Assessing accuracy of machine learning-based protein structure prediction model with molecular dynamics stabilization simulations

Cajsa Malm, 1901791

Master's thesis in physics
Faculty of Science and Engineering
at Åbo Akademi University

Supervisors:
Prof. (Research). Anssi Laukkanen
Sr. Scientist. Antti Paajanen
Team Leader (Research). Antti Puisto
Prof. Ronald Österbacka

March 29, 2023

Abstract

The ability to synthetically create materials with desired properties has been greatly enhanced by advances in biotechnology. One important application of biotechnology is the design of synthetic proteins with the ability to fold in any configuration and predict their folding in advance. However, the connection between sequence, structure, and function is not yet fully understood, and there are endless combinations of amino acids that cannot be experimentally examined. Therefore, computational tools such as machine learning techniques are needed to guide material development.

This thesis evaluates the accuracy of machine learning-based protein structure predictions by subjecting the predicted structures to molecular dynamics simulations using Gromacs software. The focus of the study is on synthetically constructed perfect tandem repeat proteins, and the goal is to test the stability of the predicted structures. The RMSD metric, often used to compare the structural similarity of proteins during molecular dynamics simulations, has limitations in its interpretation. To address this, a new measure of structural similarity called ρ_{sc} is proposed and used to assess the stability of the proteins.

The results of the simulations show that many of the proteins generated by the machine learning model are unstable, with significant conformational changes observed. This suggests that the current model may not accurately predict the stability of all proteins. The predicted proteins also contained nonphysical structures with overlapping atoms. The study highlights the importance of combining machine learning approaches with other computational approaches to improve the accuracy of protein structure prediction.

In conclusion, this study provides insights into the limitations of current machine learning models for protein structure prediction, and suggests the need for further research to better understand the underlying reasons for the observed instability. These findings could lead to improvements in protein design and prediction, ultimately leading to the creation of more advanced and functional materials with desired properties.

Contents

Chapter 1

Introduction

As technology continues to advance, the ability to create synthetic materials with precise and targeted properties is improving. Biotechnology, a field that enables the creation of customized protein sequences with tailored structures, is focused on designing synthetic proteins that can be folded in any desired configuration predicted in advance. Protein structure is widely acknowledged as being critical in determining biological interactions, and proteins can be used as building blocks to create materials with tailored properties to meet various needs.

One area where protein structure prediction has significant potential is in the food industry. Manufacturers can design protein-based foods with specific functional properties, such as texture, taste, and shelf life. This could lead to the development of new and innovative protein-based foods that are healthier and more sustainable than traditional options.

However, with endless combinations of amino acids, it is impossible to experimentally examine all of them. Therefore, computational tools such as machine learning are needed to guide material development in a faster and more cost-effective manner. These tools can suggest novel protein sequences that have not been tested before.

The main objective of my thesis is to test the accuracy of a protein structure prediction machine learning model on synthetically constructed perfect tandem repeat proteins. By running the proteins through molecular dynamics stability simulations using Gromacs software, my work will evaluate the structures predicted by the machine learning model, which will in turn give training data to the model.

Overall, the development of materials with specific properties has numerous applications in various industries, such as healthcare, electronics, and energy. By designing and synthesizing proteins with specific structures and properties using machine learning, this research has the potential to contribute to the development of new materials with targeted properties.

1.1 What is a protein and why is structure prediction important

Proteins are biomolecules essential for living matter. They are built of amino acids, of which around 500 exist in nature, but only 20 of them are naturally present in the body. Amino acids are built of one amino group (NH2), one carboxylic acid group (COOH), and one functional group (R), which determines the amino acid type. The shape and function of the protein are determined by these amino acids, which combine to create polypeptide chains through peptide bonds, i.e., bonds between the carboxylic acid group of one amino acid and the amino group of the neighboring atom [1]. Figure 1.1 shows the chemical composition of an amino acid.

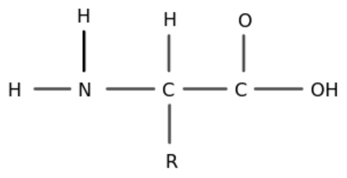


Figure 1.1: Chemical formula of amino acid.

The most important characteristics of proteins is their ability to fold into specific three-dimensional structures, e.g., their native conformation. The folding is determined by the sequence of amino acids and the physical and chemical properties of the surrounding environment. Proteins fold in a highly specific and efficient manner, driven by the formation of non-covalent interactions between different parts of the polypeptide chain [2]. Figure 1.2 shows a machine learning-predicted protein.

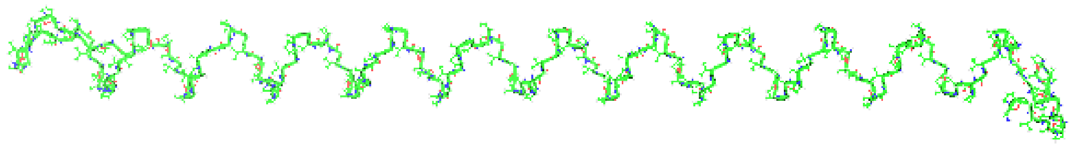


Figure 1.2: Machine learning predicted protein.

The structure of a protein determines its function and stability. Many proteins have multiple functions and therefore have evolved to adopt different conformations. For example, enzymes are proteins that catalyze chemical reactions and their active site is formed by the specific folding of the polypeptide chain. Similarly, structural proteins such as actin and myosin provide mechanical support and movement to cells and tissues [3].

Proper protein folding is crucial for the functioning of the protein, if proteins fail to fold into their functional structure, they can become pathological or lose their function all-together. Misfolded proteins in the body can cause a wide range of diseases, including Alzheimer's, Huntington's, and cystic fibrosis. Understanding the mechanisms of protein folding helps in the development of treatments for these diseases [4].

Predicting protein structure is essential for understanding the relationship between protein structure and function, as well as for the development of new therapeutics. Machine learning models have been developed to predict protein structures with high accuracy, providing insights into how proteins work and aiding in the design of drugs that target specific proteins. This field is rapidly expanding and has the potential to revolutionize our understanding of biochemistry and biomedical research [5].

1.2 Machine learning in biology

Machine learning is rapidly becoming a powerful tool in various fields of biology, including protein structure prediction [6]. The ability to accurately predict protein structures is crucial for understanding their functions and potential applications in medicine and biotechnology. The Critical Assessment of Structure Prediction (CASP) competition, which has been held every two years since 1994, serves as a benchmark for evaluating the progress and performance of different protein structure prediction methods [7].

AlphaFold2, developed by DeepMind Technologies, was the clear winner of the 2020 CASP competition. This machine learning model was trained on a large dataset of experimentally determined protein structures and is able to predict the structure of proteins with high accuracy. It has also been used to predict structures for the human proteome and other organic proteins. AlphaFold2 is better at predicting side chains and identifying evolutionarily connected proteins, which can improve the predictions [8].

One of the key advantages of machine learning algorithms like AlphaFold is that they can reduce human biases and lead to new discoveries that may not be logical to the human mind. However, a limitation is that these algorithms rely on familiar data, so there is always a possibility that the input data is too different from the existing data for the model to know how to process it [9].

While AlphaFold and other machine learning models can be very helpful in predicting initial protein structures, they still have large deviations compared to experimentally determined structures and cannot currently replace experimental methods. There is evidence to suggest that some machine learning-predicted protein structures may not be stable during molecular dynamics simulations. Despite this limitation, machine learning in biology is a rapidly growing field with great potential for future advancements [10].

The properties of proteins can be linked to the structure and order of the amino acids. In sequence databases, it is possible to obtain a clear view of the function of different amino acid sequences. It would still be beneficial to create a machine learning model that can predict protein folding and stability by just looking at the pattern of the amino acid, without any additional information. Many researchers are curious about how much of a sequence must be preserved in order to maintain functionality. Sometimes all that is required is the stability of the residues making up the active site. Research also suggests that the number of different protein folds existing in nature is limited; this knowledge is undoubtedly important in the field of protein research and modeling. One question is whether structural similarity alone can be proof of protein functionality. [11].

1.3 Limitations in machine learning-based protein structure prediction

One of the main limitations of machine learning-based protein structure predictors is prediction of regions of the protein that are flexible or have multiple conformations. These regions can include loops, disordered regions and flexible domains. The structures of these regions are important for the function of the protein and their accurate prediction is crucial for understanding the protein's behavior [12].

Another limitation is that the models struggle to predict interactions between domains. Proteins often have multiple domains that perform different functions and these interactions can play a crucial role in the proteins activity. Additionally, these models are trained on a limited amount of data, which can lead to overfitting and the models may not generalize well to unseen proteins [13].

It has been shown that including experimental information, such as electron density maps and distances between residues, can improve the accuracy of predicting disordered regions in proteins. The AlphaFold predicted protein structures have been compared to density maps of crystal structures and found to have both global and local deviations, e.g., inaccurately predicted domain orientations, backbone and side chains [14].

Despite these limitations, machine learning-based protein structure predictors have improved the field of protein structure prediction, and researchers are constantly working to improve their accuracy and generalizability. This includes the incorporation of experimental data, such as electron density maps, and the use of more diverse training sets. Another approach is to use multiple models, such as combining different machine learning techniques, or using a consensus of predictions from multiple models [15].

The accuracy of AlphaFold has been determined based on the confidence interval of atomic positional changes. The high accuracy of AlphaFold accounts for local atomic precision, meaning that it can predict the precise location of individual atoms in a protein structure with a high degree of accuracy. However, when researchers looked closer at these high-accuracy predictions, they found that they still poorly predicted certain regions of the protein, such as flexible or multi-conformational regions. Studies have shown that AlphaFold can only predict 40 percent of the human proteome accurately [16].

Though the structures are overall well predicted by machine learning-based protein structure predictors, experimental data could help to improve predictions on a more detailed level. For example, using experimental data such as electron density maps, X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and electron microscopy can provide a more accurate understanding of protein structure and function [14].

An iterative correction process can be used to refine and correct the predictions made by machine learning models, improving the accuracy of protein structure prediction. This approach incorporates experimental data to correct errors on a local level, leading to improved accuracy in other regions of the protein. The incorporation of experimental data and iterative correction processes can provide a better understanding of the dynamic nature of proteins and the complexity of their interactions. [17].

Chapter 2

Theoretical basis for molecular dynamics simulations

Molecular dynamics (MD) is a computer simulation technique used to study the movement of atoms in a system based on Newton's equations of motion [18]. In MD simulations, atoms are treated as point particles and interatomic interactions are described with analytical potential functions. Interactions influencing a molecular system include van der Waals, Coulombic, and hydrogen bonding, among others. The forces acting on individual atoms are derived from these potential functions [19].

Newton's laws of motion state that the second derivative of an objects position with respect to time is directly proportional to the net force acting on the object over its mass, i.e:

$$\frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{d}t^2} = \frac{\mathbf{F}_i}{m_i}, \tag{2.1}$$

Equivalently, the velocity is:

$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i; \quad \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \frac{\mathbf{F}_i}{m_i}. \tag{2.2}$$

The forces acting on a single atom in the system is based on its specific interactions with other atoms and external potentials. The updated positions can be determined by numerically integrating the net force at each time step. By repeating this process, the trajectory of the atoms over time is obtained.

MD simulations can provide valuable insights into protein folding, but it's worth noting that these simulations have limitations and are mere approximations of the real dynamics. The accuracy of the results can be affected by factors such as the size of the simulation box, the choice of force field parameters, and the simulation time. While MD simulations can be a powerful tool in the study of a proteins behavior, the results should be interpreted with caution and their limitations should be taken into account in the design [20].

During MD simulations, proteins often experience minor conformational changes as they are naturally flexible molecules. Thus, it is normal for protein structures to not remain completely stable throughout the simulations. It is common to observe some structural fluctuations, caused by various factors such as thermal motions, interactions with solvent molecules, and interactions with other molecules in the system [21].

However, the overall shape and stability of the protein molecule should be preserved during the simulation, and it should not undergo any large-scale unfolding or major structural rearrangements. The protein structure should remain within a reasonable range of conformations that are consistent with its native state. Therefore, while some degree of structural fluctuations is expected during MD simulations, the stability of the protein structure should be monitored and assessed to ensure that it remains within acceptable limits [22].

2.1 Dihedral angles

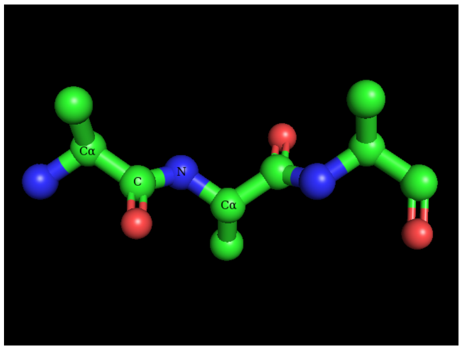
One critical step in setting up an MD simulation is selecting an initial structure for the system. The initial structure determines the starting conformation of the system and can greatly affect the behavior of the simulation. Therefore, accurately predicting the initial structure is crucial for obtaining meaningful results. Predicting dihedral angles with machine learning models can provide accurate estimates of the initial structure of a molecule, allowing for more efficient and reliable simulations.

Moreover, dihedral angle prediction using machine learning models can be particularly useful for simulating large and complex systems, where manual prediction of dihedral angles may be impractical or time-consuming. The use of machine learning models to predict dihedral angles can greatly benefit the setup of MD simulations..

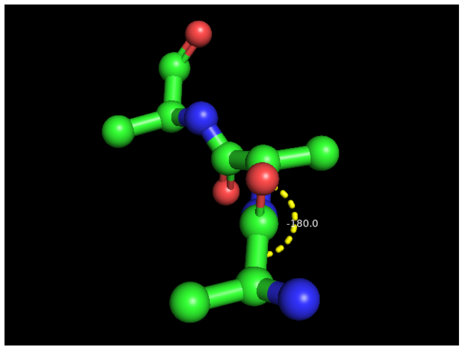
Dihedral angles provide data that can be utilized in molecular dynamics simulations to examine change in protein conformation. These angles are computed, by moving along the backbone of the peptide, which refers to the repeating sequence of N-C$_\alpha$-C atoms. The C$_\alpha$ is the atom that is connected to the functional group. The angles are determined by considering the positions of four following atoms along the backbone. Moving along the chain, all bond angles are determined. Dihedral angles provide a more thorough understanding of the internal and overall motion of the molecule. This knowledge is essential for comprehending the behavior of biomolecules in a changing environment. The bond length and angles remain fairly consistent, making the analysis of dihedral angles a more trustworthy method than merely measuring the coordinates of the atoms [23].

In figure 2.1b, the dihedral angle ω is determined. The four atoms being observed are labeled in figure 2.1a. The bond between the carbon atom (C) and the nitrogen atom (N), i.e., the peptide bond axis, makes up the backbone. The dihedral angle is defined by the angle between the first and third bond, which is shown in 2.1b.

The dihedral angle ϕ is defined by the rotation around the $N - C_\alpha$ bond axis and the dihedral angle ψ is the rotation around the $C_\alpha - C$ bond axis. A rotation clockwise, moving from left to right along the peptide, gives a positive angle, and a rotation to the left gives a negative angle. Figure 2.2a show the dihedral angle ϕ and figure 2.2b show the angle ψ.

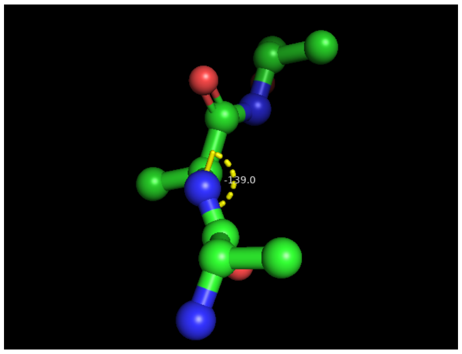


(a) Selected atoms: $C_\alpha - C - N - C_\alpha$.

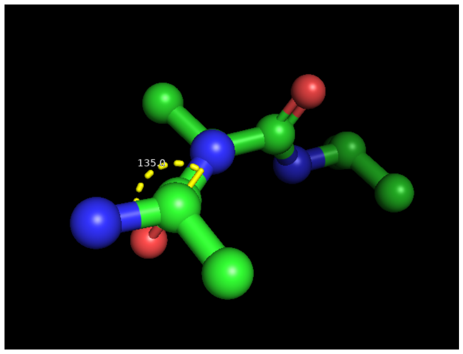


(b) Dihedral angle, ω.

Figure 2.1: Molecular model used to demonstrate the dihedral angles of the selected first four atoms (a) and the dihedral angle ω (b).



(a) Dihedral angle, ϕ.



(b) Dihedral angle, ψ.

Figure 2.2: Dihedral angles ϕ and ψ.

2.2 Forces

Molecular dynamics simulations use a force field to handle atomic interactions, which are mathematical functions that describe the potential energy of a system based on the positions and velocities of the atoms. The force field is a collection of potential energy functions that enable the calculation of forces between atoms based on the specific interactions within the system. These functions captures essential interactions between atoms and molecules, such as chemical bonds and electrostatic forces.

The net force field on an atom can be used to determine the motion and behavior of the system over time. The force acting on a single atom in a system can be expressed as:

$$\mathbf{F}_i = -\frac{\partial V}{\partial \mathbf{r}_i}, \tag{2.3}$$

where $\mathbf{r}_i$ is the position vector of the ith atom, and V is the potential energy of the system. The force acting on a single atom can also be expressed as the sum of all pairwise forces between the atoms as:

$$\mathbf{F}_i = \sum_j \mathbf{F}_{ij}, \tag{2.4}$$

where $\mathbf{F}_{ij}$ is the force between the ith and jth atoms. This equation is known as Newton's third law of motion, which states that every action has an equal and opposite reaction. Therefore, the force acting on one atom due to all the other atoms in the system is equal and opposite to the force acting on the other atoms due to the ith atom.

In MD simulations, the interactions between atoms or molecules are described by the chosen force field, which includes mathematical functions and parameters that define how atoms interact with each other. By using a force field, molecular dynamics simulations can predict the behavior of molecules and materials under different conditions. The accuracy of the simulation results depends on the quality of the force field parameters, which is related to the software used for the simulations. The force field handles all of the specific interactions between atoms, including bonded and non-bonded interactions, and determines the potential energy of the system. The atomic interactions will be explained in detail in chapter 2.3.

2.3 Potential functions in MD simulations

By defining the pairwise potential energies between the atoms, the forces can be derived using the negative gradient of the potential energy with respect to the atomic positions, which was stated in equation 2.3. The net force on individual atoms is obtained by summing up all forces influencing the atoms. In this way, MD simulations can provide insight into the structural and dynamic properties of molecular systems. The potential energy of a molecular system can equally be calculated as the sum of the potential energies of all the individual atoms comprising the system:

$$V\left(\mathbf{r}_1, \ldots \mathbf{r}_N\right) = \sum_{i<j} V_{ij}\left(\mathbf{r}_{ij}\right) \tag{2.5}$$

The interactions between particles are centro-symmetric because the potential only depends on the scalar distance between them. We typically divide the interactions into two general classes, bonded and nonbonded interactions. The bonded interactions are strong and involve the sharing or transfer of electrons between atoms in a molecule, while the nonbonded interactions are weaker and involve forces between molecules or within a molecule.

The nonbonded interactions include a repulsion term, a dispersion term, and a Coulomb term. The repulsion and dispersion term can be represented by either the Lennard-Jones or Buckingham potential. Additionally, atoms with partial charges interact through the Coulomb term. Nonbonded interactions are pair additive; i.e., the total potential is the sum of the potential between pairs and can be written as:

Overall, the potentials used in MD simulations are crucial for determining the behavior of molecular systems and the accuracy of the simulation results. The selection of the appropriate potentials is therefore an important consideration in the development and implementation of MD simulations.

2.3.1 Potential energy functions for nonbonded interactions

In molecular dynamics, nonbonded interactions refer to the interactions between atoms that are not covalently bonded to each other, i.e., van der Waals forces, electrostatic interactions, and repulsive forces [24]. These interactions are usually modeled using empirical potential functions, such as the Lennard-Jones potential or the Coulomb potential, which are used to calculate the energy and force between atoms in a simulation. Nonbonded interactions play a crucial role in determining the structure, stability, and behavior of molecular systems in MD simulations.

The Lennard-Jones potential can be separated into a repulsion term and a dispersion term as follows:

$$V_{LJ}\left(r_{ij}\right) = \frac{C_{ij}^{(12)}}{{r_{ij}}^{12}} - \frac{C_{ij}^{(6)}}{{r_{ij}}^{6}}, \tag{2.6}$$

where $C_{ij}^{(12)}$ and $C_{ij}^{(6)}$ account for different atom type pairs.

The Buckingham potential is a combination of an exponential term and an inverse power term, allowing it to represent both attractive and repulsive forces between atoms, which is expressed:

$$V_{bh}\left(r_{ij}\right) = A_{ij}\exp\left(-B_{ij}r_{ij}\right) - \frac{C_{ij}}{r_{ij}} \tag{2.7}$$

where A_{ij}, B_{ij}, and C_{ij} are parameters specific to the particular atom pair being modeled. These parameters can be determined from experimental data or from fitting to a set of data. The exponential term, $A_{ij}\exp(-B_{ij}r_{ij})$, represents the attractive forces between atoms, while the inverse power term, $-C_{ij}/r_{ij}$, represents the repulsive forces.

When two atoms with polar bonds are close together in the simulation, their partial charges create an electrostatic interaction between them. This electrostatic interaction can be attractive or repulsive, depending on the sign of the partial charges, and it can influence the behavior and properties of the system being studied. These interactions are described with the Coulomb potential as:

$$V_c\left(r_{ij}\right) = f\frac{q_i q_j}{\varepsilon_r r_{ij}}, \tag{2.8}$$

where f is the screening factor ($\frac{1}{4\pi\varepsilon_0} = 138.935458$), q_i and q_j are the charges of the two atoms being considered, and ε_r is the relative permittivity of the medium in which the atoms are located. The Coulomb potential models the interaction between two charged particles, and the magnitude of the potential depends on the distance between the particles and the magnitude of their charges. This potential is essential in describing the interactions between atoms in systems with polar bonds or charged species. The interactions between polar bonds are typically described using the partial charges assigned to atoms in the selected force field.

2.3.2 Potential energy functions for bonded interactions

Bonded interactions in MD simulations refer to the interactions between atoms that are covalently bonded or connected by other strong chemical bonds. These interactions lead to bond stretching, bond bending, and torsion angle interactions and they are usually modeled using potential energy functions, such as the Harmonic bond potential or Morse potential. These interactions are critical for accurate predictions of molecular properties such as reactivity, stability, and thermodynamics.

The Harmonic bond potential is a simple quadratic equation used to model bond stretching interactions, represented as:

$$V_b\left(r_{ij}\right) = \frac{1}{2}k_{ij}^b\left(r_{ij} - b_{ij}\right)^2, \tag{2.9}$$

The equation describes the bond potential energy (V_b) between atoms i and j. It is characterized by a spring constant (k_{ij}^b) that describes the bond's strength and a bond length (r_{ij}) that represents the distance between atoms i and j. The equilibrium bond length (b_{ij}) represents the bond length when the bond is at its most stable state. Figure 2.3 shows an illustration of bond stretching.

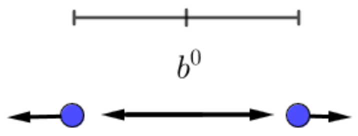


Figure 2.3: Harmonic bond stretching, at equilibrium

The Morse potential, in comparison to the harmonic bond potential, contains an asymmetrical potential well and generates no force at an infinitely distant separation. This is a way of modelling bond stretching interactions using a more complex non-linear equation, represented as:

$$V_{\text{Morse}}\left(r_{ij}\right) = D_{ij}\left[1 - \exp\left(-\beta_{ij}\left(r_{ij} - b_{ij}\right)\right)\right]^2, \tag{2.10}$$

where V_{Morse} is the potential energy of a bond between atoms i and j in a molecule. The well depth D_{ij} describes the amount of energy required to dissociate the bond between atoms i and j. The bond width parameter β_{ij} describes the steepness of the bond potential energy, and the equilibrium bond length b_{ij} is the length at which the bond potential energy is at a minimum.

The bond-angle vibration between a triplet of atoms is modeled using the harmonic angle potential, which is represented by the following equation:

$$V_a\left(\theta_{ijk}\right) = \frac{1}{2}k_{ijk}^{\theta}\left(\theta_{ijk} - \theta_{ijk}^0\right)^2, \tag{2.11}$$

where, V_a is the potential energy, k_{ijk}^{θ} is the spring constant, θ_{ijk} is the bond angle, and θ_{ijk}^0 is the equilibrium bond angle. The harmonic angle potential is used to model the bond-angle vibration between a triplet of atoms, with the bond-angle vibration represented as a harmonic potential, similar to the bond stretching, shown in figure 2.4:

These are just a few examples of the equations used to model bonded interactions in MD simulations. The specific form of the equation will depend on the force field being used and the specific interaction being modeled, which will be further explored in the chapter about numerical implementation.

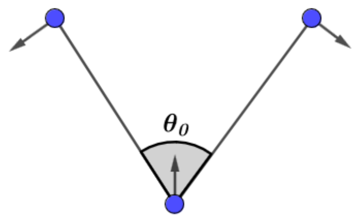


Figure 2.4: Bond-angle vibration

2.4 Initial energy minimization and equilibration

MD energy minimization and equilibration are crucial steps in preparing a molecular dynamics simulation. By iteratively adjusting the positions of the atoms in the system, energy minimization can help to eliminate any unfavorable contacts between atoms and achieve a stable initial configuration with low potential energy. Equilibration is the process of bringing the system to an equilibrium state. Equilibration is achieved in two steps, by first keeping the temperature and volume constant (NVT) secondly the temperature and pressure constant (NPT).

In molecular dynamics, the total energy of a system is comprised of both potential energy and kinetic energy. The potential energy have already been discussed in chapter 2.3 and the kinetic energy of the system follows the formula of kinetic energy:

$$E_{kin} = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2, \tag{2.12}$$

from which the temperature can be calculated with the formula $\frac{1}{2}N_{\mathrm{df}}kT = E_{\mathrm{kin}}$, where N_{df} is the degrees of freedom, which can be received from $N_{\mathrm{df}} = 3N - N_c - N_{\mathrm{com}}$.

The variable N_c represents the total number of constraints applicable to a system. In normal conditions, N_{com} is derived by subtracting three from N_c, while in a vacuum or unconstrained environment, N_{com} is calculated by subtracting six from N_c. This means that N_{com} reflects the effective number of constraints acting on the system, taking into account the specific conditions under which it is operating.

Energy minimization is important in determining the proper molecular arrangement in space. The potential energy of a molecule contains different components like stretching, bending, and torsion. Energy minimization operations can find a local minimum energy value, but it may not be the most stable conformer (global energy minimum) [25]. One method to determine the local energy minima is the particle-mesh Ewald method, which is presented in chapter 3.1. Figure 2.5 displays an energy minimization curve that is approaching a plateau, but has not yet fully converged.

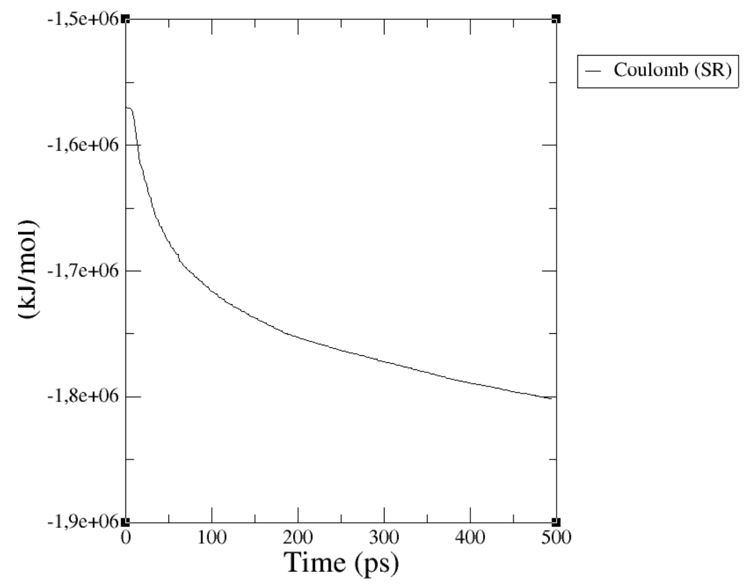


Figure 2.5: Energy minimization curve.

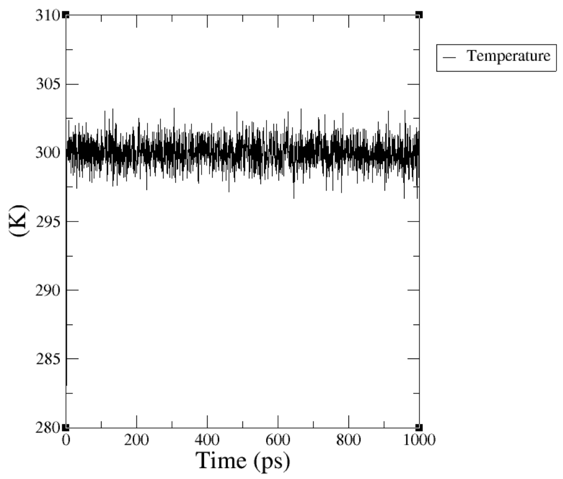
Equilibration is a process used in molecular dynamics simulations to bring the system to an appropriate temperature and pressure. This allows the simulation to accurately represent the behavior of the system at these conditions. During equilibration, the kinetic energy of the system will increase until it reaches an equilibrium value determined by the temperature and pressure, which is maintained through the balance between kinetic and potential energy.

There are two commonly used ensembles in molecular dynamics simulations: the NVT and NPT ensembles. The NVT (Constant Number of Particles, Constant Volume, Constant Temperature) ensemble is a statistical ensemble in which the number of particles in the system, the volume of the system, and the temperature are kept constant. A temperature-control algorithm, which is explained in chapter such as a thermostat, is used to maintain the ensemble. The equation for this ensemble is given by:

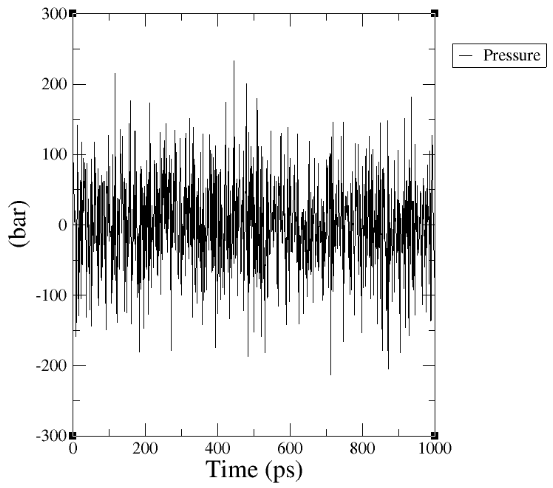
$$pV = Nk_{\mathrm{B}}T, \tag{2.13}$$

where p = Pressure, V = Volume, N = Number of Particles, k_{B} = Boltzmann's constant, and T = Temperature.

The NPT (Constant Number of Particles, Constant Pressure, Constant Temperature) ensemble is a statistical ensemble in which the number of particles in the system, the pressure of the system, and the temperature are kept constant. This ensemble is maintained using a pressure-control algorithm, such as a barostat. The procedures for determining the NVT and NPT configurations are described in more detail in chapters 3.3 and 3.4. The graphs in Figure 2.6 displays the behavior of both NVT and NPT systems, with equilibrium being achieved rapidly.



(a) Temperature equilibration curve.



(b) Pressure equilibration curve.

Figure 2.6: NVT and NPT curves.

2.5 MD parameters

The extraction of structural parameters is vital in MD simulations, enabling characterization of macromolecules' conformational properties. They provide a reference framework for comparing proteins and monitoring changes in their three-dimensional structure over time. By accurately defining these parameters, researchers can quantitatively analyze the behavior of proteins and their interactions with other molecules, enabling a deeper understanding of key biological processes. The reliable determination of these parameters is critical to obtaining informative and trustworthy results from MD simulations, as they form the foundation for the interpretation of simulation outcomes.

The Root Mean Square Deviation (RMSD) is a widely used method to measure the similarity between two sets of atomic coordinates. It calculates the square root of the average sum of the distances between each pair of equivalent atoms. However, the RMSD can be misleading because it is highly sensitive to the amplitude of errors. For example, it may give a large RMSD value even when two structures are very similar except for a small difference in a single loop or flexible terminus. This can affect the accuracy of protein structure predictions based on the RMSD [26]. Therefore, the RMSD should be used with caution and in conjunction with other measures to accurately assess the similarity between protein structures. The RMSD is calculated using the formula:

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \tag{2.14}$$

where n is the number of atoms being compared between the two structures, and d_i is the distance between atom i in the first structure and atom i in the second structure.

The radius of gyration is a measure of the protein's compactness, indicating the distribution of the mass of the protein with respect to its center of mass. It is defined as the root mean square distance of the atoms from the center of mass. The formula for calculating the radius of gyration involves the masses and positions of all atoms in the protein. A protein with a smaller radius of gyration is more compact than one with a larger radius of gyration. Radius of gyration can be expressed with the formula:

$$R_{\mathrm{gyr}}^2 = \frac{1}{M} \sum_{i=1}^{N} m_i \left(\mathbf{r}_i - \mathbf{R}\right)^2 \tag{2.15}$$

where $M = \sum_{i=1}^{N} m_i$ is the total mass and $\mathbf{R} = N^{-1} \sum_{i=1}^{N} \mathbf{r}_i$ is the center of mass of the protein. N is the number of atoms in the peptide. The changes in radius of gyration over the simulation time correlate to conformational variations of the protein [27].

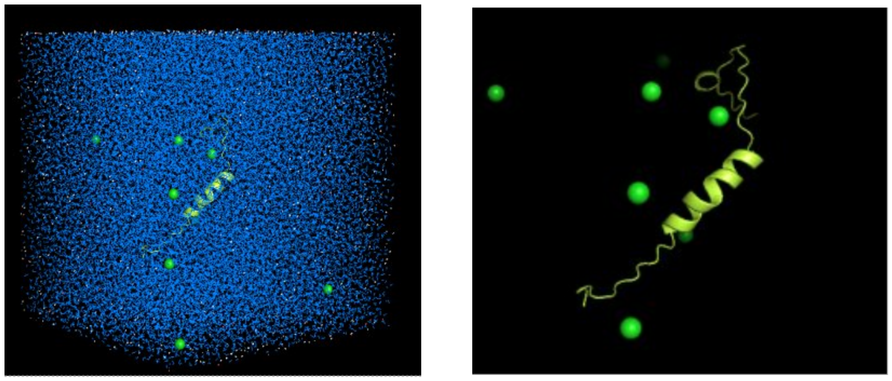
The changes in the radius of gyration during molecular dynamics simulations can be used to assess the conformational flexibility of the protein. Proteins that undergo large conformational changes during simulation generally exhibit larger changes in radius of gyration. Additionally, changes in the radius of gyration can be used to study protein folding, binding, and stability. In general, the radius of gyration is a useful metric for analyzing protein structure and dynamics.

Chapter 3

Numerical Implementation and GROMACS Workflow

The GROMACS workflow is a set of steps used to simulate molecular dynamics of biomolecules such as proteins. The process typically includes the following steps: preprocessing, energy minimization, solvent equilibration, production run and analysis.

Firstly, the protein is put in a box of water, and ions (Na+ and Cl-) are added to neutralize the system . This can be seen in Figure 3.1.



(a) The system with water. (b) The system without water.

Figure 3.1: The protein contained in a box with added water and ions.

Secondly, the peptide is equilibrated with respect to temperature (NVT) and pressure (NPT). During the production run, the coordinates and velocities of the atoms are saved for further analysis. Finally, the saved data is used to calculate various properties of the system and other tools are used to analyze the simulation.

Before the simulation can start, all initial parameters must be determined. The time step is updated using the leapfrog algorithm (presented in chapter 3.2.1), which requires known coordinate parameters at $t = t_0$ and $t = t_0 - \frac{1}{2}\Delta t$. If the velocities are unknown, they are estimated by GROMACS at a given temperature with the Boltzmann-Maxwell distribution:

$$p\left(v_i\right) = \sqrt{\frac{m_i}{2\pi kT}} \exp\left(-\frac{m_i v_i^2}{2kT}\right) \tag{3.1}$$

Only the forces between nonbonded atoms separated by a distance smaller than the cut-off radius R_c are considered. GROMACS creates pair lists of all relevant nonbonded pairs, including a displacement vector for atom i along with the neighbor list of j atoms in its R_c range. The machine learning model predicts dihedral angles from amino acid sequences, which are then processed and converted into a file format compatible with GROMACS. The GROMACS software uses various algorithms and techniques to simulate the motion of atoms and molecules based on the provided sequence data, with the aim of generating accurate results.

The numerical implementation of molecular dynamics simulations is at the centre in the study of molecular systems, and this chapter focuses on the algorithms and methods involved in these simulations. It covers the processing of the proteins, as well as the algorithms used by GROMACS. The choice of force fields and algorithms is crucial to achieving accurate results, and this will be discussed in detail in this chapter.

One key implementation in large system simulations is periodic boundary conditions (PBC) [28]. PBC is used to prevent boundary effects in simulations, where the protein does not stay in the same place and parts of the protein may move past the boundaries of the simulation box. To solve this problem, PBC creates identical boxes surrounded simulation box. This allows the particles in the system to appear to move dynamically between the boxes, ensuring that the protein does not move past the boundaries of the simulation box.

Under PBC, the system is effectively infinite, as there are an infinite number of periodic images of the original cubic box. This means that the interaction between particles in the system extends beyond the boundaries of the original cubic box and into its periodic images. This is illustrated in Figure 3.3, where the simulation box is shown in the center, surrounded by eight translational boxes.

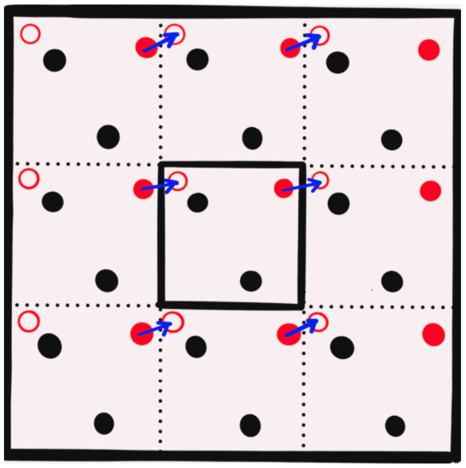


Figure 3.2: Visualization of periodic boundary conditions.

The vectors of the three-dimensional simulation box must satisfy the following boundary conditions:

$$a_y = a_z = b_z = 0 \tag{3.2}$$

$$a_x > 0, \quad b_y > 0, \quad c_z > 0 \tag{3.3}$$

$$|b_x| \leq \frac{1}{2}a_x, \quad |c_x| \leq \frac{1}{2}a_x, \quad |c_y| \leq \frac{1}{2}b_y \tag{3.4}$$

The conditions in equation 3.2 and $R_c < \frac{1}{2}\min(a_x, b_y, c_z)$ give the translational images inside the R_c cutoff, expressed with the r_{ij} vector:

$$\begin{aligned} \mathbf{r}''' &= \mathbf{r}_j - \mathbf{r}_i \\ \mathbf{r}'' &= \mathbf{r}''' - \mathbf{c} * \mathrm{round}\left(r'''_z / c_z\right) \\ \mathbf{r}' &= \mathbf{r}'' - \mathbf{b} * \mathrm{round}\left(r''_y / b_y\right) \\ \mathbf{r}_{ij} &= \mathbf{r}' - \mathbf{a} * \mathrm{round}\left(r'_x / a_x\right) \end{aligned} \tag{3.5}$$

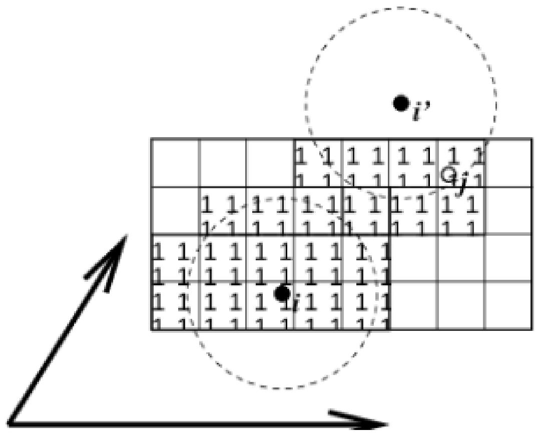


Figure 3.3: Grid search with box vectors.

3.1 The Particle-Mesh Ewald method

The Particle-Mesh Ewald (PME) method is a technique used to accurately calculate the Coulombic interactions in large biological systems. It combines real-space and Fourier space calculations to determine the potential energy of the system at each time step of a molecular dynamics simulation.

In PME, the electrostatic interactions are divided into two rapidly converging potential sums, a near-field term, which is calculated in real space using a direct summation, and a far-field term, which is calculated in Fourier space using the fast Fourier transform [29]. The near-field term is used for close-range interactions, while the far-field term is used for long-range interactions [30]. The Ewald sum can be written as:

$$U_{\text{Ewald}} = U^r + U^m + U^o, \tag{3.6}$$

where U^r is the real-space sum, U^m is the reciprocal (i.e. Fourier) sum and U^o is a constant. The real-space and reciprocal-space terms are defined by:

$$U^r = \frac{1}{2} \sum_{i,j}^{N'} \sum_{\boldsymbol{n}} q_i q_j \frac{\operatorname{erfc}(\alpha r_{ij,n})}{r_{ij,n}}, \tag{3.7}$$

and

$$U^m = \frac{1}{2\pi V} \sum_{i,j}^{N} q_i q_j \sum_{m=0} \frac{\exp\left(-(\pi \boldsymbol{m}/\boldsymbol{\alpha})^2 + 2\pi \mathrm{i} \boldsymbol{m} \cdot (\boldsymbol{r}_i - \boldsymbol{r}_j)\right)}{\boldsymbol{m}^2}, \tag{3.8}$$

where V is the volume of the simulation box, m = (l,j,k) is a reciprocal-space lattice vector, which is used to calculate the far-field term of the electrostatic interactions in Fourier space and n is the cell coordinate vector relative to the original cell, which is located at $\mathbf{n} = (0,0,0)$. The quantity $\mathrm{erfc}(\alpha r_{ij,n})$ is the complementary error function of $\alpha r_{ij,n}$, where α is a parameter related to the screening length of the Coulomb interaction between the charged particles.

The Coulomb energy of a system with N particles in a cubic box of size L and its infinite replicas under periodic boundary conditions (PBC) is calculated as follows:

$$U = \frac{1}{2} \sum_{n}' \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{r_{ij,n}} \tag{3.9}$$

In this expression, U is the total Coulomb energy of the system, q_i and q_j are the charges of particles i and j, respectively, and $r_{ij,n}$ is the distance between particles i and j in the n-th periodic image of the system. The sum includes all periodic images of the system, and the prime excludes the $n = 0$ image to avoid double-counting interactions between particles in the original box.

When considering a charge-neutral system $\sum_{i=1}^{N} q_i = 0$, the potential energy in equation 3.9 can be Ewald transformed into:

$$\sum_{n} \frac{1}{|n|} F(n) + \sum_{m} \frac{1}{|m|} (1 - F(m)) \tag{3.10}$$

The first term quickly decays as $\boldsymbol{n} \to \infty$, and the second term is a smooth function, with a fast-converging Fourier transform. The physical meaning behind the decomposition of the lattice sum can be interpreted as where each point charge in the system is considered to be surrounded by an equal and opposite Gaussian charge distribution with charge density:

$$\rho_i(\boldsymbol{r}) = q_i \alpha^3 \exp\left(-\alpha^2 r^2\right) / \sqrt{\pi^3} \tag{3.11}$$

The PME method is utilized during the energy minimization step, treating atoms as having long-range electrostatic interactions. The selection of this interaction i types beneficial for large biopolymers, because it ignores interactions with atoms far apart, reducing the computational load [31].

PME has the advantage of being scalable and having a computational cost that increases linearly with the size of the system. It also provides accurate results, even in systems with strong electrostatic interactions. Overall, this method is widely used in molecular dynamics simulations and is an essential component of many simulation software packages, including GROMACS.

3.2 Integrators

In molecular dynamics simulations, integrators are used to numerically solve the equations of motion that describe the behavior of the atoms and molecules in the system. These equations of motion are based on classical mechanics and describe how the position, velocity, and acceleration of each particle in the system change over time in response to the forces acting upon them.

Integrators are necessary because these equations of motion cannot be solved analytically for most molecular systems of interest. Instead, numerical methods are used to approximate the solutions. Different integrators use different numerical algorithms and techniques to solve these equations and advance the simulation time step-by-step.

GROMACS provides a range of integrators for simulating molecular dynamics, including basic integrators such as the velocity Verlet and leapfrog, as well as advanced integrators for more complex simulations. The choice of integrator will depend on the specific requirements of the simulation, such as the time step size, accuracy, and the type of forces involved. This chapter will present a comparison between the leapfrog and Verlet integrators, including their performance with the Trotter decomposition.

3.2.1 The leapfrog integrator

The leapfrog algorithm is used as the default MD integrator in GROMACS. Figure 3.4 shows how **r** and **v** jump over each other's time-steps, where **r** corresponds to time t and **v** with time $t - \frac{1}{2}\Delta t$.

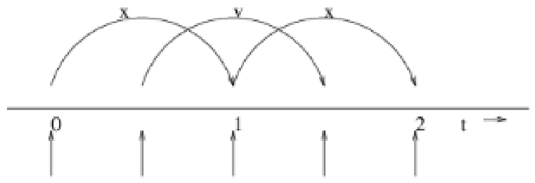


Figure 3.4: Leapfrog integrator.

The integration steps are expressed below:

$$\begin{aligned} \mathbf{v}\left(t + \tfrac{1}{2}\Delta t\right) &= \mathbf{v}\left(t - \tfrac{1}{2}\Delta t\right) + \tfrac{\Delta t}{m}\mathbf{F}(t) \\ \mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \Delta t \mathbf{v}\left(t + \tfrac{1}{2}\Delta t\right), \end{aligned} \tag{3.12}$$

and the updated position time-step has relation:

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \frac{1}{m}\mathbf{F}(t)\Delta t^2 + O\left(\Delta t^4\right) \tag{3.13}$$

3.2.2 Verlet integrator

The velocity Verlet integrator is useful for accurate integration with temperature and pressure coupling and can be expressed as:

$$\begin{aligned} \mathbf{v}\left(t + \frac{1}{2}\Delta t\right) &= \mathbf{v}(t) + \frac{\Delta t}{2m}\mathbf{F}(t) \\ \mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \Delta t \mathbf{v}\left(t + \frac{1}{2}\Delta t\right) \\ \mathbf{v}(t+\Delta t) &= \mathbf{v}\left(t + \frac{1}{2}\Delta t\right) + \frac{\Delta t}{2m}\mathbf{F}(t+\Delta t), \end{aligned} \tag{3.14}$$

which equals:

$$\begin{array}{rcl}\mathbf{r}(t+\Delta t) & = & \mathbf{r}(t)+\Delta t\mathbf{v}+\frac{\Delta t^2}{2m}\mathbf{F}(t) \\ \mathbf{v}(t+\Delta t) & = & \mathbf{v}(t)+\frac{\Delta t}{2m}[\mathbf{F}(t)+\mathbf{F}(t+\Delta t)].\end{array} \tag{3.15}$$

3.2.3 Trotter decomposition

In molecular dynamics simulations, the Trotter decomposition is used to approximate the time evolution operator of the system. The time evolution operator describes how the quantum state of the system changes over time, and is given by the exponential of the Hamiltonian operator multiplied by the time step:

$$U(t) = \exp(-iHt) \tag{3.16}$$

However, this exponential is difficult to compute exactly for many systems of interest. The Trotter decomposition provides a way to approximate this exponential as a product of exponentials that are easier to compute. Specifically, the Trotter decomposition involves splitting the Hamiltonian into two or more parts and applying each part to the quantum state for half of the time step. This process is repeated multiple times, with the time step being divided into smaller intervals, until the desired simulation time is reached.

The Trotter decomposition is not an exact method, but it can be used to obtain accurate results for many systems of interest. It is widely used in MD simulations of quantum systems, such as molecules and materials, and is an important tool for studying the properties and behavior of these systems.

The Trotter decomposition can also show the relations between the leapfrog integrator and Verlet velocity integrator. Coupled first order differential equations can be evaluated from $t = 0$ to t with the evolution operator:

$$\begin{array}{rcl}\Gamma(t) & = & \exp(iLt)\Gamma(0) \\ iL & = & \dot{\Gamma}\cdot\nabla_{\Gamma},\end{array} \tag{3.17}$$

where L is the Liouville operator and Γ is the multidimensional vector of the velocities and positions of the atoms. This short-time approximation can be used to evaluate the behavior of the system over small time intervals, and is useful in the context of molecular dynamics simulations.

$$\Gamma(t) = \prod_{i=1}^{P} \exp(iL\Delta t)\Gamma(0) \tag{3.18}$$

at $\Delta t = t/P$. The Liouville operator related to NVE dynamics (constant number of particles, volume, and energy) is:

$$iL = \sum_{i=1}^{N} \mathbf{v}_i \cdot \nabla_{\mathbf{r}_i} + \sum_{i=1}^{N} \frac{1}{m_i}\mathbf{F}\left(r_i\right) \cdot \nabla_{\mathbf{v}_i}, \tag{3.19}$$

which can be split into:

$$\begin{aligned} iL_1 = & \quad \sum_{i=1}^{N} \frac{1}{m_i}\mathbf{F}\left(r_i\right) \cdot \nabla_{\mathbf{v}_i} \\ iL_2 = & \quad \sum_{i=1}^{N} \mathbf{v}_i \cdot \nabla_{\mathbf{r}_i}. \end{aligned} \tag{3.20}$$

The short-time approximation of the dynamics can be expressed as:

$$\exp(iL\Delta t) = \exp\left(iL_2\frac{1}{2}\Delta t\right)\exp\left(iL_1\Delta t\right)\exp\left(iL_2\frac{1}{2}\Delta t\right) + \mathcal{O}\left(\Delta t^3\right), \tag{3.21}$$

where Δt equals the full velocity Verlet time step and $\frac{1}{2}\Delta t$ the half step. For time $t = n\Delta t$ the dynamics approximation is equivalently:

$$\begin{aligned} \exp(iLn\Delta t) \approx & \quad \left(\exp\left(iL_2\tfrac{1}{2}\Delta t\right)\exp\left(iL_1\Delta t\right)\exp\left(iL_2\tfrac{1}{2}\Delta t\right)\right)^n \\ \approx & \quad \exp\left(iL_2\tfrac{1}{2}\Delta t\right)\left(\exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t\right)\right)^{n-1} \\ & \qquad \exp\left(iL_1\Delta t\right)\exp\left(iL_2\tfrac{1}{2}\Delta t\right) \end{aligned} \tag{3.22}$$

The leapfrog integrator starts with $\exp\left(iL_1\Delta t\right)$ instead of the half-step and ranges from $t - \frac{1}{2}\Delta t$ to $t + \frac{1}{2}\Delta t$, which gives:

$$\exp(iLn\Delta t) = \quad \exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t\right) + \mathcal{O}\left(\Delta t^3\right) \tag{3.23}$$

Subsequently, future times $t = n\Delta t$ follows:

$$\exp(iLn\Delta t) \approx \quad \left(\exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t\right)\right)^n \tag{3.24}$$

This shows that the resulting trajectories would be identical, except for the fact that they would be shifted in time due to the different starting points used by each method. The kinetic energy of the Verlet velocity algorithm is calculated from the velocity at t and is summed over all particles:

$$\begin{aligned} KE_{\text{full}}\left(t\right) = & \; \sum_i \left(\tfrac{1}{2m_i}\mathbf{v}_i(t)\right)^2 \\ = & \; \sum_i \tfrac{1}{2m_i}\left(\tfrac{1}{2}\mathbf{v}_i\left(t-\tfrac{1}{2}\Delta t\right)+\tfrac{1}{2}\mathbf{v}_i\left(t+\tfrac{1}{2}\Delta t\right)\right)^2 \end{aligned} \tag{3.25}$$

Using the leapfrog algorithm, the average kinetic energy at $t+\frac{1}{2}\Delta t$ and $t-\frac{1}{2}\Delta t$ can be calculated as:

$$KE_{\text{average}}\left(t\right) = \quad \sum_i \frac{1}{2m_i}\left(\frac{1}{2}\mathbf{v}_i\left(t-\frac{1}{2}\Delta t\right)^2+\frac{1}{2}\mathbf{v}_i\left(t+\frac{1}{2}\Delta t\right)^2\right) \tag{3.26}$$

Equations 3.25 and 3.26 show that even though the trajectories of the Verlet and leapfrog integrals are equal, the kinetic energies are not necessarily the same. GROMACS has also added a non-standard Verlet algorithm that averages the kinetic energies. The Verlet half-step-averaged kinetic energies are almost fully comparable to the leapfrog kinetic energy, if there is no temperature or pressure coupling. Still, the leapfrog integrator is preferred for large systems because it is the least computationally expansive and will be used in the simulations of this thesis.

3.3 Temperature coupling

Temperature coupling refers to a technique used in molecular dynamics simulations to regulate the temperature of a system. In molecular dynamics simulations, the temperature of the system can change due to various factors, such as collisions between molecules, and this can lead to fluctuations in the temperature that can affect the accuracy of the simulation. To address this issue, temperature coupling algorithms are employed to control the temperature of the system, ensuring that it remains at a constant temperature throughout the simulation.

GROMACS offers a diverse set of temperature coupling algorithms, including Berendsen, Nosé-Hoover, and the Andersen Thermostat. This chapter will introduce these algorithms, which are designed to facilitate energy exchange between the system and a thermal bath while continuously regulating the temperature to maintain a stable value.

3.3.1 Berendsen temperature coupling

The Berendsen temperature coupling calculates the temperature relative to an external heat bath of temperature T_0. The system temperature deviation is corrected according to:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{T_0 - T}{\tau} \tag{3.27}$$

T_0 is the desired temperature and T is the instantaneous temperature at a given time of the simulation. The heat flow is dependent on the velocity time-steps n_{TC} of each particle with a time-factor, which equals:

$$\lambda = \left[1 + \frac{n_{\mathrm{TC}}\Delta t}{\tau_T}\left\{\frac{T_0}{T\left(t - \frac{1}{2}\Delta t\right)} - 1\right\}\right]^{1/2} \tag{3.28}$$

In normal use λ is close to 1.0. The relationship between the time constant τ and the temperature coupling constant τ_T is:

$$\tau = 2C_V\tau_T/N_{df}k, \tag{3.29}$$

where C_V is the heat capacity of the system, k is Boltzmann's constant, and N_{df} is the degree of freedom of the system. The kinetic energy is modified for every step to:

$$\Delta E_k = (\lambda - 1)^2 E_k \tag{3.30}$$

3.3.2 Nosé-Hoover temperature coupling

The Berendsen weak-coupling algorithm quickly relaxes the system to the target temperature, but an additional algorithm is needed to find the right canonical ensemble. The Nosé-Hoover temperature coupling algorithm includes a friction term in the equation of motion. The equation of motion of each particle compared to 2.1 is:

$$\frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{d}t^2} = \frac{\mathbf{F}_i}{m_i} - \frac{p_\xi}{Q}\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}, \tag{3.31}$$

where p_ξ is the momentum of the heat bath variable ξ. Q is the mass parameter of the reservoir and determines the coupling strength. The equation of motion of the heat bath variable equals:

$$\frac{\mathrm{d}p_\xi}{\mathrm{d}t} = (T - T_0) \tag{3.32}$$

The system Hamiltonian of the Nosé-Hoover equations is:

$$H = \sum_{i=1}^{N} \frac{\mathbf{p}_i}{2m_i} + U\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right) + \frac{p_\xi^2}{2Q} + N_f kT\xi \tag{3.33}$$

The mass parameter Q can be expressed in terms of the kinetic energy oscillation period between the system and the reservoir τ_T as:

$$Q = \frac{\tau_T^2 T_0}{4\pi^2}. \tag{3.34}$$

The difference between weak coupling and Nosé-Hoover is that weak coupling involves exponential relaxation, while Nosé-Hoover uses oscillatory relaxation. The default number of thermostat chains is 10. A chain of temperature controlling particles can be included by modifying the equations to:

$$\begin{aligned} \frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{d}t^2} &= \frac{\mathbf{F}_i}{m_i} - \frac{p_{\xi_1}}{Q_1}\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} \\ \frac{\mathrm{d}p_{\xi_1}}{\mathrm{d}t} &= (T - T_0) - p_{\xi_1}\frac{p_{\xi_2}}{Q_2} \end{aligned} \tag{3.35}$$

$$\begin{aligned}\frac{\mathrm{d}p_{\xi_{i-2.N}}}{\mathrm{d}t} &= \left(\frac{p^2_{\xi_{i-1}}}{Q_{i-1}} - kT\right) - p_{\xi_i}\frac{p_{\xi_{i+1}}}{Q_{i+1}} \\ \frac{\mathrm{d}p_{\xi_N}}{\mathrm{d}t} &= \left(\frac{p^2_{\xi_{N-1}}}{Q_{N-1}} - kT\right)\end{aligned} \tag{3.36}$$

The conserved Nosé-Hoover quantity of the chain is:

$$H = \sum_{i=1}^{N} \frac{\mathbf{p}_i}{2m_i} + U\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right) + \sum_{k=1}^{M} \frac{p^2_{\xi_k}}{2Q'_k} + N_f kT\xi_1 + kT\sum_{k=2}^{M}\xi_k \tag{3.37}$$

The length of the chain in the leapfrog simulation is currently restricted to 1. The Trotter decomposition can be used to show the difference between the constant-temperature integrators. The Liouville operator related to Nosé-Hoover is:

$$iL = iL_1 + iL_2 + iL_{\mathrm{NHC}}, \tag{3.38}$$

where

$$\begin{aligned} iL_1 &= \textstyle\sum_{i=1}^{N} \left[\frac{\mathbf{p}_i}{m_i}\right] \cdot \frac{\partial}{\partial \mathbf{r}_i} \\ iL_2 &= \textstyle\sum_{i=1}^{N} \mathbf{F}_i \cdot \frac{\partial}{\partial \mathbf{p}_i} \\ iL_{\mathrm{NHC}} &= \textstyle\sum_{i=1}^{N} -\frac{p_\xi}{Q}\mathbf{v}_i \cdot \nabla_{\mathbf{v}_i} + \frac{p_\xi}{Q}\frac{\partial}{\partial \xi} + \left(T - T_0\right)\frac{\partial}{\partial p_\xi}. \end{aligned} \tag{3.39}$$

The standard velocity Verlet integrator with Nosé-Hoover temperature coupling becomes:

$$\begin{aligned}\exp(iL\Delta t) = {} & \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_2\Delta t/2\right) \\ & \exp\left(iL_1\Delta t\right)\exp\left(iL_2\Delta t/2\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) + \mathcal{O}\left(\Delta t^3\right)\end{aligned} \tag{3.40}$$

The half-step averaged velocity of the Verlet integrator can be decomposed into:

$$\begin{aligned}\exp(iL\Delta t) = {} & \exp\left(iL_2\Delta t/2\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_1\Delta t\right) \\ & \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right)\exp\left(iL_2\Delta t/2\right) + \mathcal{O}\left(\Delta t^3\right)\end{aligned} \tag{3.41}$$

Choosing the starting point right before $\exp\left(iL_1\Delta t\right)$ yields the leapfrog equivalent:

$$\begin{aligned}\exp(iL\Delta t) = \quad & \exp\left(iL_1\Delta t\right)\exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) \\ \exp\left(iL_2\Delta t\right) & \exp\left(iL_{\mathrm{NHC}}\Delta t/2\right) + \mathcal{O}\left(\Delta t^3\right)\end{aligned} \tag{3.42}$$

3.3.3 Andersen thermostat

The Andersen Thermostat is a widely used temperature coupling algorithm that utilizes the NVE integrator and randomizes particle velocities according to the Boltzmann distribution. This thermostat offers two versions: the Andersen-massive, which randomizes velocities simultaneously every $\tau_T/\Delta t$ step, and the Andersen, which randomizes velocities with a small probability every time-step $\Delta t/\tau$ of each particle. When dealing with systems with constraints, the Andersen-massive version must be used due to parallelization issues. Moreover, this thermostat can only operate with the velocity Verlet integrator because it operates on particle velocities at each time-step [32].

3.4 Pressure coupling

In molecular dynamics simulations, pressure coupling is used to maintain a constant pressure in the simulation box, as fluctuations can affect the accuracy of the simulation. There are several algorithms available for pressure coupling, including Berendsen, Parrinello-Rahman, and Velocity-rescale, each of which uses different approaches to regulate pressure in the simulation.

One key aspect of pressure coupling in the software is the ability to control the barostat, which determines the frequency with which the pressure is updated in the simulation. This is important, as too frequent updates can slow down the simulation, while too infrequent updates can result in large fluctuations in pressure. The GROMACS software provides several options for controlling the barostat, including coupling to a reference pressure or to a reference volume.

In addition to controlling the barostat, the program also allows for the use of multiple pressure-coupling groups in a simulation. By dividing the simulation box into multiple regions and coupling each region to a different reference pressure or volume, complex systems with varying pressures or volumes can be simulated. The use of multiple pressure-coupling groups can help in achieving a more realistic and accurate simulation of these systems.

3.4.1 Berendsen pressure coupling

In Berendsen pressure coupling, the pressure relaxes towards a reference pressure P_0 and can be expressed similarly to the temperature coupling in equation 3.27:

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \frac{\mathbf{P}_0 - \mathbf{P}}{\tau_p}, \tag{3.43}$$

with the scaling matrix:

$$\mu_{ij} = \delta_{ij} - \frac{n_{\mathrm{PC}}\Delta t}{3\tau_p}\beta_{ij}\left\{P_{0ij} - P_{ij}(t)\right\}, \tag{3.44}$$

where β is the isothermal compressibility of the system. For anisotropic systems, the scaling matrix must be rotated to obey the boundary conditions in equation 3.2, which makes the real scaling matrix:

$$\mu' = \begin{pmatrix} \mu_{xx} & \mu_{xy} + \mu_{yx} & \mu_{xz} + \mu_{zx} \\ 0 & \mu_{yy} & \mu_{yz} + \mu_{zy} \\ 0 & 0 & \mu_{zz} \end{pmatrix}. \tag{3.45}$$

To account for the conservation of energy, the total energy must be subtracted from the work applied by the barostat for each step.

$$-\sum_{i,j}\left(\mu_{ij} - \delta_{ij}\right)P_{ij}V = \sum_{i,j} 2\left(\mu_{ij} - \delta_{ij}\right)\Xi_{ij}, \tag{3.46}$$

where δ_{ij} is the Kronecker delta and Ξ_{ij} is the virial.

3.4.2 Parrinello-Rahman pressure coupling

In addition to the weak Berendsen pressure coupling, GROMACS provides Parrinello-Rahman pressure coupling to achieve more well-defined NPT ensembles for constant-pressure simulations. The equation of motion of the box vectors follows:

$$\frac{\mathrm{d}\mathbf{b}^2}{\mathrm{d}t^2} = V\mathbf{W}^{-1}\mathbf{b}'^{-1}\left(\mathbf{P} - \mathbf{P}_{ref}\right), \tag{3.47}$$

where V is the volume of the box and $\mathbf{W}$ is the coupling strength. The modified Hamiltonian for the Parrinello-Rahman coupling is:

$$E_{\mathrm{pot}} + E_{\mathrm{kin}} + \sum_i P_{ii} V + \sum_{i,j} \frac{1}{2} W_{ij} \left(\frac{\mathrm{d}b_{ij}}{\mathrm{d}t} \right)^2 , \tag{3.48}$$

which gives the change in displacement of the atoms:

$$\begin{aligned} \frac{\mathrm{d}^2 \mathbf{r}_i}{\mathrm{d}t^2} &= \frac{\mathbf{F}_i}{m_i} - \mathbf{M} \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}, \\ \mathbf{M} &= \mathbf{b}^{-1} \left[\mathbf{b} \frac{\mathrm{d}\mathbf{b}'}{\mathrm{d}t} + \frac{\mathrm{d}\mathbf{b}}{\mathrm{d}t} \mathbf{b}' \right] \mathbf{b}'^{-1}. \end{aligned} \tag{3.49}$$

The inverse mass parameter matrix $\mathbf{W}^{-1}$ controls the deformation of the box and can be expressed in terms of the pressure time-constant τ_p as:

$$\left(\mathbf{W}^{-1} \right)_{ij} = \frac{4\pi^2 \beta_{ij}}{3 \tau_p^2 L}, \tag{3.50}$$

where L is the largest box element. If the elements in $\mathbf{W}^{-1}$ are zero, the box restrictions in Equation 3.2 will automatically be fulfilled. The surface tension can be calculated for systems with phase separation in the xy-plane. The average surface tension can be calculated by subtracting the normal pressure from the lateral pressure:

$$\begin{aligned} \gamma(t) = & \frac{1}{n} \int_0^{L_z} \left\{ P_{zz}(z,t) - \frac{P_{xx}(z,t) + P_{yy}(z,t)}{2} \right\} \mathrm{d}z \\ = & \frac{L_z}{n} \left\{ P_{zz}(t) - \frac{P_{xx}(t) + P_{yy}(t)}{2} \right\}, \end{aligned} \tag{3.51}$$

where n is the number of surfaces and L_z is the height of the box. The height of the box is scaled with μ_{zz} to correct the z-component of the pressure:

$$\begin{aligned} \Delta P_{zz} &= \frac{\Delta t}{\tau_p} \left\{ P_{0zz} - P_{zz}(t) \right\} \\ \mu_{zz} &= 1 + \beta_{zz} \Delta P_{zz} \end{aligned} \tag{3.52}$$

The equivalent correction term for the box in the x/y direction is:

$$\mu_{x/y} = 1 + \frac{\Delta t}{2\tau_p}\beta_{x/y}\left(\frac{n\gamma_0}{\mu_{zz}L_z} - \left\{P_{zz}(t) + \Delta P_{zz} - \frac{P_{xx}(t)+P_{yy}(t)}{2}\right\}\right) \tag{3.53}$$

3.4.3 MTTK pressure control algorithms

The MTTK equations are used to combine temperature and pressure coupling, where $\epsilon = (1/3)\ln(V/V_0)$, $v_\epsilon = p_\epsilon/W = \dot{\epsilon} = \dot{V}/3V$ and $\alpha = 1 + 3/N_{dof}$. The leapfrog integrator is not optimal for constant pressure systems because it does not provide information about the virial and the kinetic energy until after the time-step. The isobaric equations are given by:

$$\begin{array}{rcl} \dot{\mathbf{r}}_i & = & \frac{\mathbf{p}_i}{m_i} + \frac{p_\epsilon}{W}\mathbf{r}_i \\ \frac{\dot{\mathbf{p}}_i}{m_i} & = & \frac{1}{m_i}\mathbf{F}_i - \alpha\frac{p_\epsilon}{W}\frac{\mathbf{p}_i}{m_i} \\ \dot{\epsilon} & = & \frac{p_\epsilon}{W} \\ \frac{\dot{p}_\epsilon}{W} & = & \frac{3V}{W}\left(P_{\text{int}} - P\right) + (\alpha - 1)\left(\sum_{n=1}^{N}\frac{\mathbf{p}_i^2}{m_i}\right), \end{array} \tag{3.54}$$

where

$$P_{\text{int}} = P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{\mathbf{p}_i^2}{2m_i} - \mathbf{r}_i \cdot \mathbf{F}_i\right)\right]. \tag{3.55}$$

The acceleration term of ϵ is given by:

$$\frac{\dot{p}_\epsilon}{W} = \frac{3V}{W}\left(\alpha P_{\text{kin}} - P_{\text{vir}} - P\right) \tag{3.56}$$

The velocities can be derived and expressed as:

$$\begin{aligned}
\dot{\mathbf{r}}_i =& \quad \mathbf{v}_i + v_\epsilon \mathbf{r}_i \\
\dot{\mathbf{v}}_i =& \quad \frac{1}{m_i}\mathbf{F}_i - \alpha v_\epsilon \mathbf{v}_i \\
\dot{\epsilon} =& \\
\dot{v}_\epsilon =& \quad \frac{3V}{W}\left(P_{\text{int}} - P\right) + (\alpha - 1)\left(\sum_{n=1}^{N}\frac{1}{2}m_i\mathbf{v}_i^2\right) \\
P_{\text{int}} =& \quad P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{1}{2}m_i\mathbf{v}_i^2 - \mathbf{r}_i \cdot \mathbf{F}_i\right)\right]
\end{aligned} \tag{3.57}$$

The energy conservation Hamiltonian is:

$$H = \sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{2m_i} + U\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right) + \frac{p_\epsilon}{2W} + PV \tag{3.58}$$

Adding Nosé-Hoover temperature control variables indexed η and Q' thermostat coupling gives:

$$\begin{aligned}
\dot{\mathbf{r}}_i =& \quad \frac{\mathbf{p}_i}{m_i} + \frac{p_\epsilon}{W}\mathbf{r}_i \\
\frac{\dot{\mathbf{p}}_i}{m_i} =& \quad \frac{1}{m_i}\mathbf{F}_i - \alpha\frac{p_\epsilon}{W}\frac{\mathbf{p}_i}{m_i} - \frac{p_{\xi_1}}{Q_1}\frac{\mathbf{p}_i}{m_i} \\
\dot{\epsilon} =& \quad \frac{p_\epsilon}{W} \\
\frac{\dot{p}_\epsilon}{W} =& \frac{3V}{W}\left(\alpha P_{\text{kin}} - P_{\text{vir}} - P\right) - \frac{p_{\eta_1}}{Q'_1}p_\epsilon \\
\dot{\xi}_k =& \quad \frac{p_{\xi_k}}{Q_k} \\
\dot{\eta}_k =& \quad \frac{p_{\eta_k}}{Q'_k} \\
\dot{p}_{\xi_k} =& \quad G_k - \frac{p_{\xi_{k+1}}}{Q_{k+1}} \quad k = 1, \ldots, M-1 \\
\dot{p}_{\eta_k} =& \quad G'_k - \frac{p_{\eta_{k+1}}}{Q'_{k+1}} \quad k = 1, \ldots, M-1 \\
\dot{p}_{\xi_M} =& \quad G_M \\
\dot{p}_{\eta_M} =& G'_M,
\end{aligned} \tag{3.59}$$

where

$$\begin{aligned}
P_{\text{int}} &= P_{\text{kin}} - P_{\text{vir}} = \frac{1}{3V}\left[\sum_{i=1}^{N}\left(\frac{\mathbf{p}_i^2}{2m_i} - \mathbf{r}_i \cdot \mathbf{F}_i\right)\right] \\
G_1 &= \sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{m_i} - N_f kT \\
G_k &= \frac{p_{\xi_{k-1}}^2}{2Q_{k-1}} - kTk = 2,\ldots,M \\
G_1' &= \frac{{p_\epsilon}^2}{2W} - kT \\
G_k' &= \frac{p_{\eta_{k-1}}^2}{2Q_{k-1}'} - kTk = 2,\ldots,M.
\end{aligned} \tag{3.60}$$

The related Hamiltonian is:

$$\begin{gathered}
H = \textstyle\sum_{i=1}^{N} \frac{\mathbf{p}_i}{2m_i} + U\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right) + \frac{p_\epsilon^2}{2W} + PV + \\
\textstyle\sum_{k=1}^{M} \frac{p_{\xi_k}^2}{2Q_k} + \sum_{k=1}^{M} \frac{p_{\eta_k}^2}{2Q_k'} + N_f kT\xi_1 + kT\sum_{i=2}^{M}\xi_k + kT\sum_{k=1}^{M}\eta_k.
\end{gathered} \tag{3.61}$$

The Trotter decomposition for temperature and pressure control gives the Liouville operator:

$$iL = iL_1 + iL_2 + iL_{\epsilon,1} + iL_{\epsilon,2} + iL_{\text{NHC}-\text{ baro}} + iL_{\text{NHC}}, \tag{3.62}$$

The first term iL_1 describes the evolution of the positions and momenta of the particles in the system. The second term iL_2 describes the evolution of the forces acting on the particles. The third term $iL_{\epsilon,1}$ describes the evolution of the momentum of the temperature thermostat controlling the temperature of the system. Finally, the fourth term $iL_{\epsilon,2}$ describes the evolution of the momentum of the thermostat controlling the pressure/volume of the system. NHC − baro is the Nosè-Hoover chain of the barostat (which controls the pressure and volume of the system) and NHC the Nosè-Hoover chain of the particles (which controls the temperature of the system). The first four terms can be mathematically expressed as:

$$\begin{aligned}
iL_1 = & \sum_{i=1}^{N}\left[\frac{\mathbf{p}_i}{m_i}+\frac{p_\epsilon}{W}\mathbf{r}_i\right]\cdot\frac{\partial}{\partial\mathbf{r}_i} \\
iL_2 = & \sum_{i=1}^{N}\mathbf{F}_i-\alpha\frac{p_\epsilon}{W}\mathbf{p}_i\cdot\frac{\partial}{\partial\mathbf{p}_i} \\
iL_{\epsilon,1} = & \frac{p_\epsilon}{W}\frac{\partial}{\partial\epsilon} \\
iL_{\epsilon,2} = & G_\epsilon\frac{\partial}{\partial p_\epsilon},
\end{aligned} \tag{3.63}$$

where

$$G_\epsilon = 3V\left(\alpha P_{\text{kin}} - P_{\text{vir}} - P\right). \tag{3.64}$$

The Trotter decomposition gives:

$$\exp(iL\Delta t) = \begin{array}{r}\exp\left(iL_{\text{NHC}-\text{ baro}}\,\Delta t/2\right)\exp\left(iL_{\text{NHC}}\Delta t/2\right) \\ \exp\left(iL_{\epsilon,2}\Delta t/2\right)\exp\left(iL_2\Delta t/2\right) \\ \exp\left(iL_{\epsilon,1}\Delta t\right)\exp\left(iL_1\Delta t\right) \\ \exp\left(iL_2\Delta t/2\right)\exp\left(iL_{\epsilon,2}\Delta t/2\right) \\ \exp\left(iL_{\text{NHC}}\Delta t/2\right)\exp\left(iL_{\text{NHC}-\text{ baro}}\,\Delta t/2\right)+\mathcal{O}\left(\Delta t^3\right).\end{array} \tag{3.65}$$

$\exp\left(iL_1\Delta t\right)$ comes from the solution of the differential equation $\dot{\mathbf{r}}_i = \mathbf{v}_i + v_\epsilon\mathbf{r}_i$, where $\mathbf{v}_i = \mathbf{p}_i/m_i$, with constant v_ϵ and initial condition $\mathbf{r}_i(0)$, yielding:

$$\mathbf{r}_i(\Delta t) = \mathbf{r}_i(0)e^{v_\epsilon\Delta t} + \Delta t\mathbf{v}_i(0)e^{v_\epsilon\Delta t/2}\frac{\sinh\left(v_\epsilon\Delta t/2\right)}{v_\epsilon\Delta t/2}. \tag{3.66}$$

$\exp\left(iL_2\Delta t/2\right)$ comes from solving the differential equation $\dot{\mathbf{v}}_i = \frac{\mathbf{F}_i}{m_i} - \alpha v_\epsilon\mathbf{v}_i$, yielding:

$$\mathbf{v}_i(\Delta t/2) = \mathbf{v}_i(0)e^{-\alpha v_\epsilon\Delta t/2} + \frac{\Delta t}{2m_i}\mathbf{F}_i(0)e^{-\alpha v_\epsilon\Delta t/4}\frac{\sinh\left(\alpha v_\epsilon\Delta t/4\right)}{\alpha v_\epsilon\Delta t/4}. \tag{3.67}$$

3.5 Energy error

In molecular dynamics simulations, a pair-list cutoff is the maximum distance between any two particles beyond which their interactions are not considered in the simulation. To improve computational efficiency, pair lists are created that include only particle pairs within this cutoff distance. A Verlet buffer size is a small additional distance added to the cutoff distance of the pair list. This additional buffer distance is used to ensure that all particle pairs that interact with each other are included in the simulation calculations, even if they are just outside the actual cutoff distance. The average energy error after time t can be written as follows:

$$\langle \Delta V \rangle = \int_0^{r_c} \int_{r_\ell}^{\infty} 4\pi r_0^2 \rho_2 V\left(r_t\right) G\left(\frac{r_t - r_0}{\sigma}\right) dr_0 dr_t, \tag{3.68}$$

where ρ_2 is the density of atoms j surrounded by atom i. $V(r_t)$ can be approximated around r_c with Taylor series expansion, giving:

$$\begin{aligned} \langle \Delta V \rangle \approx \textstyle\int_{-\infty}^{r_c} \int_{r_\ell}^{\infty} 4\pi r_0^2 \rho_2 \left[V'\left(r_c\right)\left(r_t - r_c\right) + \right. \\ V''\left(r_c\right) \tfrac{1}{2}\left(r_t - r_c\right)^2 + \\ V'''\left(r_c\right) \tfrac{1}{6}\left(r_t - r_c\right)^3 + \\ \left. O\left(\left(r_t - r_c\right)^4\right)\right] G\left(\tfrac{r_t - r_0}{\sigma}\right) dr_0 dr_t \end{aligned} \tag{3.69}$$

Further replacing r_0^2 with $\left(r_\ell + \sigma\right)^2$ gives a solution to the average energy error integral:

$$\begin{aligned} \langle \Delta V \rangle \approx \left(r_\ell + \sigma\right)^2 \rho_2 \textstyle\int_{-\infty}^{r_c} \int_{r_\ell}^{\infty} \left[V'\left(r_c\right)\left(r_t - r_c\right) + \right. \\ V''\left(r_c\right) \tfrac{1}{2}\left(r_t - r_c\right)^2 + \\ \left. V'''\left(r_c\right) \tfrac{1}{6}\left(r_t - r_c\right)^3\right] G\left(\tfrac{r_t - r_0}{\sigma}\right) dr_0 dr_t \\ = \quad 4\pi\left(r_\ell + \sigma\right)^2 \rho_2 \left\{\tfrac{1}{2} V'\left(r_c\right)\left[r_b \sigma G\left(\tfrac{r_b}{\sigma}\right) - \left(r_b^2 + \sigma^2\right) E\left(\tfrac{r_b}{\sigma}\right)\right] + \right. \\ \tfrac{1}{6} V''\left(r_c\right)\left[\sigma\left(r_b^2 + 2\sigma^2\right) G\left(\tfrac{r_b}{\sigma}\right) - r_b\left(r_b^2 + 3\sigma^2\right) E\left(\tfrac{r_b}{\sigma}\right)\right] + \\ \tfrac{1}{24} V'''\left(r_c\right)\left[r_b \sigma\left(r_b^2 + 5\sigma^2\right) G\left(\tfrac{r_b}{\sigma}\right)\right. \\ \left.\left. - \left(r_b^4 + 6 r_b^2 \sigma^2 + 3\sigma^4\right) E\left(\tfrac{r_b}{\sigma}\right)\right]\right\}, \end{aligned} \tag{3.70}$$

where $E(x) = \frac{1}{2}\ \mathrm{erfc}(x/\sqrt{2})$. The energy error needs to be further averaged and weighted over all particle pair types. To obtain the average error per unit time the expression is further devided by $t = (\text{ nstlist } - 1) \times \text{dt}$, e.g. the neighbor-list life time. The energy error can be decreased by adding bond constraints that limit the degree of freedom of the particles. A particle with two degrees of freedom generates the average energy error:

$$\frac{\sqrt{\pi}}{2\sqrt{2}\sigma}\text{erfc}\left(\frac{|r|}{\sqrt{2}\sigma}\right), \tag{3.71}$$

which cannot be solved analytically but can be converted to a scaled and shifted Gaussian distribution to generate a tight upper bound.

In NVT equilibration, the energy error caused by the periodic boundary conditions is calculated from the displacement of the atoms and the potential at the cut-off. The average energy error is caused by particles that over time move from outside the pair-list cut-off r_l inside the interaction cut-off r_c. A free particle in one dimension follows the Gaussian G(x) displacement distribution, with mean $\mu = 0$ and variance $\sigma^2 = \sigma_{12}^2 = t^2 k_B T\,(1/m_1 + 1/m_2)$. The variance in distance between two atoms is, $\sigma^2 = \sigma_{12}^2 = t^2 k_B T\,(1/m_1 + 1/m_2)$. In practice, particles encounter other particles easily, which makes the displacement distribution narrower in reality. In each direction, the box has three images (-1, 0, 1) and at most one image will see the j particle.

When multiple temperature-coupling groups are used, the degrees of freedom for group i is:

$$N_{\text{df}}^i = \left(3N^i - N_c^i\right)\frac{3N - N_c - N_{\text{com}}}{3N - N_c}. \tag{3.72}$$

The kinetic energy written as a tensor is:

$$\mathbf{E}_{\text{kin}} = \frac{1}{2}\sum_i^N m_i \mathbf{v}_i \otimes \mathbf{v}_i. \tag{3.73}$$

The pressure tensor $\mathbf{P}$ can be calculated with the formula:

$$\mathbf{P} = \frac{2}{V}\left(\mathbf{E}_{\text{kin}} - \mathbf{\Xi}\right), \tag{3.74}$$

where V is the volume of the box and $\mathbf{\Xi}$ is the virial, which equals:

$$\boldsymbol{\Xi} = -\frac{1}{2}\sum_{i<j} \mathbf{r}_{ij} \otimes \mathbf{F}_{ij} \tag{3.75}$$

The virial measures the total amount of work done by the inter-particle forces in the system, and is directly related to the pressure. The scalar pressure for an isotropic system can be written from equation 3.74 as:

$$P = \text{trace}(\mathbf{P})/3 \tag{3.76}$$

The velocity-rescaling temperature coupling is similar to the Berendsen thermostat, with an additional kinetic energy-correcting term:

$$\mathrm{d}K = (K_0 - K)\frac{\mathrm{d}t}{\tau_T} + 2\sqrt{\frac{KK_0}{N_f}}\frac{\mathrm{d}W}{\sqrt{\tau_T}}, \tag{3.77}$$

where dW is a Wiener process, i.e., a stochastic process used to model random movements or fluctuations in a system. The velocity-rescaling thermostat has advantages over the Berendsen thermostat, because it has no oscillations and first-order temperature deviation decay.

3.6 GROMACS Simulation Parameters: Choosing Force Field, Solvent Model, and Integration Settings

GROMACS requires the specification of numerical parameters to carry out the simulation. These parameters include force field, solvent model, integration time step, temperature, pressure, and others, and are chosen based on the specific system being studied and the desired level of accuracy in the simulation results. It is important to carefully choose these parameters to ensure that the simulation accurately represents the physical behavior of the system being studied.

The GROMACS parameters chosen can be divided into three sections based on the simulation type: energy minimization, NVT equilibration, and NPT equilibration.

For energy minimization, the steepest descent algorithm was used with a maximum force tolerance of 1000 kJ/mol/nm, an energy step size of 0.01, and a maximum of 50,000 minimization steps. The neighbor list was updated every step using the Verlet cutoff scheme and grid method with periodic boundary conditions in all three dimensions. The treatment of long-range electrostatic interactions was performed using the PME method, and the cutoff distances for short-range electrostatic and van der Waals interactions were set to 1.0 nm.

For NVT equilibration, a leap-frog integrator was used with a time step of 2 fs, and velocities were assigned from a Maxwell distribution at 300 K. Temperature coupling was applied using the V-rescale thermostat, a modified Berendsen thermostat, with two coupling groups (protein and non-protein) and a time constant of 0.1 ps. Periodic boundary conditions were applied in all three dimensions, and a dispersion correction was used to account for the cut-off van der Waals scheme.

For NPT equilibration, the same parameters were used as for NVT equilibration with the addition of a Parrinello-Rahman barostat to maintain a constant pressure of 1 atm.

For this project the forcefield CHARMM27 and the water model TIP3P were used. CHARMM27 is a classical force field for molecular dynamics simulations that is commonly used in biomolecular simulations. It is a united atom force field, meaning that some of the hydrogen atoms are merged with their parent carbon or nitrogen atoms to simplify the simulation. TIP3P (Transferable Intermolecular Potential with 3 Points) is a popular water model used in molecular simulations that represents water molecules as three point charges (one oxygen and two hydrogens) interacting through pairwise Lennard-Jones and Coulombic potentials.

CHARMM27 force field parameters include bond lengths, angles, torsions, and non-bonded interactions such as van der Waals and electrostatic forces. The force field includes parameters for amino acids, nucleic acids, lipids, and other small molecules commonly found in biological systems. The force field can be implemented by using the appropriate force field files and topology files, which contain the necessary parameters and instructions for running the simulation. [33].

The TIP3P water model was developed by Jorgensen and co-workers in the early 1980s and has been extensively used in various biomolecular simulations. One of the advantages of this water model is that it accurately reproduces the experimental value of the heat of vaporization of water. However, TIP3P has some limitations, such as an overestimation of the hydrogen bond angle and a slightly underestimated density compared to experimental values [34].

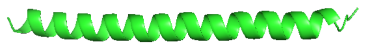
Chapter 4

Results

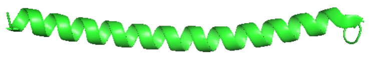
The peptides analyzed in this thesis were synthesized from repeating short amino acid motifs. Due to the large computational workload, the peptides were initially reduced to shorter sequences to gain an understanding of their behavior. The original sequences were 250–300 AA long (amino acids), but for analysis they were shortened to 50, 100, and 200 AA, to test differences in simulation success for shorter and longer peptides. Some proteins could not achieve a proper energy-minimizing structure, which caused problems during the NVT equilibration. Problems with overlapping atoms were caused by inadequate initial structure parameters. This problem could be solved with a soft-core potential that separated the overlapping atoms from each other. Some proteins were still too entangled after the soft-core step and were consequently excluded from the analysis.

MD simulation results

This section presents the results obtained from the molecular dynamics simulations of the proteins under investigation. The analysis of the simulations revealed a range of behaviors among the simulated proteins, with some exhibiting high stability while others underwent significant conformational changes. Representative cases of both types will be presented in this section. Additionally, this section will include plots of the distribution of average RMSD values for the last 10 simulation points for each protein and the distribution of radius of gyration, providing further insights into the dynamics and stability of the proteins. Furthermore, visualizations of selected proteins that could not be fixed during the simulation will be presented. These examples provide insight into the limitations and challenges of molecular dynamics simulations. Some of the predicted proteins were folded into almost perfect alpha helices, which can be seen in Figures 4.1, 4.2 and 4.3. These structures also accounted for the structures with the smallest average RMSD during the stabilization. In comparison, the proteins with the largest RMSD can be seen in figures 4.4, 5.6 and 4.6.

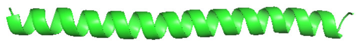


(a) Machine learning-predicted initial structure.

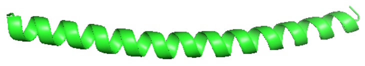


(b) Structure after 20 ns MD simulation.

Figure 4.1: Sequence nr 157, 50 AA.

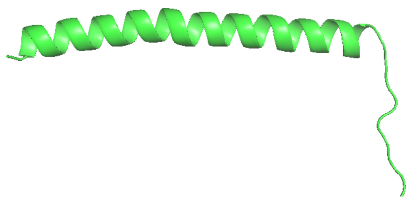


(a) Machine learning-predicted initial structure.

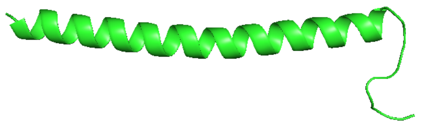


(b) Structure after 20 ns MD simulation.

Figure 4.2: Sequence nr 155, 50 AA.

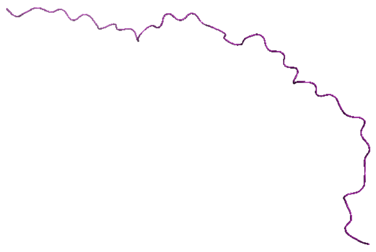


(a) Machine learning-predicted initial structure.

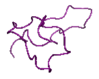


(b) Structure after 20 ns MD simulation.

Figure 4.3: Sequence nr 40, 50 AA.

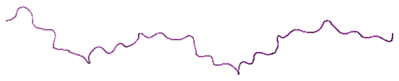


(a) Machine learning-predicted initial structure.

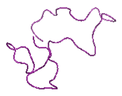


(b) Structure after 20 ns MD simulation.

Figure 4.4: Sequence nr 145, 50 AA.

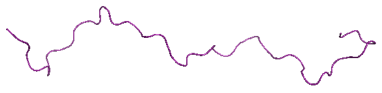


(a) Machine learning-predicted initial structure.

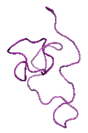


(b) Structure after 20 ns MD simulation.

Figure 4.5: Sequence nr 33, 50 AA.



(a) Machine learning-predicted initial structure.



(b) Structure after 20 ns MD simulation.

Figure 4.6: Sequence nr 165, 50 AA.

All simulations were not run successfully due to overlapping atoms and a cluster-like initial machine learning predicted structure. The majority of the cases with parts of overlapping structures could be fixed with a soft-core potential energy minimization step. This additional step separated overlapping atoms, which allowed the simulations to run. The action of the soft-core potential step can be seen in figure 4.7.

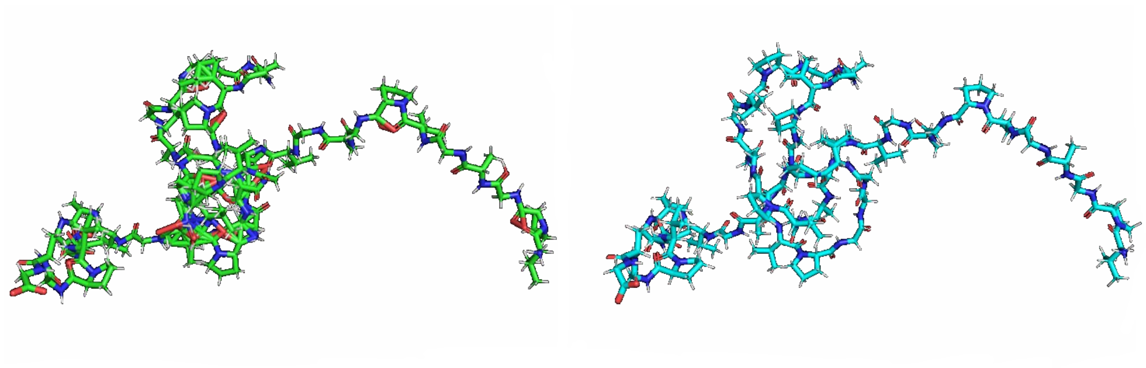
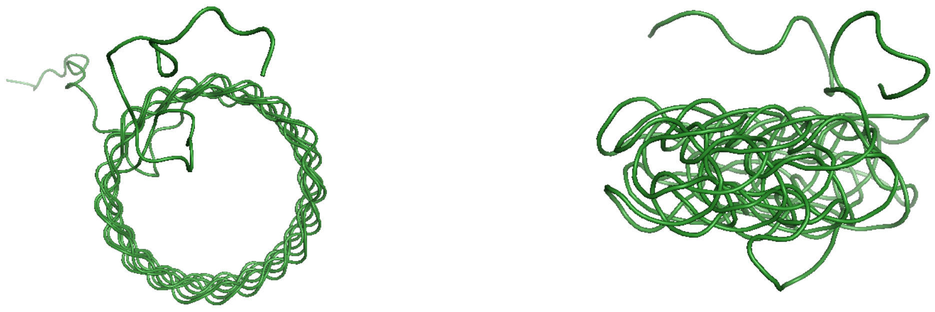


Figure 4.7: Overlapping protein before and after soft-core potential.

Some cases were so entangled that they could not be fixed with a soft-core potential. These nonphysical structures were left outside of the analysis and examples can be seen in figure 4.8:



(a) Machine learning-predicted structure, nr. 59.

(b) Machine learning-predicted structure, nr. 162.

Figure 4.8: Nonphysical original structures (not fixable).

Figure 4.9 shows the distribution of RMSD for proteins of amino acid lengths 50, 100, 200 and larger proteins. Due to extremely overlapping atoms, 1/178 of the 200 AA proteins, and 3/178 of the original sequences could not be simulated. As a result, the x-axis in Figure 4.9 was normalized to allow for a fair comparison between the simulated proteins. The normalization was done by scaling the x-axis to a range of 0 to 1, where 1 represents the maximum value of the remaining data points.

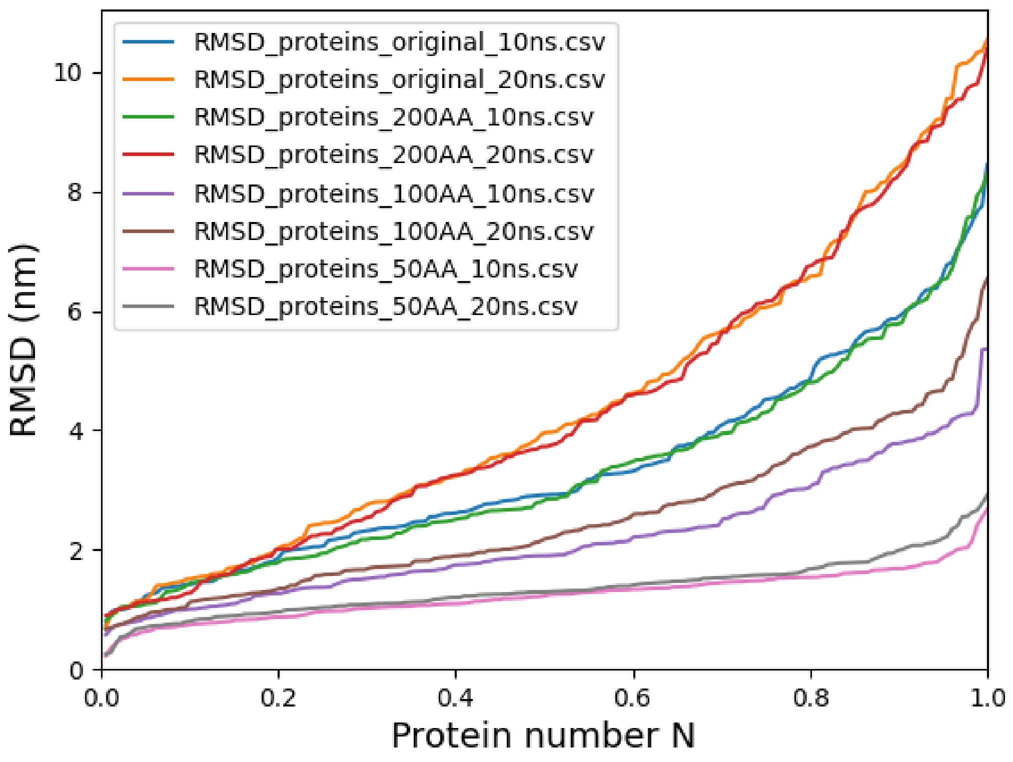


Figure 4.9: RMSD, normalized distribution.

As can be seen in figure 4.9, it seems that the distribution for 200 AA and the original structure follows an almost identical distribution for both shorter (10 ns) and longer simulation time (20 ns). The longest sequences seems to have the largest RMSD, which agrees with previous studies [35]. To compare the sequences independent of length aspects, the ρ_{sc} values will be compared in the next chapter.

Figure 4.10 shows the distribution of radius of gyration for the proteins.

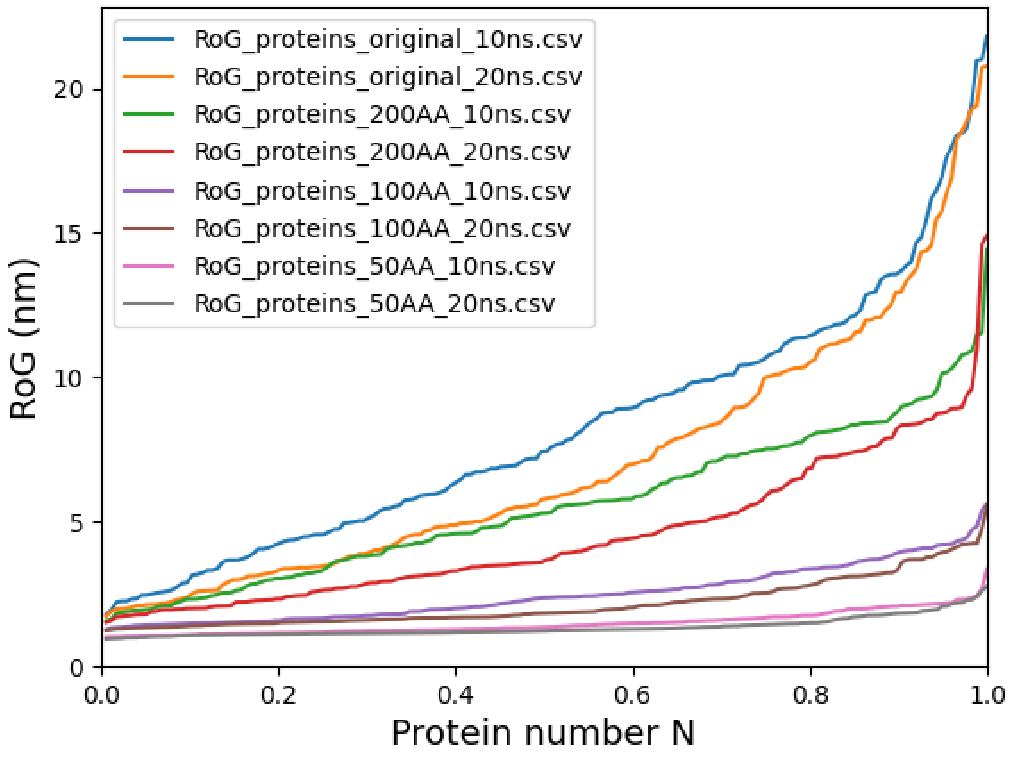


Figure 4.10: Radius of Gyration, normalized distribution.

The distribution of radius of gyration follows a clear trend, whereby the longest sequences with the shortest simulation time exhibit the largest conformational changes and have the broadest distribution of radii. Conversely, the shortest sequences with the longest simulation time exhibit minimal changes in compactness and have a narrower distribution of radii. This trend can be attributed to the fact that the longer sequences have a greater degree of conformational freedom and can adopt a wider range of conformations in response to the simulated environment. Conversely, the shortest sequences with the longest simulation time exhibit minimal changes in compactness and have a narrower distribution of radii. This observation can be explained by the fact that the shorter sequences have fewer degrees of freedom and are more constrained in their conformational space, resulting in a more limited range of possible conformations [36].

4.1 Estimating the accuracy of machine learning model

The goal of this project is to assess the accuracy of protein structures predicted by a machine learning model, equivalent to AlphaFold, using molecular dynamics simulations. This thesis utilizes molecular dynamics simulations to evaluate the stability of structures predicted by a machine learning model, to determine if they can be considered to be in their native state. If the structure of the proteins deviate significantly from their initial state, the machine learning model has not found a stable conformation, if one exists. Additionally, anomalous behavior of the proteins during the simulation may indicate that the prediction does not align with the objectives of the study.

The RMSD metric is often used to compare the structural similarity of proteins during molecular dynamics simulations. However, the interpretation of RMSD values can be challenging, particularly above a certain arbitrary cutoff value for significant similarity. A lower RMSD value generally indicates a better fit between two structures, but there is no universally agreed upon cutoff value for significant dissimilarity. The cutoff value considered significant can vary depending on several factors, such as the size and complexity of the molecules being compared and the research question. To address this limitation, a new measure of structural similarity called ρ_{sc} has been proposed. Unlike RMSD, ρ_{sc} is independent of the sizes of the molecules being compared.

The ρ value is a measure between two arbitrary configurations A and B, based on the ratio of the radii of gyration (equation 2.15) for the difference and sum structures. R(s) is the radius of gyration of the sum structure, which is a measure of the overall size of the two molecules when combined. R(d) is a measure of the structural differences between the two molecules. Mathematically, it is expressed as:

$$\rho(A,B) = \frac{2R(d)}{R(s)} = \frac{2D(A,B)}{\left[2R^2(A) + R^2(B) - D^2(A,B)\right]^{1/2}} \tag{4.1}$$

Here, $R^2(A)$ and $R^2(B)$ are the radii of gyration for structures A and B. The denominator is calculated using the Euclidean distance formula for three-dimensional space, $D(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$, where (x_A, y_A, z_A) and (x_B, y_B, z_B) are the coordinates of the corresponding atoms in molecules A and B, respectively. The common expression between the structures A and B is:

$$D^2(A,B) = n^{-1} \sum_i (\mathbf{a}_i - \mathbf{b}_i)^2 \,, \tag{4.2}$$

where n is the total number of atoms in the structures, and $\mathbf{a}_i$ and $\mathbf{b}_i$ are the positions of the ith atom in molecules A and B, respectively.

The ρ_{sc} values are obtained by first spherically scaling the structures. The scaling factor is chosen so that the sizes, overall shapes, and numbers of points in the structures are independent. These can be compared to certain threshold values to determine the degree of structural similarity between the structures being compared.

When ρ_{sc} is less than 0.3-0.5, proteins are visually recognized as having obvious similarity, while a cutoff of $\rho_{sc} < 1.0$ corresponds to an overall similarity in folding motif. The ρ_{sc} value intervals and their interpretation can be seen in table 4.1. Importantly, the intrinsic cutoff is independent of the number of amino acids or points being compared. It is worth noting that for proteins with fewer than 100 amino acids, geometrically significant similarity can often occur by chance [37].

In cases where the sizes of proteins being compared differ significantly, ρ_{sc} can provide a more reliable metric than RMSD for assessing structural similarity. Moreover, the intrinsic cutoff value of ρ_{sc} can offer a standardized and objective measure for determining the overall similarity of protein folding motifs. Given these advantages, the analysis of the systems in this thesis will primarily rely on the ρ_{sc} value as the main comparison metric.

0	Identical structures
$<$ 0.3-0.5	Visually recognizable similarity
$<$ 0.894	Antisimilarity impossible
$<$ 1	Structural commonality exceeds difference

Table 4.1: Interpretation of ρ_{sc} values.

Nonphysical behavior in proteins during MD simulations can also indicate inaccurate predictions by the machine learning model. Overlapping atoms were not an uncommon issue during energy minimization and temperature equilibration. Typically Gromacs handles such issues through a special procedure called soft-core potential that means the overlapping atoms are separated from each other, making the structures manageable for the MD simulation program. These fixes could improve the accuracy of the machine learning model and provide information on unwanted behavior, although they do not guarantee accurate structures.

Figure 4.11 displays the distribution of ρ_{sc} among the various protein systems of different lengths. The proteins are divided into two groups based on their ρ_{sc} values. Those with values lower than or equal to 1 are located on the left side of the corresponding vertical dashed red line, while those with values greater than 1 are located on the right side of the line.

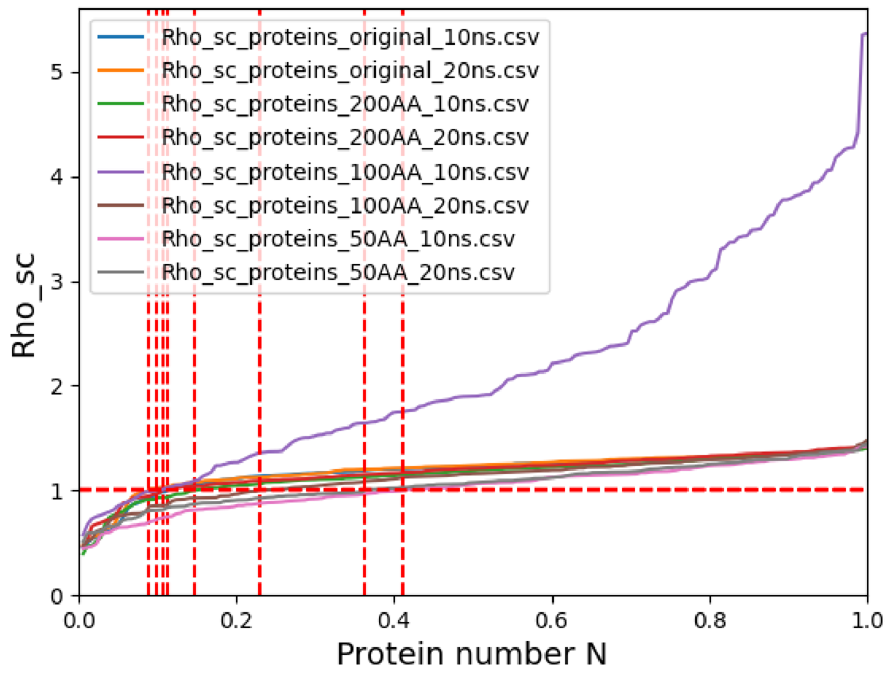
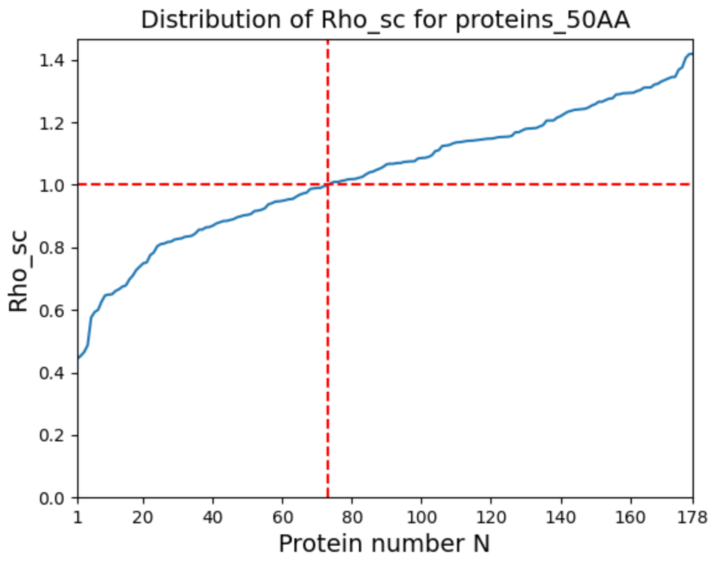
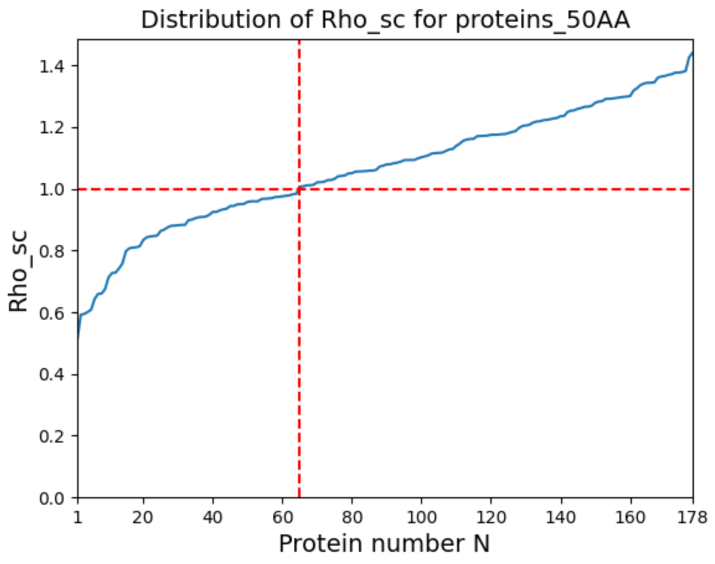


Figure 4.11: ρ_{sc} summation normalized.

The majority of the proteins in this study experienced a significant conformational change, as reflected by their ρ_{sc} values greater than 1, with many final structures differing significantly from the initial structures. Evidently, the machine learning model appears to have generated proteins that are unstable during MD simulations. Investigating the underlying reasons for this trend would be beneficial for improving protein structure prediction. The ρ_{sc} distribution of each protein system for 10 ns and 20 ns simulations is plotted in Figure 4.12, 4.14, 4.14 and 4.15.

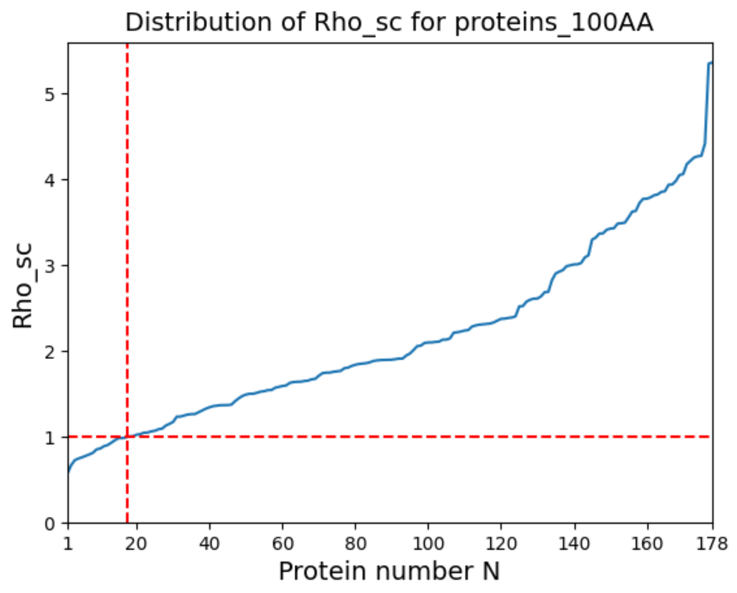


(a) ρ_{sc} after 10 ns simulation.

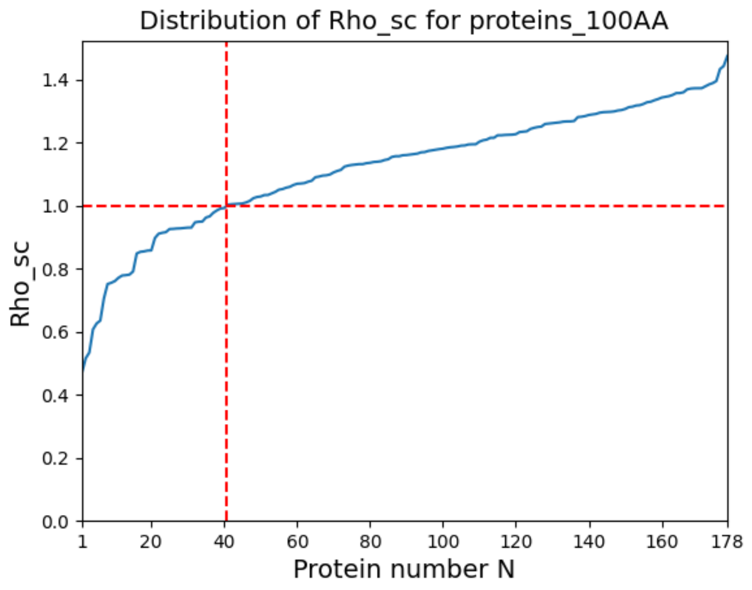


(b) ρ_{sc} after 20 ns simulation.

Figure 4.12: Proteins built of 50 amino acids.

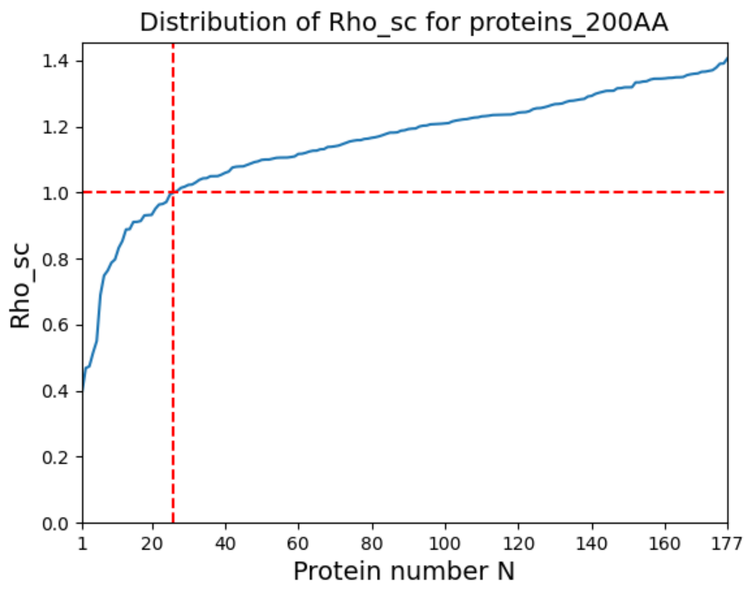


(a) ρ_{sc} after 10 ns simulation.

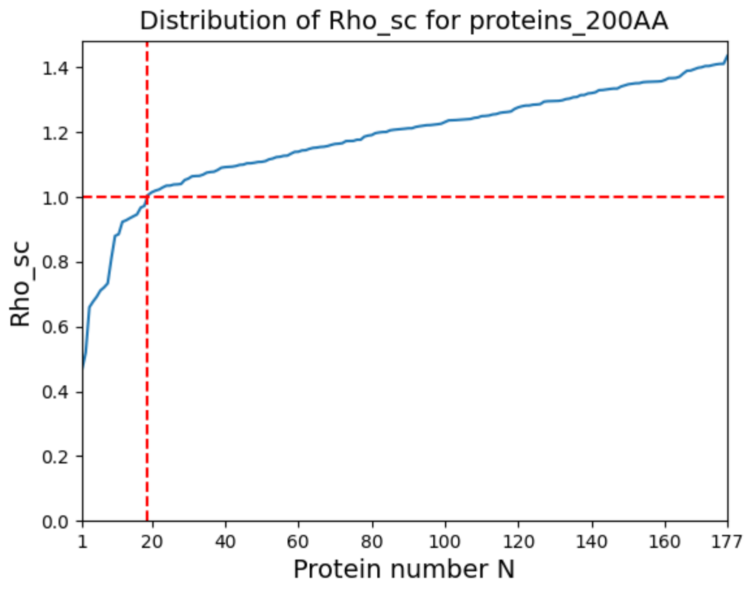


(b) ρ_{sc} after 20 ns simulation.

Figure 4.13: Proteins built of 100 amino acids.

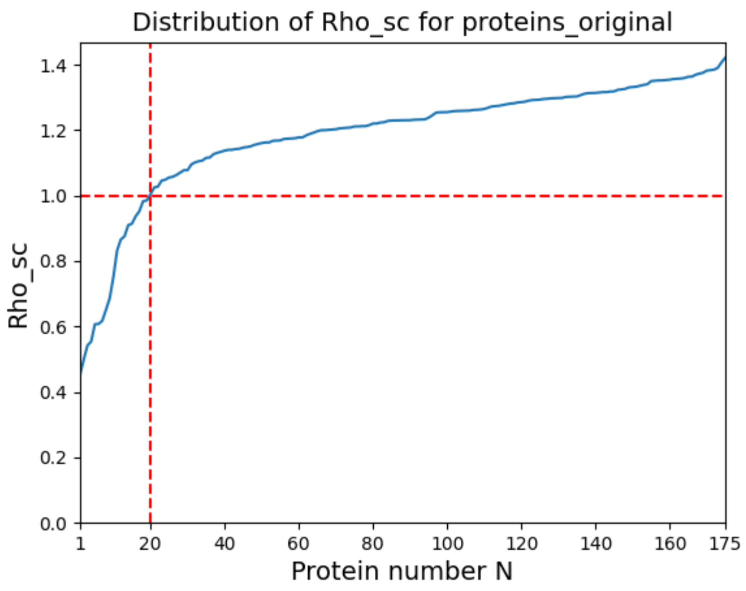


(a) ρ_{sc} after 10 ns simulation.

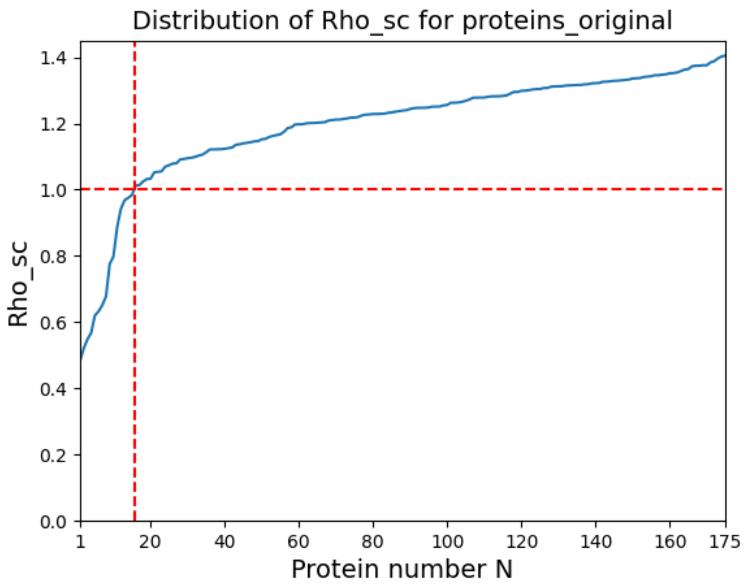


(b) ρ_{sc} after 20 ns simulation.

Figure 4.14: Proteins built of 200 amino acids.



(a) ρ_{sc} after 10 ns simulation.



(b) ρ_{sc} after 20 ns simulation.

Figure 4.15: Proteins built of 250-300 amino acids.

The analysis showed that shorter protein sequences demonstrated greater stability, with approximately 36% of proteins having a ρ_{sc} value below 1 after 20ns. In contrast, longer sequences were associated with larger conformational changes, indicating lower prediction accuracy. Table 4.2 shows the performance of proteins of different lengths and on different time frames.

	10 ns	20 ns
50 AA	41.0 %	36.0 %
100 AA	9.6 %	22.5 %
200 AA	14.1 %	10.2 %
Original	10.9 %	8.6 %

Table 4.2: Percentage of proteins with $\rho_{sc} \leq 1$

Notably, only about 10% of the proteins in their original form displayed near stable structures during the MD simulation, indicating that the initial structures were not at their native state. Due to their synthetic nature, the machine learning model may have had difficulty generating stable conformations for these proteins. Synthetic peptides may exhibit different behavior than those found *in vivo*, which could limit the quality of the input data used in the model. For all sets of sequences except 100 AA, the number of peptides of adequate ρ_{sc} value decreased with increased simulation time.

Another way to determine the success of the machine learning prediction was by looking at the nonphysical behavior of some proteins, such as overlapping proteins. Unwanted behavior in the MD simulations implies that the machine learning model fails to predict the protein structure accurately. Problems during the energy-minimization step and temperature equilibration step mainly occurred due to overlapping atoms.

Instead of completely out-ruling these proteins, a soft-core potential step was added to keep the initial protein as close to the prediction as possible, but with separated overlaps to run them through the simulation. These steps could be important when feeding data back to the machine learning model. These fixes do not necessarily give accurate structures either, but they could help tweak the machine learning model towards more accurate predictions and give information on unwanted behavior.

For proteins of length 200 AA, 1 protein was unfixable, i.e., the protein structure could not be resolved with soft-core potential, and for the original sequences 3 were not fixable. These cases are missing from the analyses.

4.1.1 Conclusion

The field of protein structure prediction using machine learning models is rapidly advancing, but it still faces several limitations. One of the main challenges is the complexity of proteins, which can make it difficult to accurately predict their structure. In addition, the limited diversity of the training dataset can also hinder accurate predictions. These limitations highlight the need for continued research and innovation in the field. Furthermore, synthetic proteins may present even greater challenges for machine learning predictions.

Despite these challenges, there have been some promising advancements in the field. There are models that have been proven to produce highly accurate predictions for some types of proteins. However, these predictions may not be applicable to all types of proteins.

Overall, the study of protein structure prediction using machine learning models is a promising field that holds great potential for advancements in medicine, biotechnology, and food science. With continued innovation and collaboration, there is no doubt that this field will continue to grow and make significant contributions to our understanding of protein behavior and its applications.

Chapter 5

Future outlooks

Liquid-liquid phase separation (LLPS) is a phenomenon that controls the condensation of cell structures, allowing membraneless organelles, such as the nucleoli, to organize into compartments inside the cell. The function of cells depends on their ability to restrict biochemical processes to specific areas inside the cell. LLPS is present in various biological reactions, such as RNA metabolism, gene expression, and cell signaling. Furthermore, unregulated LLPS is believed to be the root cause of many diseases [38]. LLPSDB is a database that gives information about phase separation conditions confirmed by in vitro studies, such as protein sequence, amino acid modifications, and biological functions, along with external factors such as temperature, salt concentration, and pH.

Liquids are states of matter in which components rearrange easily, striving towards a state of higher entropy. This is crucial in biochemical processes, but it needs to be kept in a contained environment. The cell consists of cytoplasm, which is a liquid that is kept inside the cell because of the cell membrane. However, several liquid-like organelles exist inside the cell, which leads to the conclusion that these liquids must have characteristics that separate them from the surrounding liquid. An example of liquids that separate into two phases is oil and vinegar. Maximal entropy is still conserved because vinegar and oil are made up of many different chemical compounds that interact strongly with parts of the same liquid [39].

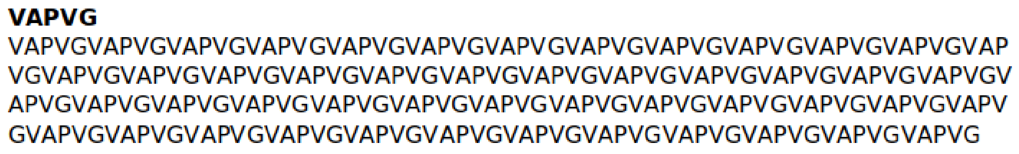
Intrinsically disordered proteins are known to undergo phase separation in cells. Phase separation behavior occurs when macromolecules are separated into a different phase than their surrounding liquid. The two phases differ in concentration but consist of similar components [40].

Proteins are often positively charged and interact with negatively charged RNA sequences. When RNA is low, it will bind to the protein and phase separate, but large quantities of RNA will dissociate. The RNA-binding protein FUS phase separates through its amino-terminal prion-like domain (PrLD). FUS should be malleable in healthy cells, and unregulated accumulation of FUS has been linked to variants of neurodegenerative diseases, e.g., Amyotrophic Lateral Sclerosis (ALS) [41]).

Cell compartments that undergo phase separation are stress granules (SGs), nucleoli, Cajal bodies, and P bodies. When SGs are subjected to pressure, they converge into phase-separated droplets, and when the pressure is removed, the proteins can move more freely again. A study found that the concentration of RNA inside the SGs could be partly responsible for the condensation. G3BP1 is a protein in the SG that binds to free-floating RNA, which leads to conformational changes. Additionally, it is known that the concentration of RNA outside the nucleus is much lower than inside, and a study found that the shape of transcription complexes can be regulated by a feedback loop that controls the existence of free RNA. The primary cause of phase separation is forces between the atoms in the protein, i.e., electrostatic cation-pi and pi-pi interactions [42].

Studies have shown that transcription factors create phase-separated structures through the interaction between a low complexity domain (LCD) and a carboxy-terminal domain (CTD), which is a tandem repeat sequence of RNA polymerase II [43]. Transcription factors also create high-density compartments for RNA transcription.

The sequences analyzed in this thesis were synthetically produced perfect tandem repeats, with the majority of them consisting of five amino acid units repeated about 50–60 times (original length). Minisatellites are DNA structures made up of repetitions of 10–60 short nucleotide sequence motifs, and microsatellites are even fewer; however, perfect tandem repeat proteins are unlikely to be found *in vivo*. The majority of the simulated proteins could be compared to microsatellites in DNA. The protein with the longest motif was built of 27 amino acid units. Figure 5.1 shows an example of the synthetically constructed sequences:



VAPVG
VAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAP
VGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGV
APVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPV
GVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVGVAPVG

Figure 5.1: Example of original protein chain. (This protein consists of repetitions of the sequence motif VAPVG)

Tandem repeats in DNA are known to be unstable and susceptible to mutation as a result of polymorphism. Polymorphism refers to the occurrence of variations in DNA sequences among individuals or populations, which can result in instability and mutations in tandem repeats. They experience a 10–100,000-fold higher mutation rate than the rest, which can have pathological consequences. The benefits are that they contribute to fast adaptation and accelerated changes in gene expression [44]. Microsatellites are made of both coding and non-coding DNA, with the majority being non-coding. Non-coding DNA is thought to exist primarily for gene activity control [45].

The phenomenon of LLPS is a topic of increasing interest in biophysics and bio-engineering. While it is well-established that LLPS occurs in solutions containing polymers and that all proteins are capable of undergoing LLPS under certain conditions, the specific factors that drive phase separation in cellular environments are not yet fully understood. Nevertheless, LLPS has potential in the creation of synthetic materials, including proteins with perfect tandem repeats, by encouraging the formation of desired structures. This could open up possibilities for developing synthetic proteins with unique functions for biomedical applications, such as drug delivery or tissue engineering. Additionally, studying LLPS can provide insights into creating novel biomaterials, as has been demonstrated by recent advancements in the development of LLPS-based technologies for cell culture and drug discovery [46].

Bibliography

[1] S. Damodaran, "Amino acids, peptides and proteins," *Fennema's food chemistry*, vol. 4, pp. 217–329, 2008.

[2] C. M. Dobson, "Protein folding and misfolding," *Nature*, vol. 426, no. 6968, pp. 884–890, 2003.

[3] M. B. Radke, M. H. Taft, B. Stapel, D. Hilfiker-Kleiner, M. Preller, and D. J. Manstein, "Small molecule-mediated refolding and activation of myosin motor function," *Elife*, vol. 3, p. e01603, 2014.

[4] P. A. Temussi, L. Masino, and A. Pastore, "From alzheimer to huntington: why is a structural understanding so difficult?," *The EMBO journal*, vol. 22, no. 3, pp. 355–361, 2003.

[5] H. Hegyi and M. Gerstein, "The relationship between protein structure and function: a comprehensive survey with application to the yeast genome," *Journal of molecular biology*, vol. 288, no. 1, pp. 147–164, 1999.

[6] A. L. Tarca, V. J. Carey, X.-w. Chen, R. Romero, and S. Drăghici, "Machine learning and its applications to biology," *PLoS computational biology*, vol. 3, no. 6, p. e116, 2007.

[7] V. Marx, "Method of the year: Protein structure prediction," *Nature methods*, vol. 19, no. 1, pp. 5–10, 2022.

[8] E. Callaway, "'it will change everything': Deepmind's ai makes gigantic leap in solving protein structures," *Nature*, vol. 588, no. 7837, pp. 203–205, 2020.

[9] J. Moult, "A decade of casp: progress, bottlenecks and prognosis in protein structure prediction," *Current opinion in structural biology*, vol. 15, no. 3, pp. 285–289, 2005.

[10] A. David, S. Islam, E. Tankhilevich, and M. J. Sternberg, "The alphafold database of protein structures: A biologist's guide," *Journal of Molecular Biology*, vol. 434, no. 2, p. 167336, 2022.

[11] C. A. Orengo, A. E. Todd, and J. M. Thornton, "From protein structure to function," *Current Opinion in Structural Biology*, vol. 9, no. 3, pp. 374–382, 1999.

[12] E. Callaway, "What's next for the ai protein-folding revolution," *Nature*, vol. 604, pp. 234–238, 2022.

[13] A. Perrakis and T. K. Sixma, "Ai revolutions in biology: The joys and perils of alphafold," *EMBO reports*, vol. 22, no. 11, p. e54046, 2021.

[14] T. C. Terwilliger, B. K. Poon, P. V. Afonine, C. J. Schlicksup, T. I. Croll, C. Millán, J. Richardson, R. J. Read, P. D. Adams, *et al.*, "Improved alphafold modeling with implicit experimental information," *Nature methods*, vol. 19, no. 11, pp. 1376–1382, 2022.

[15] T. C. Terwilliger, B. K. Poon, P. V. Afonine, C. J. Schlicksup, T. I. Croll, C. Millán, J. S. Richardson, R. J. Read, and P. D. Adams, "Improving alphafold modeling using implicit information from experimental density maps," *bioRxiv*, 2022.

[16] T. C. Terwilliger, D. L. Leibschner, T. Croll, C. J. Williams, A. J. McCoy, B. K. Poon, P. Afonine, R. D. Oeffner, J. S. Richardson, R. J. Read, *et al.*, "Alphafold predictions: great hypotheses but no match for experiment," *bioRxiv*, 2022.

[17] J. Roney, *Evidence for and Applications of Physics-Based Reasoning in AlphaFold*. PhD thesis, 2022.

[18] W. Jorgensen and J. Tirado-Rives, "Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids," *Journal of the American Chemical Society*, vol. 110, pp. 1657–1666, 1988.

[19] D. van der Spoel, E. Lindahl, and B. Hess, "Gromacs: fast, flexible and free," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1701–1719, 2005.

[20] S. A. Hollingsworth and R. O. Dror, "Molecular dynamics simulation for all," *Neuron*, vol. 99, no. 6, pp. 1129–1143, 2018.

[21] M. Karplus and J. A. McCammon, "Molecular dynamics simulations of biomolecules," *Nature structural biology*, vol. 9, no. 9, pp. 646–652, 2002.

[22] N. A. Gonzalez, B. A. Li, and M. E. McCully, "The stability and dynamics of computationally designed proteins," *Protein Engineering, Design and Selection*, vol. 35, 2022.

[23] A. Altis, P. H. Nguyen, R. Hegger, and G. Stock, "Dihedral angle principal component analysis of molecular dynamics simulations," *The Journal of chemical physics*, vol. 126, no. 24, p. 244111, 2007.

[24] M. P. Allen *et al.*, "Introduction to molecular dynamics simulation," *Computational soft matter: from synthetic polymers to proteins*, vol. 23, no. 1, pp. 1–28, 2004.

[25] K. Roy, S. Kar, and R. N. Das, "Chapter 5 - computational chemistry," in *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment* (K. Roy, S. Kar, and R. N. Das, eds.), pp. 151–189, Boston: Academic Press, 2015.

[26] I. Kufareva and R. Abagyan, "Methods of protein structure comparison," *Homology Modeling: Methods and Protocols*, pp. 231–257, 2012.

[27] M. Saito, "Molecular dynamics model structures for the molten globule state of -lactalbumin: aromatic residue clusters I and II," *Protein Engineering, Design and Selection*, vol. 12, pp. 1097–1104, 12 1999.

[28] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "Gromacs 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation," *Journal of chemical theory and computation*, vol. 4, no. 3, pp. 435–447, 2008.

[29] A. Y. Toukmaji and J. A. Board Jr, "Ewald summation techniques in perspective: a survey," *Computer physics communications*, vol. 95, no. 2-3, pp. 73–92, 1996.

[30] R. E. Isele-Holder, W. Mitchell, and A. E. Ismail, "reciprocal-space lattice vector," *The Journal of chemical physics*, vol. 137, no. 17, p. 174107, 2012.

[31] H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten, "Generalized verlet algorithm for efficient molecular dynamics simulations with long-range interactions," *Molecular Simulation*, vol. 6, no. 1-3, pp. 121–142, 1991.

[32] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, and E. Lindahl, "Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1, pp. 19–25, 2015.

[33] B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, *et al.*, "Charmm: the biomolecular simulation program," *Journal of computational chemistry*, vol. 30, no. 10, pp. 1545–1614, 2009.

[34] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, "Comparison of simple potential functions for simulating liquid water," *The Journal of chemical physics*, vol. 79, no. 2, pp. 926–935, 1983.

[35] Y. Zhang, "I-tasser server for protein 3d structure prediction," *BMC bioinformatics*, vol. 9, pp. 1–8, 2008.

[36] K. A. Dill, "Dominant forces in protein folding," *Biochemistry*, vol. 29, no. 31, pp. 7133–7155, 1990.

[37] V. N. Maiorov and G. M. Crippen, "Size-independent comparison of protein three-dimensional structures," *Proteins: Structure, Function, and Bioinformatics*, vol. 22, no. 3, pp. 273–283, 1995.

[38] B. Wang, L. Zhang, T. Dai, Z. Qin, H. Lu, L. Zhang, and F. Zhou, "Liquid–liquid phase separation in human health and diseases," *Signal Transduction and Targeted Therapy*, vol. 6, no. 1, pp. 1–16, 2021.

[39] A. A. Hyman, C. A. Weber, and F. Jülicher, "Liquid-liquid phase separation in biology," *Annual Review of Cell and Developmental Biology*, vol. 30, no. 1, pp. 39–58, 2014. PMID: 25288112.

[40] B. Shen, Z. Chen, C. Yu, T. Chen, M. Shi, and T. Li, "Computational screening of phase-separating proteins," *Genomics, Proteomics Bioinformatics*, vol. 19, no. 1, pp. 13–24, 2021.

[41] S. Sahadevan, K. M. Hembach, E. Tantardini, M. Pérez-Berlanga, M. Hruska-Plochan, S. Megat, J. Weber, P. Schwarz, L. Dupuis, M. D. Robinson, *et al.*, "Synaptic fus accumulation triggers early misregulation of synaptic rnas in a mouse model of als," *Nature communications*, vol. 12, no. 1, pp. 1–17, 2021.

[42] Q. Guo, X. Shi, and X. Wang, "Rna and liquid-liquid phase separation," *Non-coding RNA Research*, vol. 6, no. 2, pp. 92–99, 2021.

[43] K. A. Burke, A. M. Janke, C. L. Rhine, and N. L. Fawzi, "Residue-by-residue view of in vitro fus granules that bind the c-terminal domain of rna polymerase ii," *Molecular cell*, vol. 60, no. 2, pp. 231–241, 2015.

[44] A. Sperling and R. Li, "Repetitive sequences," in *Brenner's Encyclopedia of Genetics (Second Edition)* (S. Maloy and K. Hughes, eds.), pp. 150–154, San Diego: Academic Press, second edition ed., 2013.

[45] D. Srivastava, M. M. Ahmad, M. Shamim, R. Maurya, N. Srivastava, P. Pandey, S. Siddiqui, and M. H. Siddiqui, "Chapter 12 - modulation of gene expression

by microsatellites in microbes," in *New and Future Developments in Microbial Biotechnology and Bioengineering* (H. B. Singh, V. K. Gupta, and S. Jogaiah, eds.), pp. 209–218, Amsterdam: Elsevier, 2019.

[46] E. Gomes and J. Shorter, "The molecular language of membraneless organelles," *Journal of Biological Chemistry*, vol. 294, no. 18, pp. 7115–7127, 2019.

Appendix

The workflow is shown in Figure 5.2:

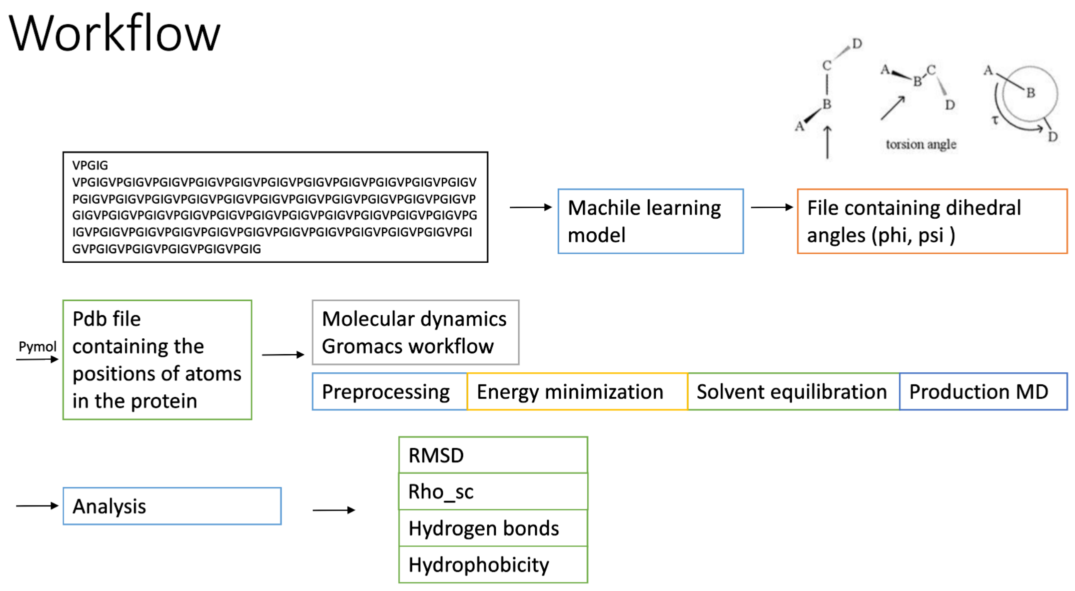

Workflow
VPGIG
VPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGV
PGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVP
GIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPG
IGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGI
GVPGIGVPGIGVPGIGVPGIGVPGIG
Machile learning model
File containing dihedral angles (phi, psi)
torsion angle
Pymol
Pdb file containing the positions of atoms in the protein
Molecular dynamics Gromacs workflow
Preprocessing
Energy minimization
Solvent equilibration
Production MD
Analysis
RMSD
Rho_sc
Hydrogen bonds
Hydrophobicity

Figure 5.2: Workflow.

The process of predicting protein structure starts with FASTA files, strings of amino acid names. A machine learning model is used to predict the ϕ and ψ angles for each residue in the protein sequence. The output is processed with PyMOL software, which converts it into a PDB file format compatible with GROMACS simulations. These PDB files are used to analyze the conformational stability of the proteins, with the steps explained in the following chapter. The simulations were conducted on the supercomputer LUMI, allocating one node per protein (2 GPUs and 64 CPU cores)

The GROMACS commands are specified below:

The command -ignh ignores hydrogens. As the output file is obtained, a .gro file in Gromos87 format is generated, from which the trajectory can be extracted. In this simulation, the water models used were tip3p and forcefield charmm27.

```
gmx_mpi pdb2gmx -f protein.pdb -o protein.gro
tip3p -ff charmm27 -ignh
```

The protein is contained in a box filled with solvent. The command editconf defines the box dimensions. The unit cell chosen is cubic (-bt cubic), and the protein is placed in the center (-c), 1 nm from the edges of the box (-d 1.0):

```
gmx_mpi editconf -f protein.gro
-o protein_newbox.gro -c -d 1.0 -bt cubic
```

The box containing the protein is created and filled with solvate. The solvate chosen is spc216, which is a three-point solvant model. The solvate keeps track of the number of water molecules added and updates the topology file:

```
gmx_mpi solvate -cp protein_newbox.gro
-cs spc216.gro -o protein_solv.gro -p topol.top
```

In the next step, ions are added to the system because living organisms strive for a net charge of zero. A molecular dynamics parameter file (.mdp) together with the coordinates and topology create a .tpr file, which contains all the coordinate parameters in the system on an atomic level.

```
gmx_mpi grompp -f ions.mdp -c protein_solv.gro
-p topol.top -o ions.tpr
```

The geniom module replaces watermolecules with ions in the topology file according to the .tpr file. SOL is chosen for embedding ions. The output file is a .gro file of the system with added ions. Once again, the topology file is processed and updated according to the ions added. Positive (-pname) and negative (-nname) ions are specified, and the command geniom is told to only add the ions needed to neutralize the system (-neutral):

```
echo SOL | gmx_mpi -quiet genion -s ions.tpr -o
protein_solv_ions.gro -p topol.top -pname NA -nname CL
-neutral
```

When the system is setup, energy minimization is performed. The grompp module is once again used to create a .tpr file, assembling topology, structure, and simulation parameters. Energy minimization is obtained by running the em.tpr file through mdrun. Files obtained from mdrun are em.log (ASCII-text log file of the EM process), em.edr (a binary energy file), em.trr (a binary full-precision trajectory), and em.gro (an energy-minimized structure):

```
gmx_mpi grompp -f em.mdp -c protein_solv_ions.gro
-p topol.top -o em.tpr
gmx_mpi -mdrun -deffnm em
```

Before the molecular dynamics simulation can begin, the solvent and ions must be equilibrated around the protein. The orientation of the solvent and ions around the protein is optimized, by bringing the system to a temperature suitable for the simulation. The temperature is equilibrated based on kinetic energy, processed under isothermal-isochoric conditions (NVT, where N = number of particles, V = volume, and T = temperature):

```
gmx_mpi grompp -f nvt.mdp -c em.gro -r em.gro -p topol.top
-o nvt.tpr
gmx_mpi mdrun -deffnm nvt
```

When the temperature is equilibrated, pressure is applied to the system to reach the right density.

```
gmx_mpi grompp -f npt.mdp -c nvt.gro -r nvt.gro -t nvt.cpt
-p topol.top -o npt.tpr
gmx_mpi mdrun -deffnm npt
```

After equilibration, the restraints on the system can be released to start the simulation. The simulation time is specified in the md.mdp file:

```
gmx_mpi grompp -f md.mdp -c npt.gro -t npt.cpt -p topol.top
-o MD_run_name.tpr
gmx_mpi mdrun -deffnm MD_run_name
```

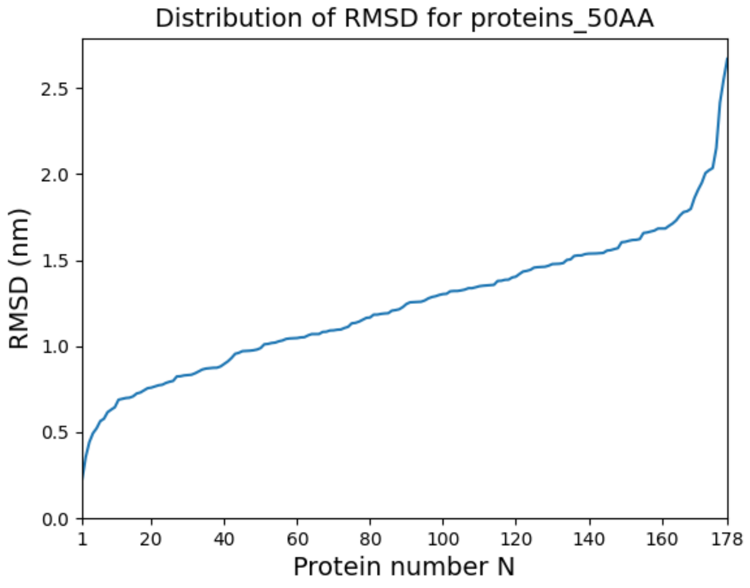
A convenient post-processing module is trjconv, which manages periodicity in the simulation and smooths out irregularities. The simulation will ask which group to center and which one to put in the output file. In this simulation, group number one is chosen, which is the protein:

```
echo 1 1 | gmx_mpi trjconv -s MD_run_name.tpr -f MD_run_name.xtc
-o MD_run_name_noPBC.xtc -pbc mol -center
```

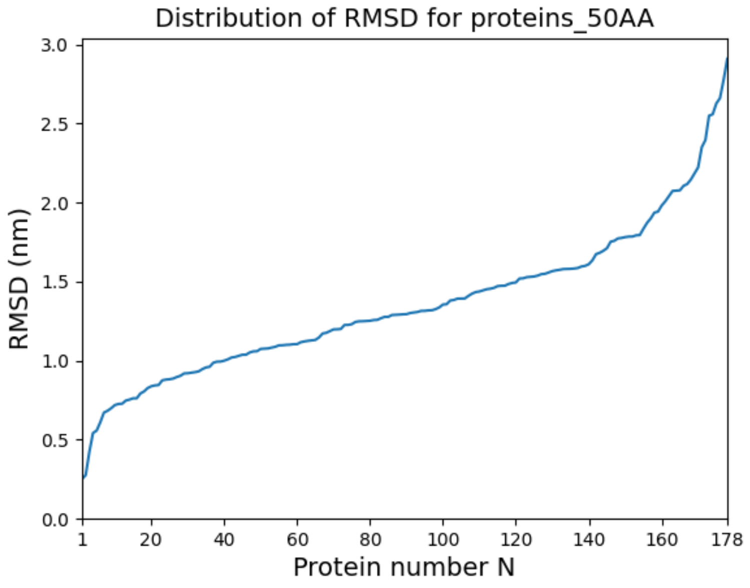
When all of the preceding steps have been completed, the GROMACS rms-command can be used to calculate structure stability. Backbone (group number 4) is chosen for the RMDS calculation. For convenience, the simulation time scale is changed from ps to ns (-tu ns):

```
echo 4 4 | gmx_mpi rms -s MD_run_name.tpr
-f MD_run_name_noPBC.xtc -o MD_run_name_rmsd.xvg -tu ns
```

The results for RMSD will be presented one by one on the next page:

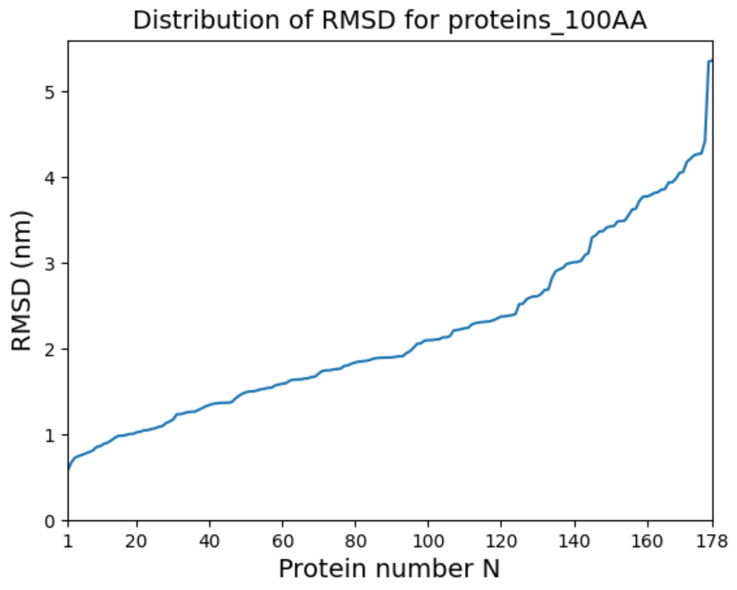


(a) RMSD after 10 ns simulation.

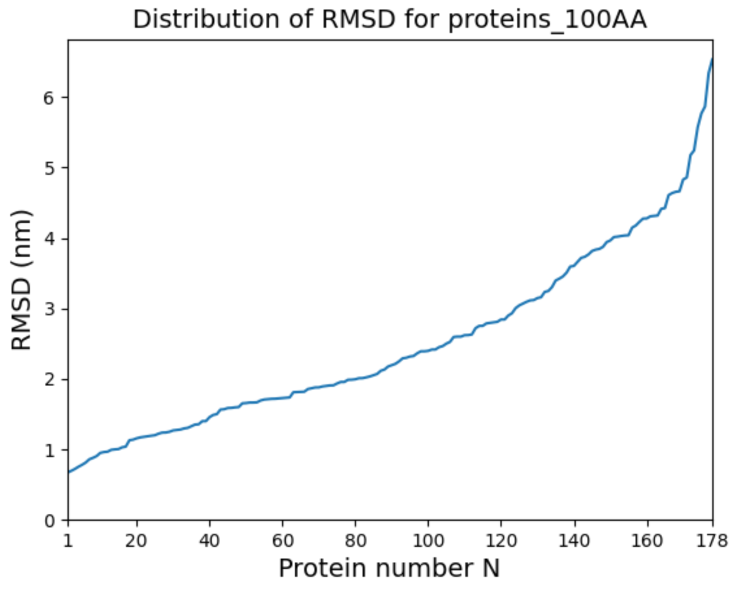


(b) Structure after 20 ns simulation.

Figure 5.3: Proteins built of 50 amino acids.

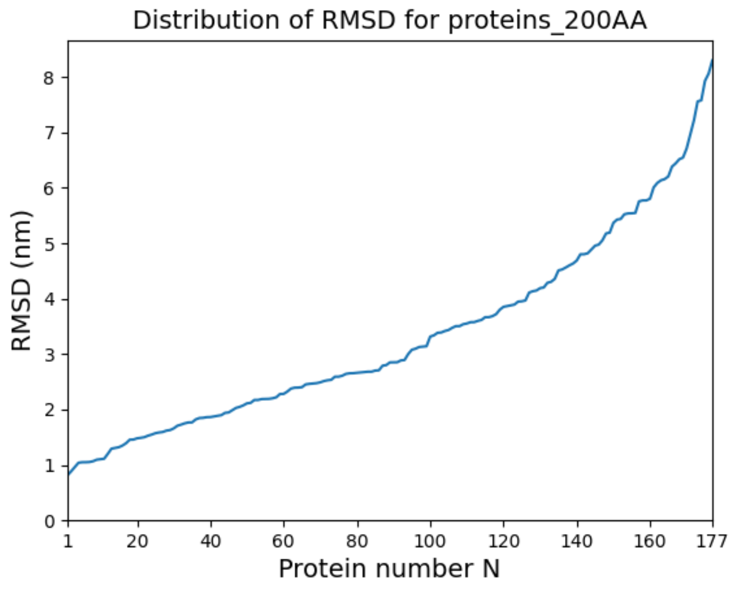


(a) RMSD after 10 ns simulation.

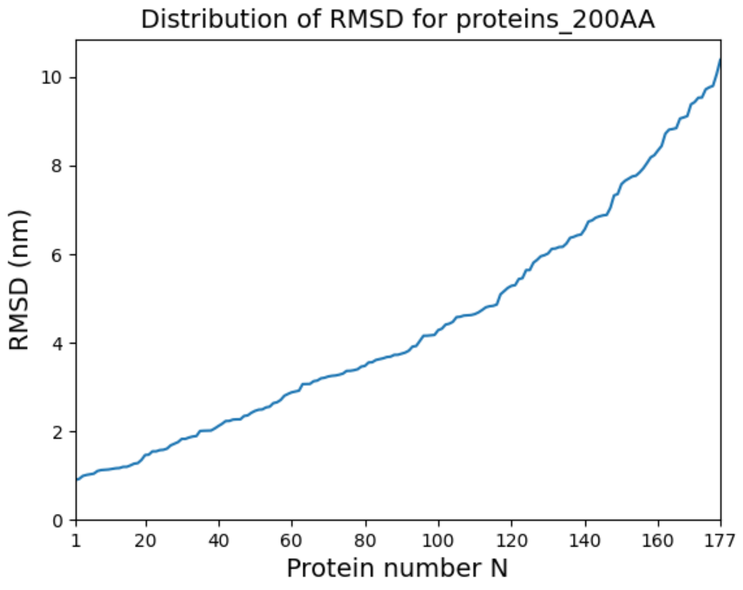


(b) Structure after 20 ns simulation.

Figure 5.4: Proteins built of 100 amino acids.

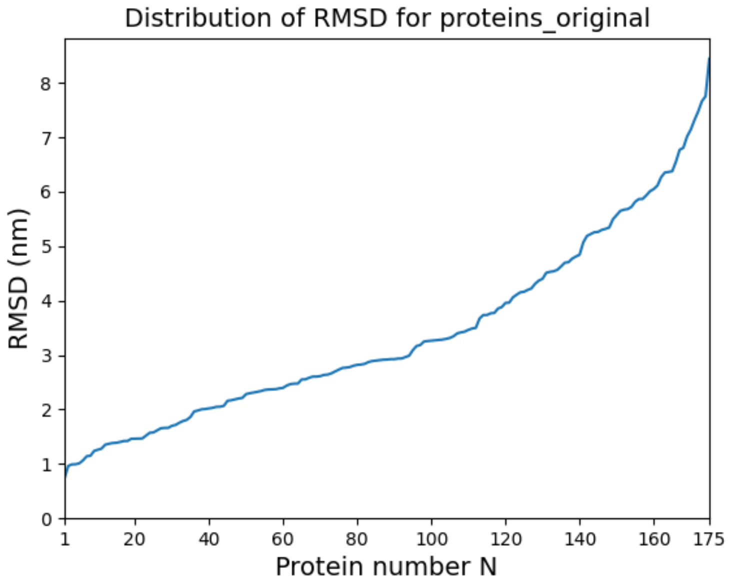


(a) RMSD after 10 ns simulation.

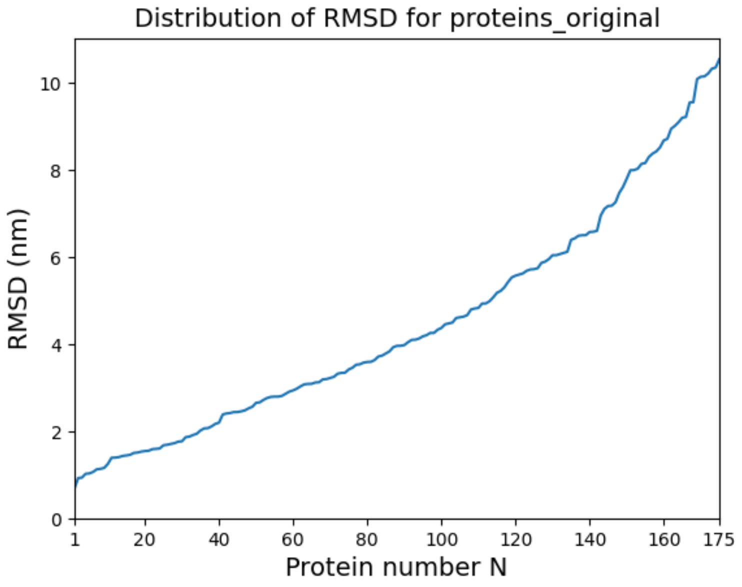


(b) Structure after 20 ns simulation.

Figure 5.5: Proteins built of 200 amino acids.



(a) RMSD after 10 ns simulation.



(b) Structure after 20 ns simulation.

Figure 5.6: Proteins built of 250-300 amino acids.

Svensk sammanfattning

Med dagens teknik är det möjligt att syntetiskt skapa material med önskade egenskaper. Kärnan i den här avhandlingen är digital design av material. Med hjälp av bioteknik är det möjligt att bygga godtyckliga proteiner med önskad aminosyrauppsättning. För forskningsvärlden är det önskvärt att kunna förutsäga hur ett slumpmässigt utvalt protein kommer att vika sig samt skapa nya syntetiska proteiner med valfri tredimensionell struktur. Med kunskap om godtyckliga proteiners egenskaper är det möjligt att skapa syntetiska material enligt endamål. Målet med avhandlingen är att testa noggrannheten hos en AlphaFold-ekvivalent maskininlärningsmodell på syntetiskt konstruerade ideala tandemupprepningsproteiner genom att köra dem genom molekylärdynamiksimuleringar i Gromacs.

Det har länge varit känt att strukturen är kritisk för att ett protein ska kunna fungera som det ska, men det finns ännu mycket som behöver utvecklas för att förstå sambandet mellan sekvens, struktur och funktion. Databaserade beräkningsverktyg behövs som stöd för materialutveckling, eftersom det finns oändligt många kombinationer av aminosyror och det är omöjligt att experimentellt undersöka dem alla. Maskininlärningstekniker är snabbare och mer kostnadseffektiva. Maskininlärningsmodellen kan också föreslå nya sekvenser som inte har testats. Mitt arbete undersöker stabiliteten hos de strukturer som har förutspåtts, vilket senare kan användas som data tillbaka till maskininlärningsmodellen.

År 2020 vann DeepMind Technologies överlägset med AlphaFold2, vilket var ett stort steg för maskininlärningsbaserad förutsägelse av proteinstrukturer. DeepMind tävlade först med AlphaFold1 i CASP13, men AlphaFold2 var en betydande uppgradering. Strukturerna för proteinerna i proteindatabanken PDB bestäms experimentellt med röntgenkristallografi, kärnmagnetisk resonansspektroskopi eller elektronmikroskopi. AlphaFold tränades att hitta mönster bland dessa proteiner och kunde inte ha gjorts utan existerande experimentella data. AlphaFold är bättre på att förutsäga sidokedjor och letar även efter evolutionära kopplingar till proteinerna i sina förutsägelser.

Maskininlärningsalgoritmer matas med befintliga data och använder de för att hitta mönster och förutsäga resultat för godtyckliga indata, dvs. ju mer kända data maskininlärningsmodellen har, desto bättre kommer förutsägelserna att bli. Maskininlärningsmodeller minskar mänskliga påverkningar, vilket kan leda till nya upptäckter. En nackdel med maskininlärningsmodeller är att de förlitar sig på bekanta data,

vilket innebär att det alltid finns en möjlighet att indata är för obekanta för att maskininlärningsmodellen ska kunna bearbeta dem.

AlphaFold har bidragit till att lägga till förutspådda strukturer för det mänskliga proteomet och andra organiska proteiner. Den höga noggrannheten hos AlphaFold baseras på lokal atomprecision med låg standardavvikelse. När man undersöker närmare, uppvisade även dessa dåligt förutspådda regioner. Studier visar att AlphaFold endast kan förutsäga 40 % av det mänskliga proteomet med ett högt konfidensintervall.

Studier visar att AlphaFold-modellen har svårigheter att förutsäga delar som kan bilda alternativa konformationer samt förhållandet mellan olika domäner. Det finns forskning som tyder på att komplettering av experimentell information, t.ex. elektrondensitetskartor och avstånd mellan sidokedjor, ökar prediktionsnoggrannheten. En densitetskarta kan skapas med kristallografiska data från PDB och jämföras med en densitetskarta som erhålls genom att itererade AlphaFold-predicerade strukturerna. Jämförelserna visar att proteiner som ansetts ha en korrekt struktur i många fall består av felaktigt förutsagda domänorienteringar på global nivå och fel i ryggrad och sidokedjor på lokal nivå.

Maskininlärningsmodellen som strukturstabilitetsanalyserna baseras på är jämförbar med AlphaFold. Dessa maskininlärningsförutsägelser är användbara för att förutsäga den initiala strukturen, baserat på dihedriska vinklar, men har fortfarande stora avvikelser jämfört med experimentellt bestämda strukturer och kan för närvarande inte ersätta experiment. Resultaten av denna avhandling visar att de flesta av de maskininlärning förutsagda strukturerna som undersöktes inte är stabila under molekyldynamiksimuleringar.

Gromacs är ett simuleringsprogram för molekylär dynamik som är användbart vid undersökningen av proteinstabilisering. Simuleringarna är baserade på peptider som består av upprepade aminosyraenheter. De ursprungliga peptiderna var 210–300 aminosyror långa. På grund av den stora beräkningsbelastningen reducerades peptiderna initialt till kortare sekvenser för att få en förståelse för deras beteende.

Oönskat beteende i MD-simuleringarna innebär att ML-modellen misslyckas med att förutsäga proteinstrukturen korrekt. Ett sätt att bestämma framgången med maskininlärningsförutsägelsen är att titta på icke-fysiskt beteende i proteinerna. Problem under simuleringarna uppstod huvudsakligen av överlappande atomer. Istället för att utesluta dessa proteiner helt och hållet tillämpades ett extra steg i simuleringen för att separera atomerna från varandra. Dessa steg kan vara användbara i inlärningen av maskininlärningsmodellen. De korrigerade strukturerna är initialt icke-fysikaliska, men kan hjälpa till att justera maskininlärningsmodellen mot mer exakta förutsägelser och ge information om oönskat beteende. Upptäckterna i denna avhandling tyder på att det finns utrymme för förbättringar i ML-modellen. Även om strukturerna överlag är väl förutspådda, kan experimentella data hjälpa till att justera proteinet på en mer detaljerad nivå. När en region är fixerad korrigeras även resten av proteinet, vilket utnyttjas i en iterativ korrigeringsprocess.