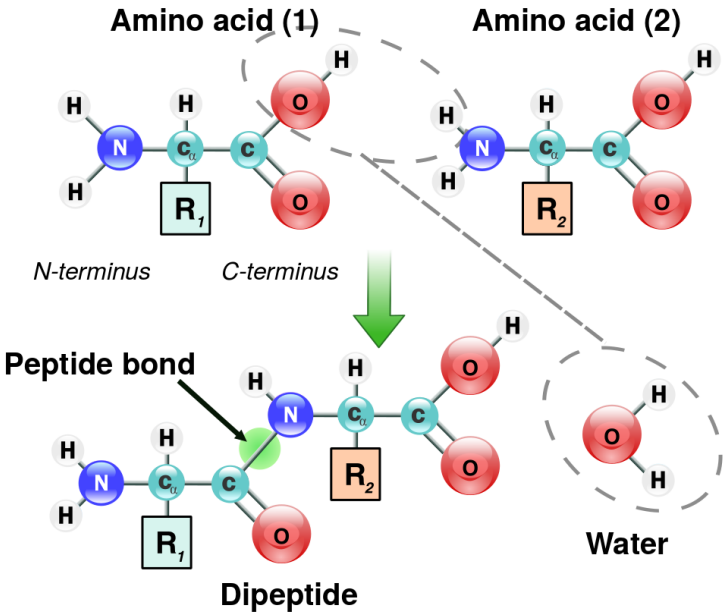


Figure 1.2: *Two amino acids forming a link.*

The fact that the connection of the amino acids always occurs in the same way for all amino acids, also partially explains the unbelievable versatility of proteins, since by DNA mutation one amino acid can be slotted in for another at anywhere along an amino acid chain, potentially changing the functionally not at all, only mildly or even radically, but always staying within the same construction set.
The differences in chemical and mechanical properties of the amino acids are caused by the side chain, marked with **R** in fig 1.1 and 1.2, that is attached to the so called α-carbon, giving these the name α amino acids. The results is an array of 21 different amino acids, which can be seen in fig. 1.3.

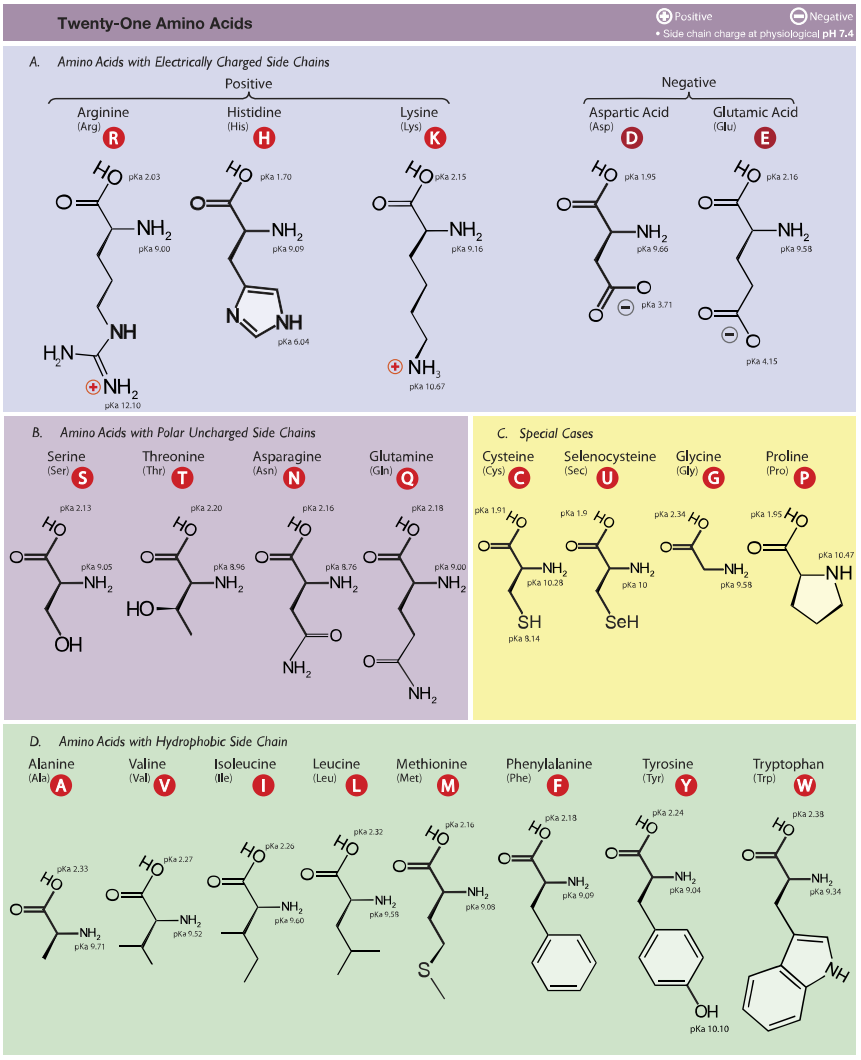


Figure 1.3: *The 21 α amino acids found in eukaryotes. Author: Dan Cojocari, Princess Margaret Cancer Centre, University of Toronto. license: https://creativecommons.org/licenses/by/3.0/*

There are only 20 amino acids that are represented in the genetic code, and the complexity and vast multi functionality of proteins arises from the complicated interplay of the amino acids, when they after being chained together, form three dimensional structures.

Levels of Organization and Common Structural Motifs

The structure of proteins is normally described at four levels of structure: **primary**, **secondary**, **tertiary** and **quaternary**. Primary, secondary and tertiary structure deal with single chains of proteins, which are also called monomers, quaternary structure describes the organization of complex monomers into even bigger units. The higher levels of organization that proteins display, tertiary and quaternary structure are not the topic of this lab manual, because while they are essential to the function of proteins, they are also further out of the reach of investigation with simulations due to the sizes of the structures and the complexity of their formation process.
In the below figures the amino acids are shown in stick and ball representation, partly overlayed with with the backbone representation, which is the turquoise band which is just a spline fitted along the backbone of the protein, for ease of representation.

Primary structure/sequence:
Simply the one dimensional sequence of amino acids in a protein. Below with highlighted backbone.

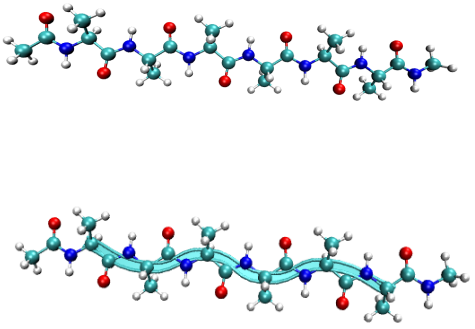
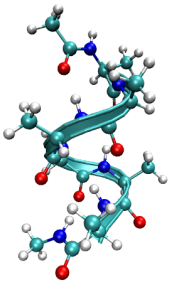


Figure 1.4: *Primary structure of amino acids simply means the sequence of occuring amino acids*

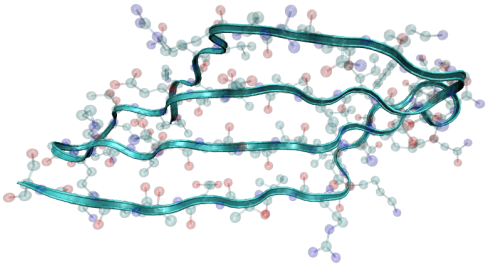
The second level of organization, the secondary structure describes the two most common regular motifs that are locally formed in proteins and the unordered state of a randomly coiled backbone:

Secondary structure:

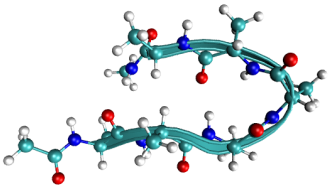
The two most common local structures are the alpha-helix and the beta sheet, which in it's smallest form, is a hairpin like structure (lower right) and can form into larger sheet like structures also (lower left).



alpha-helix



beta sheet



beta-haipin

Figure 1.5: *Most common secondary structures in proteins.*

The α in "α helix" just signifies that the helix was discovered before the β sheet.

Dihedral Angles

The secondary structures can more quantitatively be characterized by the dihedral angles the backbone of the protein forms. The definition of a dihedral angle in an a nucleic acid chain and in the context of a protein backbone is illustrated in figure 1.7. In the backbone of two neighboring amino acids there are always two bonds that the molecule can rotate around, which are the ones inside the amino acid, these angles are named Φ and Ψ and the **peptide bond** who's rotational angle is, statically fixed at 180 degrees because of it's partial double bond nature, it is named ω.

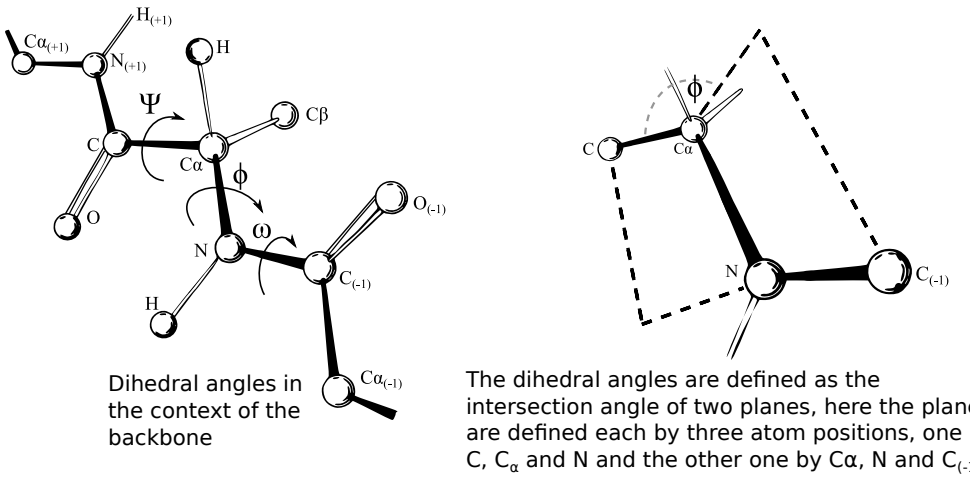


Figure 1.6: *Definition of dihedral angle and dihedral angles in a Protein backbone chain.(By Adam Ręd̨zikowski, liscence: https://creativecommons.org/licenses/by/3.0/)*

The distribution of the dihedral angles in a protein can directly be related to it's secondary structure, this is simply a consequence of geometry. If we plot all dihedral angles in a single diagram the resulting Φ, Ψ plot is called Ramachandran plot. We can clearly delineate regions in the Φ, Ψ plane that will if most of the dihedral angles land within these regions result in the known secondary structures.
Of course the dihedral angles in a protein are not all equally energetically favorable. The rotation around each angle is subject to a potential landscape that is quantum mechanical in origin, the form of the potential of course depends on the surrounding configuration of bound atoms.

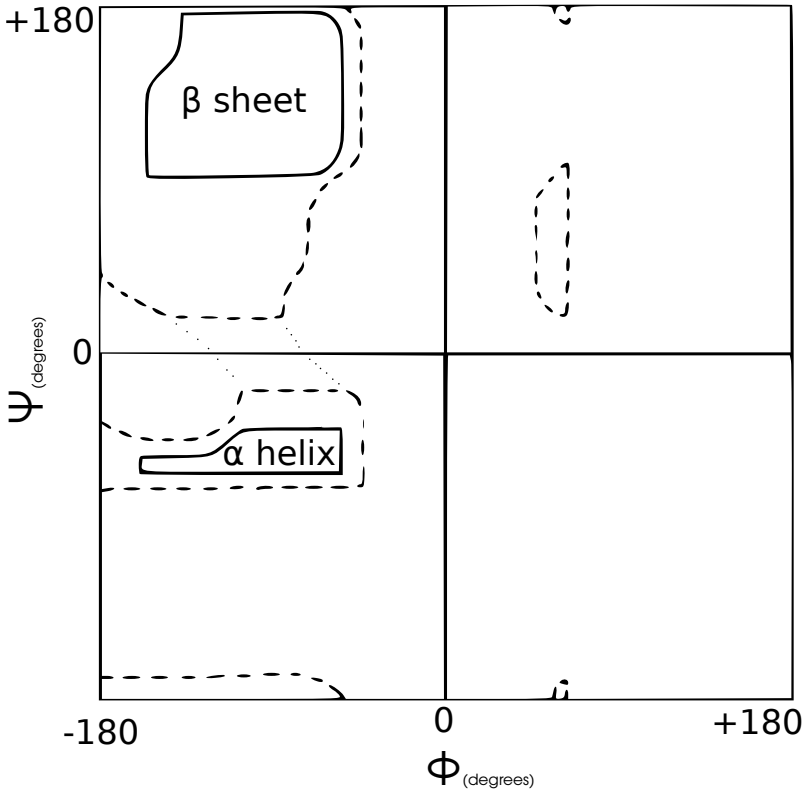


Figure 1.7: *Ramachandran plot with alpha helix and beta sheet regions drawn in.*

Stabilization of Secondary Structure – Hydrogen Bonds

The secondary structures are stabilized mainly by intra molecular hydrogen bonding. Hydrogen bonding can occur whenever a hydrogen (H) atom, that is covalently bound to a more electronegative atom or group, like nitrogen (N) or oxygen (O), comes close to an electronegative atom with one or more free electron pairs. Electronegativity largely scales with the number of protons the atom possesses and describes the ability of an atom to attract partial charges from electrons it shares in covalent bonds.
An example of this is occurs even in water (fig. 1.8). Where the oxygen atom is able to attract partial negative charges δ^- from the hydrogen atoms, leaving the hydrogens with positive charges δ^+. With these partial charges an electrostatic interaction between oxygen atoms and hydrogen atoms of different molecules is possible. This kind of interaction is especially strong with hydrogen because these atoms are the smallest in terms of Van der Waals radius. Which allows a hydrogen to come especially close to the negatively charged oxygen (in this case) atom making the Coulomb interaction bigger then it would be with other larger atom types. The This interaction is dependent on the angle, as the partial charges are somewhat localized.

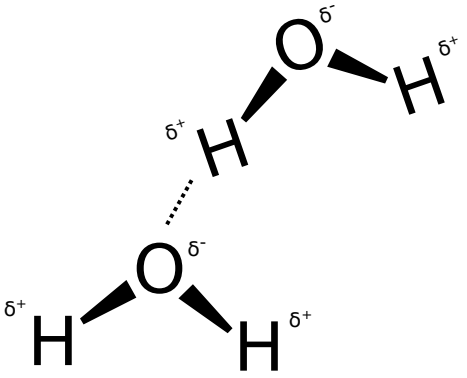


Figure 1.8: *Hydrogen bridge between two water molecules.*

For larger Proteins other effects such as the hydrophobic effect also play a large role, but these are not the topic of this lab course.

Comparing Structures

An important measure when comparing different configurations is the **root-mean-square deviation of atomic positions** (RMSD), within simulations or to experiment.

$$\mathrm{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2} \tag{1.1}$$

Where δ_i is the distance between atom i and it's position in a **reference structure**, which can be an experimental structure, or a frame from a simulation. Often when looking at proteins only the atoms of the backbone are included in the calculation, to minimize fluctuation from the more mobile side chains.
Normally a **rigid body superposition** to the reference structure is performed, minimizing the RMSD (see figure 1.9). This alignment is necessary so both molecules are viewed in the same reference frame and pure rotations of the whole molecule and translations of the center of mass yield a zero RMSD, basically after the alignment only internal degrees of freedom should be having an impact on the RMSD.
Typically the RMSD goes up when the considered configuration very different from the reference and down when more similar. If they are identical it will become zero which is of course extremely improbable.
As the RMSD is a relatively abstract measure it's main use is to identify plateaus in the RMSD which can often correctly be associated with stable or metastable states. When comparing simulated structures to experimental data, the protein data bank (PDB) is an important tool, the PDB is an online database containing the structure data of proteins and nucleic acids, typically by NMR spectroscopy or X-Ray crystallography or cryo-electron microscopy.

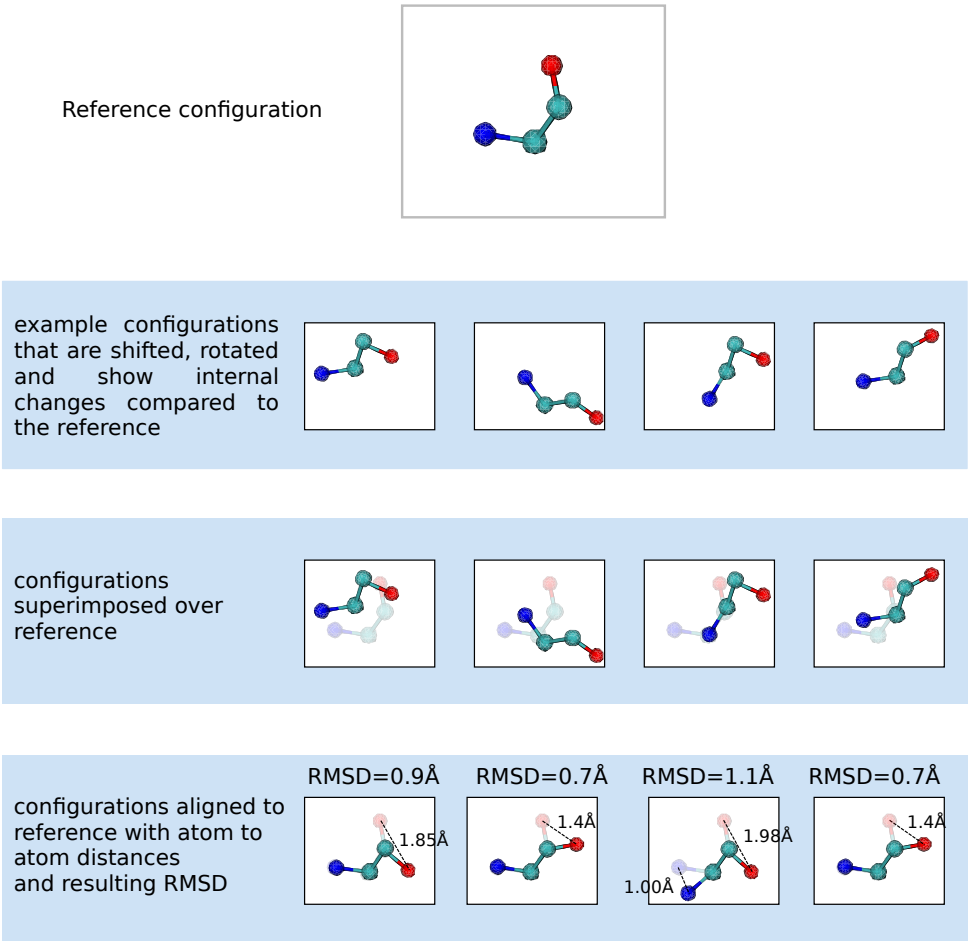


Figure 1.9: *Simple illustration of aligned structures in RMSD calculation for a ficticious example, in four different configurations. Two of the configurations are identical after alignment.*

2 What is Molecular Dynamics?

Quantum Mechanics & Molecular Dynamics

The most accurate models of microscopic systems are quantum mechanical in nature. Many important molecular processes involve chemical reactions, like phosphorylation or formation of covalent peptide bonds. These aspects can only be studied in quantum mechanical models that consider at least the outer electron shells of atoms and their interactions.

The problem of these models is their analytic complexity and the incredible expense in obtaining computational solutions or approximations thereof, even for comparatively small systems and very short time scales.

When considering any process on larger time and size scales, where chemical reactions are not involved, a classical approximation of the quantum properties of molecules like charge distributions, bond strengths and Van der Waals interactions between neutral molecules is often sufficient to capture system properties. The decisive advantage of the classical model of *molecular dynamics* is the considerable increase in scope of simulated time scales as well as system sizes.

Molecular Dynamics – Forces

The concept of molecular dynamics simulations is stated quickly. A small molecular system is simulated by using a classical approximation of all quantum mechanical forces and numerically integrating *Newton's equations of motion*. To solve Newtons equations, first the forces acting on the particles need to be calculated from a potential energy function U. This energy function is called **force field** in molecular dynamics. When talking about a force field both the general analytical form of the potential function as well as the sets of specific parameters that go into it from experiments and resource intensive quantum mechanical simulations are meant.
The force field is simply a potential energy function as known from classical mechanics that can be differentiated to give the force applicable to each atom, giving the equations of motion for an N particle system as:

$$\mathbf{F}(\mathbf{x}) = -\nabla U(\mathbf{x}) \tag{2.1}$$

$$\mathbf{p}_i(t) = \mathbf{v}_i m_i = \frac{d\mathbf{x}_i}{dt} m_i \tag{2.2}$$

where $\mathbf{x} = (\mathbf{r}_1, ..., \mathbf{r}_N, \mathbf{p}_1, ..., \mathbf{p}_N)$ is the vector of all individual atom positions $\mathbf{r}_i$ and momenta $\mathbf{p}_i$. The forces on an individual atom can be calculated by differentiating by it's coordinates. The force field can take different forms depending on the model used.

Force Field Terms

One of the most widely used potential functions is the AMBER force field. The AMBER force field comprises four different types of terms for each atom. The interactions that happen between atoms that are bonded covalently, and electrostatic interactions and Van der Waals interactions that happen between all atoms.

$$U_{\text{total}} = U_{\text{bonded}} + U_{\text{nonbonded}} \tag{2.3}$$

Non Bonded Interactions

The non-bond interactions are split into two parts:

$$U_{\text{nonbonded}} = U_{\text{electrostatic}} + U_{\text{van der Waals}} \tag{2.4}$$

All electrostatic interactions are described by *Coulomb's law*

$$\sum_{i<j} \frac{q_i q_j}{\epsilon R_{ij}}.$$

The atoms effective charge distribution is represented by atom centered point charges. The electronic repulsion of the Pauli exclusion and van der Waals attraction are modeled typically by a *Lennard Jones* potential

$$\sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} \right]$$

as a $1/R^{12}$ and $1/R^6$ respectively.

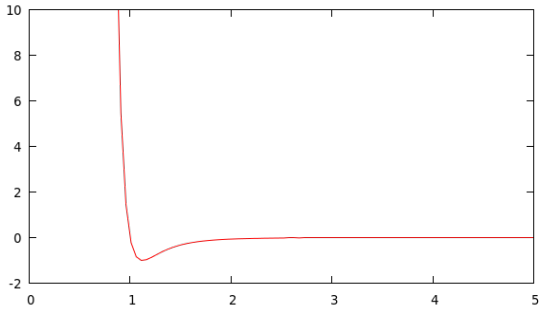


Figure 2.1: *Lennard Jones potential form.*

Both electrostatic and Lennard Jones potential are only calculated for atoms in different molecules, or if they are in the same molecule and separated by three or more bonds.
The calculation of the forces is the largest time investment in molecular dynamics simulations[1], due to this fact a plethora of schemes have been devised to render this process more efficient.

Non Bonded Interaction Cutoff

As the effect of the unbounded interactions beyond a certain distance becomes very small the potential is truncated at a certain *cutoff distance*, to avoid the problems caused by a discontinuous potential a switching function switches the potential and resulting forces to zero smoothly. Although these interactions become very weak over greater distances their total sum in the long distance limit still contributes to the total energy of each particle and thus a analytical correction term is used.

Bonded Interactions

The bond related interactions are modeled by three terms.

$$U_{\text{bonded}} = U_{\text{bond}} + U_{\text{angle}} + U_{\text{dihedral}} \tag{2.5}$$

Bond length stretching and bond angle restoring forces are approximated as harmonic potentials

$$\sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{eq})^2.$$

The dihedral potential is modeled by a truncated Fourier series term

$$\sum_{\text{dihedrals}} \frac{V_n}{2}[1 - \cos(n\phi - \gamma)].$$

Otherwise identical atoms in different molecular configurations are often modeled by different force field parameters.

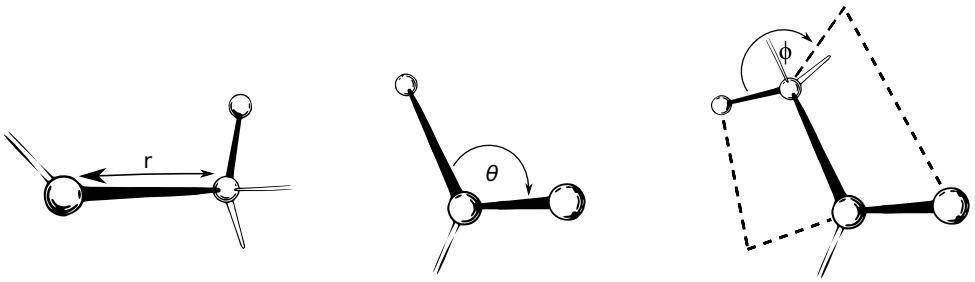


Figure 2.2: *Bonded Interactions.*

Solving the Equations of Motion

When the intermolecular forces are calculated Newton's equations of motion can finally be integrated. The question of how to do this is not a trivial one. Standard numerical schemes for integration of differential equations, like the Euler algorithm, have a good precision in the short term, but in the long term exhibit a drift of physical constants, like the total energy, because these were not incorporated into the schemes themselves[1].

The solution to this problem are *symplectic integrators*, these are systematically derived, using Hamiltonian mechanics and *Liouville propagator* theory, in a way that respects the physical properties of Newton's equations, like conservation of the total energy and time reversibility, while numerically integrating the equations. The most widely used example of this is the *Verlet algorithm*[1], giving the position in the next time step $t + \Delta t$ as a function of the position $\vec{x}(t)$ at the current time step t and the position $\vec{x}(t - \Delta t)$ at the time step before the last, as well as the acceleration $\vec{a}(t)$ at the current time step.

$$\vec{x}(t + \Delta t) = 2\vec{x}(t) - \vec{x}(t - \Delta t) + \vec{a}(t)\Delta t^2 + O(\Delta t^4) \tag{2.6}$$

$$\tag{2.7}$$

Where the acceleration is derived from the potential.

$$\vec{a}(t) = -\frac{1}{m}\nabla U(x(t)) = \frac{\vec{F}}{m} \tag{2.8}$$

The integration via (2.7) is carrying an error of order $O(\Delta t^4)$ analytically and machine errors that are caused for example by the finite number of digits used to represent a float number and rounding errors. To understand why this can cause big problems one can imagine a simple simulation of a planet in the gravitational field of a sun, if the errors are placing the planet evenly a bit too far from, and a bit too close too the sun then every move toward the sun will be conserved because of the potential drawing the planet close while every move away from the sun would be suppressed by the gravity potential. Eventually the planet will dive into the sun.
The Verlet algorithms conserves the potential energy and thus the errors will only cause an oscillation around the proper trajectory, not a catastrophic failure.

The propagation through time then follows a simple loop (fig. 2.3)

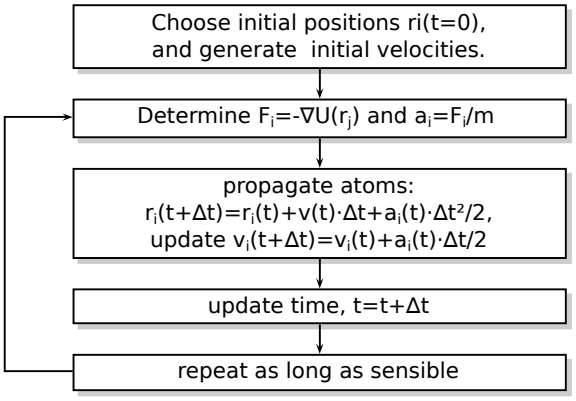


Figure 2.3: *Molecular dynamics propagation loop.*

Temperature

The integration via the Verlet algorithm conserves the potential energy exactly, which is in principle fine, but in practice this is not realistic. A simulation should normally be carried out at a constant temperature, which will comprise a set of different total energies that match the considered temperature. To achieve this the simulation is coupled to a heat bath of fixed temperature, allowing the exchange of energy so that states of different total energy can be sampled.
The simulation of coupling to a heat bath can be achieved in different ways with different up- and downsides. Here a *Langevin thermostat*[2] is used. For this the Langevin equation of motion is employed, which describes the effectively random motion of particles in fluid, due to collision with the fluid particles, called *Brownian motion*. For this a friction term represented by $-\gamma p_i$ and a random force term $\delta F_i(t)$ are introduced into the equation of motion.

$$\begin{aligned}\frac{dr_i}{dt} &= \frac{p_i}{m_i} \\ \frac{dp_i}{dt} &= -\frac{\partial U(r_i)}{\partial r_i} - \gamma p_i + \delta F_i(t).\end{aligned}$$

Here γ is a friction constant and $\delta F(t)$ a *white noise* force term, mathematically characterized by a vanishing mean $\langle \delta F(t) \rangle = 0$ and an *autocorrelation* of the form $\langle \delta F_i(t) \delta F_j(t') \rangle = 2B\delta_{ij}\delta(t-t')$ describing an in time *uncorrelated* force, with B describing its strength. As white noise is characterized by a *Gaussian distribution,* this defines the distribution fully. The equation can be coupled to the temperature by the *fluctuation dissipation theorem*

$$B = k_B T \gamma m,$$

which connects the equipartition theorem for each individual atom $\langle v^2 \rangle_{eq} = 3k_BT/m$ and the movement described by the Langevin equation in the long time limit. The fluctuation dissipation theorem describes the balance between random thermal noise, driving the system , and friction, slowing the system down[3]. This thermostat ensures a canonical velocity distribution and allows more stable use of longer time steps in integrating the equations of motion[2], but will distort diffusion calculations due to the introduced drift.

3 Polyalanine – Simulation and Analysis

We are first going to simulate a short artificial polypeptide that consist of only Alanine amino acids. We will be simulating under artificial conditions, because this will make for a very simple and quick folding transition, making it possible to run a successful simulation in spite of the time constraints of the lab course. This simulation is only for purposes of illustration of the tools and the simulation process, and carries no scientific value.

Building the initial coordinates and topology

We will setup and run an MD simulation of an oligo-alanine peptide in vacuum and with a distance dependent dielectric using the Amber18 simulation package. Before we can start the simulation we need to create the following files:

1. Initial coordinates
2. Topology file (describes the molecular mechanics force field of the peptide)
3. Input files for energy minimization and MD run.

Initial coordinates can be generated either from a database of experimental structures or we can generate the initial structures, using the a program. Here we use tleap from the Amber package to generate the start structure of oligo-alanine. The sequence is 6 alanine amino acids long. Because tleap will simply cut of the Alanines at the ends of the protein, we need to add cap groups that provide more natural starting and ending points, so the sequence is expressed as ACE ALA ALA ALA ALA ALA ALA NME in tleap.
Open tleap in the console, by typeing **tleap** and type the following commands, pressing return after each. We have to load the force field parameters with the **source** command leaprc.protein.ff14SB refers to the specific forcefield we will be loading atom parameters from:

```
source leaprc.protein.ff14SB
```

Use the command **sequence** to create the structure, type:

```
protein = sequence {ACE ALA ALA ALA ALA ALA ALA NME}
```

the coordinates and topology need to be saved to files using the command saveamberparm

```
saveamberparm protein ala.prmtop ala.rst7
```

finally quit tleap by typing:

```
quit
```

Now in your working directory you will find the two files (ala.prmtop, ala.rst7). From this topology and coordinate file we can generate a pdb file which can be used to visualize the peptide structure. The amber package contains a module called ambpdb to do this job. In the command prompt type

```
ambpdb -p ala.prmtop -c ala.rst7 > ala.pdb
```

Use VMD to visualize the structure

```
vmd ala.pdb
```

Task: Notice, that the structure is probably not in a natural configuration right after being generated. (Hint: It's as straight as a mathematical line) Take a picture of this.

Minimising the structure

Before we start running MD we need to perform a short minimization of our starting structure. The energy minimization removes steric clashes, saving the simulation from crashing, and will move the structure to the nearest local minimum, saving us simulation time, as a structure very far from equilibrium will lead to longer times for the simulation to reach the configurations that are more realistic for the molecule.
For this purpose, we need an input file which you can find in the folder (filename `min.in`). Run the minimization with the MD-executable of AMBER called **sander**. The sander program takes several flags:

Sander Flags:
to specify inputs
-i *input command file*
-p *input topology file*
-c *input coordinate file*
-ref *input for restraining reference coordinatefile*
and outputs
-o .out file: *general information output file*
-r .rst7 file: *coordinate output file*
-x .nc file: *for trajectory output (which is not generated by a minimization)*

Here you have an example of what the full command should look like:

```
sander -i min.in -p ala.prmtop -c ala.rst7 -o alaMin.out -r alaMin.rst7
```

If the simulation was successful, we have two more files, alamin.out and alaMin.rst7 in the working directory. Look at the output files `alamin.out` and `alamin.rst7`. Now convert the output coordinate file into pdb, so that we can visualize the structure

```
ambpdb -p ala.prmtop -c alaMin.rst7 > alaMin.pdb
```

Task: Compare the minimized structure to the initial starting structure using vmd. What has changed?

Heating up the system.

The next stage is to run preparatory MD run. As we normally do the MD simulations at 300K (room temperature) we should heat up the system to 300k in a step by step manner. The heating in stages will equilibrate the system at each temperature. We do it in three steps 100K, 200K and 300K and for 10ps in each step. After that we add two steps in which the restraints we use to keep the protein from being disturbed too much are reduced.

Modifying the amber input file In your folder, you can find an input file named `heating1.in`. Make copies of this file and modify the `temp0` parameter appropriately, to create the desired steps in temperature. The simulation will be restrained in this run to minimize any changes in structure during this phase.
Additionally you need to change the parameters **ntx** and **irest**. **ntx=1** means only coordinates will be read, and **irest=0** means this is new simulation, so these are the right settings for the first heating step. After the 100K heating step you need to set **ntx=5** and **irest=1**, so that coordinates as well as velocities are read from the coordinate input file.
Add two more .in files called loosen1.in and loosen2.in with a decreasing force constant, use the input file you created to warm up to 300K as a template. The force constant is set by the parameter **restraint_wt** to be 1 kcal/molÅ^2 in this file, copy and rename the file and change this value to 0.5 and 0.1 respectively.

Setting flags to run the simulation. We want to run the MD simulations using the new input files in sequence. For this create a file called **heating.sh** and input the commands into the file.
The first line using the input file for the first heating step should look like this:

```
sander -O -i heating1.in -p ala.prmtop -c ala.rst7 -o heating1.out
-r heating1.rst7 -x heating1.nc -ref alaMin.rst7
```

For the next lines you need to take into account that we want to continue each run using the coordinate output file (file ending .rst7) of the last run as input for the next simulation and of course use our newly created input files as input. You also need to give new names to all the outputs from run to run, so no files are overwritten by the following simulation. To see which flags you need to change, look at the **explanation of the input and output flags above**.
After adding three heating runs and the loosening runs execute the commands by typing in the terminal:

```
bash heating.sh
```

Now use vmd to look at the RMSD during these simulations using the command below, replace **TRAJECTORIES** with the file names of the .nc trajectory files you just created, in the sequence they where executed in.

```
vmd ala.prmtop TRAJECTORIES
```

Production run

Now we start a production MD simulation, meaning a run meant to produce results, not just prepare the system. Usually one needs to run it for several nanoseconds, but due to the time constraints of this lab course we run it for only 1 ns. The input file for the production run is called '`mdpr.in`'. The machine you are working on has a 4-core processor, to use all 4 efficiently and to decrease the real simulation time, we will use the MPI version of the command:

```
mpiexec -n 4 sander.MPI
```

Please add the relevant input and output flags yourself, we don't need the **-ref** flag this time, as we don't want to restrain the system to a reference state. The run should take around 3 minutes.

4 Chignolin

The second protein we are going to look at is a designed protein, that forms a stable β-hairpin in water. It is composed of only ten amino acids (GLY TYR ASP PRO GLU THR GLY THR TRP GLY), but it still displays interesting and subtle behavior of folding cooperatively. By this we mean that entering certain states makes the reaching of other states more likely, again giving further states a higher probability, in this case leading the protein down the path to folding a β-hairpin. This simulation was conducted in advance due to the time constraints of this lab course.

Chignolin – Analysis

Start by getting a visual overview of what happens in the trajectory. If you have identified the general stages of the folding process, you can go into more detail.
For the detailed analysis go through the steps listed below. Make sure to take appropriate pictures in vmd and note down the frame number in the filename, as well as giving the a descriptive name generally. If you want to, you can try to choose representations that make it easier to see what you are trying to illustrate with the image.
Generally useful representations in VMD are: A representation using the **HBonds** drawing method with angle $35°$, distance 3.3 , and line thickness 3 and a representation using the **NewRibbons** drawing method. Follow these steps for the analysis:

1. Look at a plot of the **end to end distance** and one of the **RMSD** with reference to the first frame. There are at least two states identifiable in these. One easily identifiable stable state and one meta stable state at the beginning of the trajectory.
   a) Explain the differences in the two plots generally. Use the xy_plot.py script to compare these two plots directly.
   b) Regarding the initial metastable state: Why does the end to end distance take values from the same value range as the final stable state, while the RMSD does not do this during this phase?

2. Look at a RMSD with the last frame as a reference. We now want to do a full analysis of the details of the folding process.
   a) Why do we get a more detailed picture of the folding process with this reference? (Look at an analytical derivative along one arbitrary distance for an explanation)
   b) Characterize the folding process in detail, which stages exist? Relate them to the RMSD plots, if possible.
   c) Also use a hydrogen bond count to characterize these stages. Can the hydrogen bonds explain the sequence of folding stages?
   d) What causes the disturbances of the stable state, that can only be seen in the full protein selection RMSD? You can look at an RMSD with only the backbone atom selected to illustrate if the backbone or the side chains are responsible.
   e) This is a bit harder: Try to see what changes when the molecule goes from the meta stable state into the stable state, why can suddenly more HBonds be formed? (This transition is only 100ps long at most. Don't focus on the side chains here.)
   f) If you find anything else interesting document it.

3. To do a sanity check if our simulation is delivering reasonable results, use an RMSD with an experimental structure as a reference, to this download the pdb with the pdb number 1UAO from www.rcsb.org

5 Lab Report

1. MD generally
   - What are goals of MD and what is it's purpose in relation to biophysics, what are limitations and advantages?
   - Explain the basic workings of Molecular dynamics i.e. forcefield, integration of equations of motion and thermostat. (these can be kept short, but the equations for forcefield and integration would be nice.)
2. Explain the procedure of the preparation of an MD simulation for a production run, using what you did with poly alanin as an an example.
3. Analysis of poly alanin and chignolin simulations (please one for each).
   Again, try to answer these questions by finding illustrative pictures and corresponding features in the plots. The chignolin analysis should contain the points from the Chignolin – Analysis section.
   a) What folding stages can you find?
   b) Where does the folding start?
   c) What may hinder/accelerate the folding?
   d) What makes some of the states more stable then others?

Also please keep in mind that this is supposed to be practice for scientific writing, so assume a reader that is not perfectly familiar with everything you write about. This means:

- Illustration descriptions should tell the reader what they are looking at, and give them the information needed to understand the plot.
- If you claim something in the text explain why, if you don't know why make an educated guess but say it is a guess and why you think it is probably correct. Supply more quantitative information like plots, numbers etc. when possible to further your argument.
- Try to order you thoughts from larger to smaller, this should help the reader follow your logic.

Bibliography

[1] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier (formerly published by Academic Press), 2002.

[2] Thomas Soddemann, Burkhard Dnueweg, and Kurt Kremer. Dissipative particle dynamics: A useful thermostat for equilibrium and nonequilibrium molecular dynamics simulations. *Physical Review E*, 68(4):046702, 2003.

[3] Robert Zwanzig. *Nonequilibrium statistical mechanics*. Oxford University Press, 2001.