Introduction to Molecular Dynamics Simulations for Biological Systems

Ángel Piñeiro

27th November 2009

Contents

1 Introduction **2**

2 Historical Evolution **2**

3 Theory **3**
3.1 Initial Conditions 4
3.2 Potential Function 5
3.3 Running Molecular Dynamics Simulations 6
3.4 Trayectory analysis 6

4 Simulating a Real Protein: Lysozyme **7**
4.1 Introduction 7
4.2 Obtain the Protein Structure 8
4.3 Generate the Protein Topology 8
4.4 Prepare the Simulation Box 9
4.5 Minimize the System 9
4.6 Equilibrate the Water Around the Protein 10
4.7 Production Simulation 11
4.8 Trajectory Analysis 12

5 Artificial Peptide Folding **14**
5.1 Introduction 14
5.2 Simulating a Peptide 15
5.3 Artificial Peptide Folding 16

6 References **17**

1 Introduction

Functional proteins are not rigid and static objects; they experience local and global movements at different time scales. This dynamic behaviour plays an essential role in their activity, as well as in their structural stability. However, it is difficult to obtain experimental structural information on individual molecules as a function of time. During the last three decades atomistic molecular dynamics (MD) simulations based on classical force fields have contributed considerably to the understanding of protein performance. By employing this approach, simulated systems can be controlled and manipulated in previously unthinkable ways, providing data on properties and features that can be compared to experimental results, together with other significant information not available from experiments. Among many other applications of this powerful computational technique, it is now possible to describe the complete folding pathway of peptides about 20 residues long, to suggest reliable structures of protein-ligand aggregates that take the solvent into account explicitly, or to study very complex systems like membrane proteins considering specific lipid compositions of the bilayer in addition to the intra and extracelullar environments, all this in atomic detail. During this mini-course, the theoretical bases of **classical** MD simulations **at atomic level** are summarized, and a couple of simple applications to protein systems are studied in detail.

2 Historical Evolution

Thirty years ago, the first molecular dynamics (MD) simulation of a protein at atomic level was performed by J. Andrew McCammon, when he was a research fellow at Harvard collaborating with Martin Karplus. The simulated protein was the small bovine pancreatic trypsin inhibitor (BPTI), with approximately 500 atoms, in vacuum, and the simulation time was only 9.2 ps. Even with these limitations, the significant atomic fluctuations showed by the BPTI in that pioneer work contributed to change the classical view of proteins as rigid structures. Several MD simulations of proteins in vacuum were reported in the following years. The Simple Point Charge (SPC) water model published in 1981 by Berendsen et al. made possible the implementation of biomolecular dynamics simulations with explicit solvent. A variety of water models arose in the early 80's. The former simulations in solution were performed with small proteins or DNA polynucleosides in the ps time scale due to the high computational cost of the resulting systems, with several thousands of water molecules in addition to the solute. The transition from the ps to the ns and μs timescales for systems with nearly 10^4 atoms had place almost simultaneously nearly ten years ago ($\sim$1998). Currently, atomistic MD simulations with many thousands of particles (up to 10^5) and tens of ns are common. Large time and size scales (μs and μm) were achieved with programs and algorithms able to take advantage of the accessibility to many processors simultaneously. Those resources are not currently available to everybody. However, considering the progress on both hardware and algorithms implemented in molecular dynamics engines, and taking the Moore's law[1] as a reference, it is reasonable to guess

[1] Moore's law describes a long-term trend in the history of computing hardware. Since the invention of the integrated circuit in 1958, the number of transistors that can be placed inexpensively on an integrated circuit has increased exponentially, doubling approximately every two years.

that what today is practically unattainable tomorrow will be routine. A conservative extrapolation of representative time scales, following the historical evolution of average time scales of single trajectories, is presented in Figure 1. By assuming that typical simulations of medium-sized proteins in 1990, 1995, 2000, and 2005 were 0.1, 0.5, 5, and 50 ns long, respectively, the average time scale of single trajectories expected for 2020 is longer than 20 μs, longer than 6 s for 2050, and nearly 7 min for 2060. Figure 2 shows a similar estimation performed for the average number of simulated atoms. It considers the 1977 BPTI simulation with 500 atoms, and typical-size simulations of 10^4 and 10^5 atoms in 1998 and 2005, respectively.

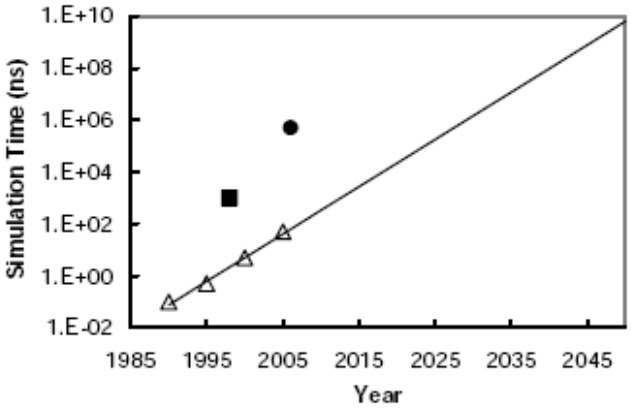
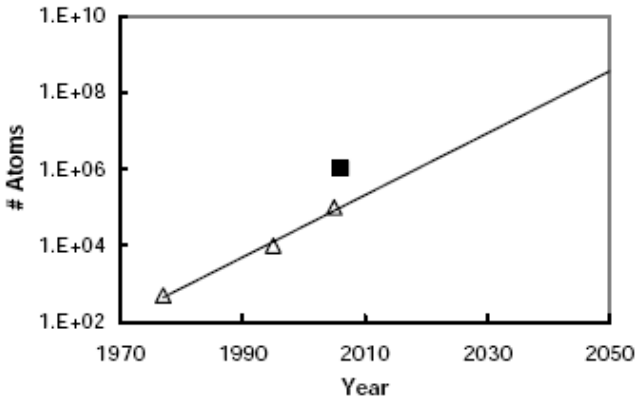


Figure 1: Typical simulation times of protein systems between 1990 and 2005 (triangles). Linear fit extrapolated till 2050 (line). Simulation times reported for the α-helical villin headpiece in 1998 (solid square) and 2006 (solid circle). The simulations performed for the villin subdomain suggest that the extrapolation underestimates the actual progress in this field.

Figure 2: Typical number of atoms of protein systems simulated between 1977 and 2005 (triangles). Linear fit extrapolated till 2050 (line). Number of atoms simulated by Freddolino et al. (solid square). The simulation of the complete satellite tobacco mosaic virus performed in 2006, with more than 10^6 atoms, suggests that the extrapolation underestimates the actual progress in this field.



3 Theory

In order to perform a MD simulation several elements need to be combined:

- The initial conditions that include mainly the composition of the system, atomic coordinates, and initial velocities.

- The topology of the molecules and the potential function that, together, define the interactions of all atoms with each other.
- The conditions under which the simulation will be performed: algorithms to be employed to evaluate the different types of interactions, constraints and restraints to the movement of atomic groups, external forces, time scale of the simulation, time step, frequency at which the different data will be saved, etc.
- A molecular dynamics engine, i. e., the program and the electronic devices that will compute the dynamic evolution of the system and will store the results.
- Finally the results should be analyzed.

These items are briefly described in this section.

3.1 Initial Conditions

Typical molecular dynamics simulations describe the movement of every atom in the system being studied, by solving numerically the Newton or Lagrange equations of motion. This means that quantum effects such as proton transfer, high frequency degrees of freedom, or low temperatures are not commonly treated. To perform a MD simulation, an initial set of the atomic coordinates and velocities is required. The protein coordinates are generally taken from the Protein Data Bank if the structure of the molecule was previously determined by NMR or X-ray spectroscopy, or from computational models when the experimental structure is unavailable. The initial conformation of the protein to be studied, defined by such coordinates, is then rotated and translated as a whole in a convenient way, to introduce it in a virtual mathematical box bigger than the molecule itself. When the solvent employed is homogeneous, pre-equilibrated small boxes of the solvent are successively copied throughout the main simulation box and, in a following step, the added molecules which overlap any protein atom are removed from the coordinates file. The resulting system is a box completely filled with the protein and the solvent atoms. Sometimes it is convenient to include also ions or different kinds and amounts of molecules; this is performed similarly to the addition of the main solvent. A special case is the simulation of membrane proteins since such macromolecules have to be introduced in a lipid bilayer with a certain orientation before adding the solvent as described above. Periodic boundary conditions are usually employed to avoid artefacts caused by nonrealistic walls. This means that replicas of the box, also termed the unit cell of the simulation, are infinitely copied in three dimensions in such a way that the Euclidean space is completely filled. The use of periodic boundary conditions restricts the geometry of the simulation box. Classical boxes are cubic or rectangular, although nowadays octahedron or dodecahedron boxes are common in order to optimize the size of the system and the isotropy of any artifact due to periodic boundary conditions. The initial velocities of every atom in the system are usually randomly assigned under the restriction that they follow a Maxwell distribution corresponding to the temperature at which the MD simulation will be performed.

3.2 Potential Function

In addition to the coordinates and velocities, to determine the dynamics of the system, the force on every atom has to be calculated. This is done by means of the following expression:

$$F = -\vec{\nabla} V \tag{1}$$

where $\vec{\nabla}$ is the well known vector differential operator defined by

$$\vec{\nabla} = \frac{\partial}{\partial x}\,\vec{i} + \frac{\partial}{\partial y}\,\vec{j} + \frac{\partial}{\partial z}\,\vec{k}$$

and V is a semi-empirical potential function for which several expressions and sets of parameters, altogether called the force field, have been proposed. A typical molecular force field has the following form:

$$V = V_{bonded} + V_{non\text{-}bonded} + V_{special} \tag{2}$$

where

$$V_{bonded} = V_{bonds} + V_{angles} + V_{improper} + V_{proper} \tag{3}$$

and

$$V_{nonbonded} = V_{L\text{-}J} + V_{Coulomb} \tag{4}$$

and $V_{special}$ is an optional potential function artificially imposed to bias the behaviour of the system in a special or particular way. Typically, $V_{special}$ is used to restrain the positions or distances of particular groups of atoms. V_{bonded} includes the covalent bond-stretching (V_{bonds}), angle-bending (V_{angle}), and improper dihedrals or torsions ($V_{improper}$) interactions, which are, all of them, modelled by means of harmonic potentials. Proper dihedrals (V_{proper}) interactions, also named rotations of molecular groups, are typically defined by a sinusoidal function. The four contributions to V_{bonded} act between atoms separated by less than three covalent bonds. $V_{non\text{-}bonded}$ is pair-additive for all the atoms in the system. Typically, it contains a Lennard- Jones potential (V_{L-J}) for the dispersion and repulsion terms, and a Coulomb potential ($V_{Coulomb}$) for (partially) charged atoms. V_{L-J} is a short range interaction and only the contribution of the atoms contained in a sphere with a radius shorter than 1 nm is significant. However, $V_{Coulomb}$ has important effects even at long molecular-scale distances. Since it is evaluated by means of a pair-additive potential, the number of contributions for a N-particles system is N(N$-$1)/2, so it is said to scale as N^2. Many algorithms, grouped in lattice summation techniques and fast multipole methods, have been developed to save computational time in the evaluation of this latter contribution, because it represents the highest percentage of the resources employed in a molecular dynamics simulation. The most popular of these techniques are the artificial use of some cut-off for the long range interactions, the Ewald summation, and its efficient discrete refinements: the Particle Particle - Particle Mesh (PPPM or P3M), and the Particle Mesh Ewald (PME) methods. P3M and PME scale as N ln(N). Each of the potential contributions needs a series of atom-dependent parameters like spring constants and equilibrium distances to be included in the harmonic potentials, the amplitude and reference angle in the proper dihedral, the L-J radii, and the partial charges for the Coulomb term. Several works focusing in the comparison of different force fields and parametrizations have been published. They

conclude that in many cases at the time scales studied, the results do not depend on the combination of equations and parameters employed. CHARM, AMBER, GROMOS, and OPLS are among the most employed force fields in MD simulations of protein systems.

3.3 Running Molecular Dynamics Simulations

The set of parameters defining all the interactions in a molecular system is named "topology" and contains all the information needed to calculate the force on each particle as a function of the coordinates of all the atoms in the system. The trajectory of every atom is calculated from the force associated to it, by integration of the movement equation during very small time steps (Δt) typically of 1 or 2 fs (1 fs = 10^{-15} s). Then, the coordinates and velocities of all the atoms in the system are updated. The integration is performed by using specific algorithms such as the popular Verlet or some of its variants, which assume that the force on every atom is constant during the time step. The time step by itself is also a critical parameter that determines both the stability and the speed at which the MD simulation is performed. Stable simulations require a time step at least 4 times lower than the minimum period, corresponding to the fastest degree of freedom, of the system. The degrees of freedom of the simulated system can be grouped as a function of their frequency. For instance, it is well-known that bond-length vibrations happen at frequencies within the quantum limit and they are usually disregarded in classical simulations by considering fix bond lengths. It has been proved that this assumption does not change the dynamics of the system, in contrast with restrictions imposed on angles. Fastest degrees of freedom in biological MD simulations are those involving hydrogen atoms due to their low mass and several approaches have been proposed to make them slower. Under the previous conditions, the total energy of the simulated system should be constant, but cumulative errors, mainly due to integration schemes and computer precision, produce drift energies. A number of algorithms are employed to perform the simulation under a particular thermodynamic ensemble. The most common of these algorithms are thermostats and barostats which correct the velocities and coordinates of the atoms to keep constant the temperature and pressure, respectively, of the system. Several molecular dynamics engines are currently available, the best known being CHARM, AMBER, GROMACS, GROMOS, NAMD, Q, and TINKER. By using these computational packages, molecular dynamics simulations of biological systems are accessible even for non specialized people. However, considering the multiple options, algorithms and parameters involved in any simulation, it is strongly recommendable to know the theoretical basis of this technique before working with it.

3.4 Trayectory analysis

During the simulation, several data, such as the coordinates and velocities of all the atoms, the temperature, pressure, and energy contributions, are stored for further analysis. The resulting files are usually very large due to the number of particles involved. Thus, only data corresponding to one of every 10^2 to 10^4 steps are typically stored. The analysis performed on the trajectory will depend on the objectives of the simulation itself. Usually it is specific for each work but for protein systems the analysis of several properties is standard. Before starting to determine structural or energetic properties, one should visualize the trajectory

by means of some molecular viewer, like for example RASMOL, PyMOL, or VMD. Such programs allow the representation of proteins in different ways and colours, generating high quality images and movies. The deviation of a protein from a reference conformation, usually the experimental or model structure, as a function of the time is normally quantified by means of the Root Mean Square positional Deviation (RMSD). RMSD plots appear in almost every publication dealing with molecular dynamics simulations of proteins systems. Typically such plots present a sharp slope during the first few nanoseconds and oscillate around a constant average value during the rest of the simulation. Another typical analysis for these systems is the calculation of Root Mean Square positional Fluctuations (RMSF) averaged for each protein residue. RMSF is defined as the standard deviation of the RMSD values throughout the trajectory. RMSF values are closely related to the crystallographic B-factors, both of them measuring the magnitude of the fluctuations of every residue in a protein. Proteins radii of gyration can also be easily calculated from MD simulations and are specially useful when comparing the dependence of proteins MD simulations on different conditions such as force fields or water models and in protein folding simulations. Another important and very common analysis in MD simulations of protein systems is the identification of secondary structure blocks of amino acids. The Dictionary of Secondary Structure of Proteins (DSSP) defines different kinds of local structure as a function of geometrical parameters like intramolecular angles and distances. Such definitions are employed to identify residues with common characteristics. Then, plots with a colour code for every type of secondary structure are displayed as a function of the time. Visual analysis, RMSD, RMSF, radius of gyration and determination of secondary structure are just examples of the most typical analysis performed on simulations of protein systems but, from the information contained in a MD trajectory, a very large number of properties can be calculated including fluctuation correlations between different parts of the system, calculus of energetic contributions, temperature, pressure, determination of relative orientations and distribution of molecules, and estimation of thermodynamic properties such as binding energies.

4 Simulating a Real Protein: Lysozyme

4.1 Introduction

The aim of this section is to show you how to start a molecular dynamics simulation of an actual protein, as well as how to analyze the resulting trajectory, by using the GROMACS package. This is a very basic course and we will not go into fine details of algorithms and commands but following this tutorial you are expected to be able to run and analyze a basic Protein MD simulation.

GROMACS (acronym for GROningen MAchine for Chemical Simulations) is one of the most powerful programs package for general purpose MD simulations. It has been recognized to be the fastest engine for MD simulations. Additionally it is free, supports a wide variey of force fields, and it is very flexible. Other program packages are available but we have adopted GROMACS for this course.

4.2 Obtain the Protein Structure

Lysozyme was chosen as the victim for your first MD simulation. It is one of the most studied proteins. It is present in tears, saliva, mucus, and egg white. If you do not know this protein and are curious, just type lysozyme in your preferred web browser.

The first thing you will need is a structure of the protein to be simulated. It can be downloaded from the Protein Data Bank at `http://www.pdb.org`. Look for the structure with code 1LYD and download the pdb file in a new folder created for this exercise. Alternatively you can type:

```
wget http://www.pdb.org/pdb/files/1lyd.pdb.gz
gunzip 1lyd.pdb.gz
```

to get the pdb file from the command line in a linux terminal.

It is important to know the system to be simulated and to have clear what you want to look for in your simulation. You should always take a look to the structure (use, for instance, `PyMOL`), take note of the most important parameters of the structure, number of residues, structure, etc. In this case, you should notice that there are some Oxygen atoms around the protein structure. They are part of the water molecules that remained after the crystallization process. It is possible to add the corresponding Hydrogen atoms to observe these waters during the simulation but just for simplicity they will be removed by editing the pdb file. This can be done by hand using a text editor such as gedit or emacs or by using the following command:

```
egrep -v "HOH" 1lyd.pdb > lysozyme0.pdb
```

4.3 Generate the Protein Topology

Next you need to create a topology for your molecule. This is one of the most delicated parts of the MD input preparation since the behavior of the molecule will depend critically on this topology. Fortunately GROMACS do this automatically with the following command:

```
pdb2gmx -f lysozyme0.pdb -o lysozyme.pdb -v
```

As you can see the program asks you for a force field; please, **choose the option 4** that corresponds to the 53a6 parameterization of the GROMOS 96 force field. The topology of the molecule will be saved in a file named "topol.top". Please, open this file with a text editor (perhaps "gedit" or "kwrite"). Would you imagine to generate all that information by hand? Now you should thank to "pdb2gmx" for its work! The ability of "pdb2gmx" to generate topologies depends on a database of building blocks. The 20 natural aminoacids form part of this database, as well as many other molecular groups including several solvent molecules and lipids, but unfortunately not all the molecules can be handled in this way.

4.4 Prepare the Simulation Box

Now you should introduce your molecule in a simulation box and fill it with water. For this you should use the following commands:

```
editconf -f lysozyme.pdb -bt dodecahedron -d 0.8 -o lysozyme.pdb
genbox -cp lysozyme.pdb -cs -p topol.top -o lysozyme_w.pdb
```

How many water molecules were added? Notice that the "editconf" command allows you to use boxes of different geometries. In this case we are employing a dodecahedron-shaped box but you can choose other alternatives (try them!). Take a look to the output file "lysozyme_w.pdb" using PyMOL. Probably the view of the system is a bit strange with the protein in a corner of the simulation box. This is due to the geometry of the box. The coordinates can be modified to produce a better view with your preferred molecular viewer using the following command:

```
echo 0 | trjconv -s lysozyme_w.pdb -f lysozyme_w.pdb -o lyso_view.pdb
-pbc atom -ur compact
```

Take now a look to the file "lyso_view.pdb" with PyMOL.

4.5 Minimize the System

Before starting the MD simulation the system should be minimized to avoid unrealistic interactions. This will be done using the steepest descent algorithm that slightly moves the atomic positions optimizing the interatomic interactions and the total potential energy of the system. For this you should create a text (call it "em.mdp") file with the following information:

```
title        = LYSOZYME ENERGY MINIMIZATION
integrator   = steep
nsteps       = 500
constraints  = none
nstlist      = 1
ns-type      = Grid
pbc          = xyz
rlist        = 0.8
coulombtype  = cut-off
rcoulomb     = 1.4
vdw-type     = cut-off
rvdw         = 1.4
nstenergy    = 20
```

Next, a binary file will be generated combining the coordinates file "lysozyme_w.pdb", the topology file "topol.top", and the minimization parameters file "em.mdp":

```
grompp -f em.mdp -p topol.top -c lysozyme_w.pdb -o em.tpr
```

Read carefully the output of this program and you will see that the charge of the system is +8. It is not realistic to have a system with net charge, so some ions will be introduced. We need at least 8 negative ions to compensate for this charge. We will add 4 Na^+ and 12 Cl^-:

```
echo 12 | genion -s em.tpr -o lysozyme_ions.pdb -p -np 4 -nn 12 -pname
NA+ -nname CL-
```

Let's build again the binary file:

```
grompp -f em.mdp -p topol.top -c lysozyme_ions.pdb -o em.tpr
```

and the minimization can be performed. This will be done in a new folder, so type:

```
mkdir Rmin
mv em.tpr Rmin/
cd Rmin
```

Start the minimization with the command:

```
mdrun -v -s em.tpr
```

The minimization should finish after 500 steps, as indicated in the "em.mdp" file. This should take a couple of minutes. As a result several files should be created, including "confout.gro" that contains the coordinates of all the atoms of the system -the same information that the pdb file. Now the minimization is finished and we can move back to the previous directory.

4.6 Equilibrate the Water Around the Protein

Before starting the simulation it is convenient to equilibrate the water molecules by performing a short MD simulation with the position of the heavy protein atoms restrained. For this we will need to create a file (call it "eqwat.mdp") with the following text:

title	= WATER EQUILIBRATION AROUND THE PROTEIN
integrator	= md
nsteps	= 500 ; 50000 is better
constraints	= all-bonds
dt	= 0.002
nstlist	= 1
ns-type	= Grid
rlist	= 0.8
coulombtype	= cut-off
rcoulomb	= 1.4
vdw-type	= cut-off
rvdw	= 1.4
nstenergy	= 100
tcoupl	= Berendsen
tc-grps	= protein non-protein
tau-t	= 0.1 0.1
ref-t	= 298 298
Pcoupl	= Berendsen
tau-p	= 1.0
compressibility	= 5e-5
ref-p	= 1.0
define	= -DPOSRES

Again the coordinates, topology and simulation parameters files need to be assembled into a binary file. The coordinates file employed is the output file "Rmin/confout.gro" obtained from the minimization:

```
grompp -f eqwat.mdp -p topol.top -c Rmin/confout.gro -o eqwat.tpr
```

As for the case of the minimization, a new directory is created to perform the equilibration of the water molecules:

```
mkdir Reqwat
cd Reqwat
cp ../eqwat.tpr .
mdrun -v -s eqwat.tpr
```

Again, after a couple of minutes, the short equilibration finish and several new files appear in the directory. The coordinates of the final structure will be in the "confout.gro" file. Go one directory back.

4.7 Production Simulation

The system is now prepared to run the simulation. The process is similar to the previous ones, the only changes are the simulation parameters file and the initial coordinates file. This time the following parameters will be employed:

title	= PRODUCTION SIMULATION
integrator	= md
nsteps	= 1000; 5000000 for 10 ns
dt	= 0.002
constraints	= all-bonds
nstlist	= 1
rlist	= 0.8
coulombtype	= cut-off
rcoulomb	= 1.4
vdw-type	= cut-off
rvdw	= 1.4
tcoupl	= Berendsen
tc-grps	= protein non-protein
tau-t	= 0.1 0.1
ref-t	= 298 298
Pcoupl	= Berendsen
tau-p	= 1.0
compressibility	= 5e-5
ref-p	= 1.0
nstenergy	= 100
nstxout	= 1000
nstvout	= 10000
nstxtcout	= 1000

Introduce this information in a file called "run.mdp". The binary file is created by combining again the coordinates, topology and simulation parameters files:

```
grompp -f run.mdp -p topol.top -c Reqwat/confout.gro -o run.tpr
```

and the simulation is ready to start:

```
mkdir Rrun
cd Rrun
cp ../run.tpr .
mdrun -v -s run.tpr
```

Now you can go back again to the previous directory.

4.8 Trajectory Analysis

With the simulation parameters file specified above you will obtain a too short trajectory to be analyzed. We have precalculated two significantly long trajectories (12 ns) that you will use for analysis. Please, copy the folder named "MDlyso" from "/media/compartidos/" to your current directory. In that folder you will see two "xtc" files named "298K.xtc" and "500K.xtc"; two "edr" files named "298K.edr" and "500K.edr"; and one "tpr" file named "topol.tpr". The "xtc" files are two relatively long trajectories for the same system you have been working with. The main difference between both trajectories is the simulation

temperature that, as you will see, significantly affects the resulting protein behavior.

A preliminary visual analysis is performed by extracting some protein conformations from the trajectories. Use the tool `trjconv` to do that:

```
trjconv -f 298K.xtc -o 12ns298K.pdb -dump 12000 -s
trjconv -f 500K.xtc -o 12ns500K.pdb -dump 12000 -s
```

Use PyMOL to compare those structures and, if you want, any other intermediate structure just changing the "12000" at the end of the command by the corresponding simulated time in picoseconds (change also the output file after the "-o" flag). Start the quantitative analysis by calculating the Root Mean Square positional Deviation (RMSD) of the protein backbone as a function of time, taking the initial (crystal) structure as a reference:

```
echo 4 4 | g_rms -f 298K.xtc -o rmsd298K.xvg
echo 4 4 | g_rms -f 500K.xtc -o rmsd500K.xvg
```

Take a look to the plots with xmgrace:

```
xmgrace rmsd298K.xvg rmsd500K.xvg
```

The Root Mean Square positional Fluctuation (RMSF) of the protein backbone are the standard deviation of the RMSD values throughout the trajectory. RMSF values represent the mobility of the studied atomic groups. The RMSF per residue backbone is determined by typing:

```
echo 1 | g_rmsf -f 298K.xtc -s topol.tpr -o rmsf298K.xvg -res
echo 1 | g_rmsf -f 500K.xtc -s topol.tpr -o rmsf500K.xvg -res
```

Again, the "rmsf???K.xvg" file can be visualized with xmgrace.

To calculate the evolution of the secondary structure as a function of time:

```
echo 1 | do_dssp -f 298K.xtc -s -o ss298K.xpm
echo 1 | do_dssp -f 500K.xtc -s -o ss500K.xpm
```

This command produces a xpm file that you can convert to postscript:

```
xpm2ps -f ss298K.xpm -o ss298K.eps -by 1
xpm2ps -f ss500K.xpm -o ss500K.eps -by 1
```

To determine and visualize the radius of gyration as a function of time, type:

```
echo 1 | g_gyrate -f 298K.xtc -s -o gyrate298K.xvg
echo 1 | g_gyrate -f 500K.xtc -s -o gyrate500K.xvg
xmgrace gyrate298K.xvg gyrate500K.xvg
```

The number of intramolecular protein hydrogen-bonds is determined as follows:

```
echo 1 1 | g_hbond -f 298K.xtc -s -num hbintraprot298K.xvg
echo 1 1 | g_hbond -f 500K.xtc -s -num hbintraprot500K.xvg
```

It is also interesting to calculate the number of protein-water hydrogen-bonds (it is quite slow):

```
   echo 1 12 | g_hbond -f 298K.xtc -s -num hbprotwat298K.xvg -b 10000
-e 12000
   echo 1 12 | g_hbond -f 500K.xtc -s -num hbprotwat500K.xvg -b 10000
-e 12000
```

The plots can be observed with xmgrace. Let's generate distributions from this type of plots, for instance:

```
   g_analyze -f hbintraprot298K.xvg -dist distrhbintraprot298K.xvg
```

That can also be visualized with xmgrace.

The following command allows to determine the evolution of the hydrophobic area per molecule exposed to the solvent (SAS) as a function of time, the SAS per residue and the volume of the protein:

```
   g_sas -f 298K.xtc -s topol.tpr -o area298K.xvg -or resarea298K.xvg -tv
volume298K.xvg
   g_sas -f 500K.xtc -s topol.tpr -o area500K.xvg -or resarea500K.xvg -tv
volume500K.xvg
```

To complete the analysis, try the different options of the program `g_energy` using the files "298K.edr" or "500K.edr" as input with the flag "-f". From all this analysis generate a report including your conclusions about the behaviour of the protein (structural, dynamic, and energetic) at each temperature.

5 Artificial Peptide Folding

5.1 Introduction

The previous example illustrates how to perform a simulation of an actual protein. In the present section you will see the power of MD simulations to generate specific structures by using special − sometimes non-physical − potentials.

Single and bundles of α-helix peptides are common in biology. Many proteins are stabilized by domains of several helices interacting to each other and the structure of some proteins (like, for instance, apolipoproteins) is almost 100 percent helical. Just to put a representative example, several families of membrane receptors like that of the rhodopsin − G-protein coupled receptors of family A − that are involved in the hormonal system and present seven transmembrane α-helices, are very attractive targets for drug development.

5.2 Simulating a Peptide

An artificial structure of a peptide can be created using PyMOL. This can be done interactively or from the command line. Since we prefer to use the command line lets start by creating a file (call it buildpeptide.pml) with the following text:

```
for aa in "AAAAAAAAAAAAAAAAAAAA":  cmd._alt(string.lower(aa))
save peptide.pdb
quit
```

and then type:

```
pymol buildpeptide.pml
```

This will create a 20 residues long polyalanine which structure you can see with PyMOL. Now, try to do a short simulation (100 ps long) of this peptide in vacuum following the instructions given for the lysozyme with some slight changes. In order to create the topology of the peptide type:

```
pdb2gmx -f peptide.pdb -o peptide.pdb -v -ignh -ter
```

where you should choose again the G53a6 force field (option 4) and the option 1 for both terminals. Thus, you will use uncharged amino and carboxylic termini. This is because the electrostatic interactions in vacuum are strong due to its low permitivity and, using charged termini, these groups would tend to stay together throughout the trajectory.

To minimize the structure, use the following "em.mdp":

title	= PEPTIDE ENERGY MINIMIZATION
integrator	= steep
nsteps	= 500
nstlist	= 1
ns-type	= Grid
pbc	= xyz
rlist	= 0.8
coulombtype	= cut-off
rcoulomb	= 1.4
vdw-type	= cut-off
rvdw	= 1.4
nstenergy	= 20

Since there is no water, the short equilibration simulation performed for lysozyme before the production simulation is not necessary in this case. Additionally, the pressure control has no sense in vacuum. Finally, it is better to use a 1 fs timestep for these unrealistic conditions. For all these reasons, use the following "run.mdp" for the production simulation:

title	= PRODUCTION SIMULATION
integrator	= md
nsteps	= 100000 ; 100 ps
dt	= 0.001
nstlist	= 1
ns-type	= Grid
pbc	= xyz
rlist	= 0.8
coulombtype	= cut-off
rcoulomb	= 1.4
vdw-type	= cut-off
rvdw	= 1.4
tcoupl	= Berendsen
tc-grps	= protein
tau-t	= 0.1
ref-t	= 298
nstenergy	= 100
nstxout	= 1000
nstvout	= 10000
nstxtcout	= 1000

Take a picture of the final structure and calculate the RMSD, the radius of gyration and the evolution of the secondary structure.

5.3 Artificial Peptide Folding

Using the same initial configuration and topology file of the previous subsection you are now going to push the folding of the peptide into a α-helix. For this you will introduce an artificial potential that will take the atoms to the position they should have in the helix. The selection of these residues will be performed by using the following instruction:

```
mkdisres.sh 1 20 peptide.pdb disre.itp
```

this is not a gromacs instruction but a linux home-made script that we wrote for this exercise.

Then modify the "run.mdp" file used for the previous simulation, just adding one line with the following text:

disre = simple

Modify also the "topol.top" file employed for the previous simulation typing:

```
sed -e 's/; Include Position restraint file/#include "disre.itp"/'
topol.top > tmp
```

```
mv tmp topol.top
```

Now you can run the simulation:

```
grompp -f run.mdp -p topol.top -c Rmin/confout.gro -o runfold.tpr
mkdir Rrunfold
cd Rrunfold
cp ../runfold.tpr .
mdrun -v -s runfold.tpr
```

Take a picture of the last conformation and analyze the trajectory as in the previous case. If you have time try to fold a sequence of your interest.

6 References

- Ángel Piñeiro, Norma Díaz-Vergara, Eduardo Jardón-Valadez, Aned de Leon and Sergio Rodríguez-Morales. Applications of molecular dynamics simulations to protein systems. In Advances in Protein Physical Chemistry, 2008: ISBN: 978-81-7895-324-3.

- D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen and H. J. C. Berendsen, Gromacs User Manual version 3.3 (2005). `www.gromacs.org`.

- David van der Spoel. Gromacs exercises (2004). `http://www.csc.fi/english/research/sciences/chemistry/courses/gmx2004/exercises/index_html`.

- Erick Lindahl. Pre-Workshop Hands-On Tutorial. Gromacs Workshop 2007 CSC, Espoo/Helsinki, Finland. `https://extras.csc.fi/chem/courses/gmx2007/tutorial1/tutorial1.pdf`.