UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Molecular Dynamics Simulations Utilizing the GPU

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Christina Krieg

Erstgutachter:	Prof. Dr. Stefan Müller Institut für Computervisualistik, Leiter der Arbeitsgruppe Müller
Zweitgutachter:	Bastian Krayer, M.Sc. Institut für Computervisualistik

Koblenz, im Februar 2020

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☒	☐

.....

(Ort, Datum) (Unterschrift)

Abstract

Molecular dynamics (MD) as a field of molecular modelling has great potential to revolutionize our knowledge and understanding of complex macromolecular structures. Its field of application is huge, reaching from computational chemistry and biology over material sciences to computer-aided drug design. This thesis on one hand provides insights into the underlying physical concepts of molecular dynamics simulations and how they are applied in the MD algorithm, and also briefly illustrates different approaches, as for instance the molecular mechanics and molecular quantum mechanics approaches. On the other hand an own all-atom MD algorithm is implemented utilizing and simplifying a version of the molecular mechanics based AMBER force field published by [Cornell et al. (1995)]. This simulation algorithm is then used to show by the example of oxytocin how individual energy terms of a force field function. As a result it has been observed, that applying the bond stretch forces alone caused the molecule to be compacted first in certain regions and then as a whole, and that with adding more energy terms the molecule got to move with increasing flexibility.

Zusammenfassung

Die Molekulardynamik als Bereich der molekularen Modellierung hat ein großes Potenzial, unser Wissen und Verständnis komplexer makromolekularer Strukturen zu revolutionieren. Ihr Anwendungsgebiet ist groß und reicht von der computergestützten Chemie und Biologie über die Materialforschung bis hin zum computergestützten Wirkstoffentwurf. Diese Bachelorarbeit gibt einerseits einen Einblick in die den Molekulardynamik-Simulationen zugrundeliegenden physikalischen Konzepte sowie in ihre Anwendung im MD-Algorithmus, und skizziert außerdem verschiedene Ansätze, wie z.B. die Ansätze der molekularen Mechanik und der Quantenchemie. Andererseits wird durch Verwendung und Vereinfachung einer von [Cornell et al. (1995)] veröffentlichten Version des auf der Molekularmechanik basierenden AMBER-Kraftfeldes ein eigener all-atom MD-Algorithmus implementiert. Die resultierende Simulation wird anschließend verwendet, um am Beispiel von Oxytocin zu zeigen, wie einzelne Energieterme eines Kraftfeldes funktionieren. Als Ergebnis konnte beobachtet werden, dass sich das Molekül durch die alleinige Anwendung der durch die Valenzschwingung der Bindung entstehenden Kräfte zuerst in bestimmten Regionen und dann als Ganzes verdichtet und sich schließlich durch die Zugabe weiterer Energieterme zunehmend flexibler bewegen kann.

Contents

1 Introduction

The study of macromolecular behavior is an important step in order to find appropriate treatment for diseases linked to protein dysfunction. A deep knowledge about dynamical properties of those molecules can be used to forecast their modification in certain situations such as solvation, force effects or changes in temperature. Moreover it can be deployed to determine equilibrium values or to explorate folding mechanisms of proteins with the goal of predicting their secondary and tertiary structures. The results of protein investigation are used to, for example, acquire knowledge about the misfolding of proteins, which is a suspected partial cause of many diseases, for example Alzheimer's disease, Parkinson's disease, cataract or type 2 diabetes mellitus.

Considering the properties and dynamical behavior of proteins, ribonucleic acid (RNA) and other macromolecules, but also of fluids and metals, *molecular dynamics* (MD) simulations are becoming an increasingly important tool to gain further information of those. The applications of MD are numerous, taking place in the areas of computational biology and chemistry, material science and drug design. MD applications simulate the probable trajectories of particles within molecules with respect to time. They are making it possible to observe materials outside of the laboratories, providing the ability to run an experiment several times under the same conditions, which simplifies its analysis and evaluation and increases information gain of the latter. Additionally they support researchers in designing new drugs by simplifying the process of observing how a certain molecule and its target interact with each other before they are synthesized. To further improve this already useful instrument, the major computing power of graphics processing units (GPUs) can be availed. GPUs have long since ceased to be used only for graphics computation, as they have been found to be likewise valuable for general purpose computation on graphics processing units (GPGPU). For MD simulations this is especially the case because by providing high-level parallelism GPUs enable researchers to simulate larger system sizes on longer time scales than only with central processing units (CPU) [Rovigatti et al. (2015)]. Considering these forceful tools, which are in continuous development, this method of computational simulation may enable researchers to design even better drugs for the curative treatment of protein related diseases.

To launch any kind of MD simulation, it is required to allocate initial positions and velocities for each of the system's particles, where by system is meant the entirety of both the molecule and, if applicable, its surroundings. For instance, the molecule can be encompassed by a gas or fluid. The positions are most often inferred in laboratory from an analysis of the desired molecule and the velocities are derived from a velocity distribution suitable to the specific simulation environment. Instead of autonomously performing a protein analysis, online macromolecule databases can be accessed, which provide data sets containing the data required to simulate certain molecules. After preparations have been made, models of the intra- and intermolecular forces are applied to determine the individual force

effect introduced onto each particle by any other particle. Finally an equation of motion is applied and a mathematical integration is performed, to determine the positions and velocities of the next time step for each particle and to therefore obtain each particle's trajectory.

The aim of this thesis is to provide insights into the underlying physical concepts of molecular dynamics and how they are applied in the MD algorithm, and to further make use of the resulting simulation to show how individual energy terms of an all-atom molecular mechanics force field affect the motion behavior of a molecule to show the functionalities of those terms. *Chapter 2* examines the physical and algorithmic basics of molecular dynamics simulations as well as the basic information concerning proteins. In *Chapter 3* the particular methods used in this thesis are portrayed in detail. This includes the inputs for the simulation, the force field and mathematical integration algorithm utilized, as well as the implementation of those in the compute shader and in the draw shaders. *Chapter 4* then discusses the results of the simulation, whereas *Chapter 5* draws a conclusion and gives an outlook on future work.

2 Basics

This chapter deals with the basic knowledge needed to write a simple MD application. *Chapter 2.1* examines the underlying physical principles to calculate the trajectory of a molecule. In *Chapter 2.2* the basic information about proteins is described, since the algorithm implemented for the purpose of this thesis utilizes a force field (*Chapter 2.3.3*) optimized for proteins, nucleic acids and organic molecules and simulates the trajectory of a peptide. The general and algorithmic basics of molecular dynamics are stated in *Chapter 2.3*.

2.1 Physical Basics

In this chapter the underlying physical principles employed in molecular dynamics applications are examined. *Chapter 2.1.1* makes a connection between experimental assemblies in the laboratory and the concept of statistical ensembles, or more specific thermodynamic ensembles, which is utilized in MD to ensure statistical correctness of simulations. *Chapter 2.1.2* describes the calculation of the energy and force of a physical system, whereas in *Chapter 2.1.3* the equations of motion needed to generate a trajectory using classical mechanics are explained in detail.

2.1.1 Thermodynamic Ensembles

Each molecular dynamics simulation is fitted to a certain *statistical* or more precisely *thermodynamic ensemble*, which is described in this chapter. To derive averages of certain properties of a system over its trajectory, the system would have to be observed over a sufficiently long period of time. However, typical systems of interest are too big and too diverse in behavior [Hill (1986)], so that the duration of observation would have to be excessively long. A thermodynamic ensemble encapsulates a set of physical systems organized under the same thermodynamic conditions. The average of a certain property of a system can be derived by calculating its average for the system's ensemble [Hill (1986)]. This means that several relatively short simulations can be rolled out under the same conditions to derive an average value for a system. Although an ensemble can never be fully investigated, the correctness of ensemble averages is verified by the *ergodic hypothesis*. The hypothesis signifies, that thermodynamic systems act randomly and therefore every possible state of the system which is energetically accessible, will be accessed.

Ensembles are generated by setting certain *ensemble properties* as constant, depending on the experimental assembly in which the system is to be investigated, e.g. if the system is closed or insulated. Each thermodynamic ensemble corresponds to a determined experimental assembly (Figure 1). The ensemble properties and several ensembles are outlined below.

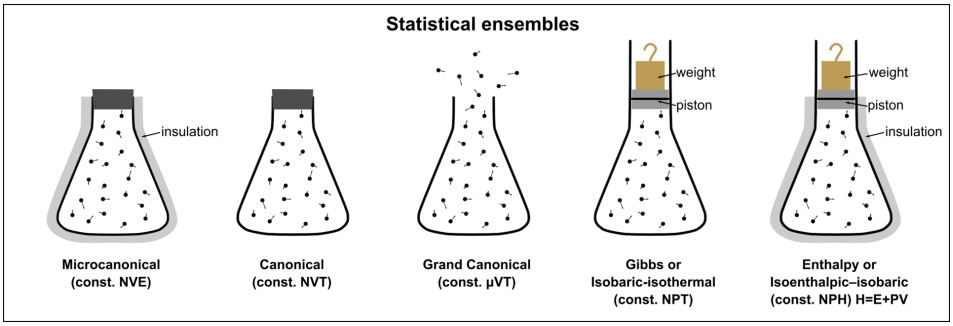


Figure 1: Statistical Ensembles / Experimental Assemblies (Figure by [Nzjacobmartin (2017)])

To specify the properties of physical ensembles there are seven variables of which three can be set as constant. Six of these variables form pairs, where only one out of each pair may be set as constant (also see Table 1). This is due to the fact that the other variable of the pair can be calculated depending on the other. For the first pair, the number of particles is stated by N, while μ is the chemical potential per particle type. Here, by particle is meant either a single atom or a compound of atoms. In the next pair, P describes the pressure and V the volume of the system, whereas in the last T and E quantify its temperature and energy. The behavior of the seventh ensemble parameter, the entropy S of the system, varies depending on the considered ensemble.

	Either	Or
Pairs	N	μ
	V	P
	E	T

Table 1: Ensemble Properties

Several ensembles can be derived from the possible combinations, two frequently used ensembles are described hereafter and more are mentioned below. For molecular dynamics simulations, one widely applied ensemble, on which this thesis is focused, is the *microcanonical* ensemble. It represents an idealized experimental assembly that consists of a closed, isolated system which is unable to exchange energy with its environment. It is also called the NVE-ensemble, because it consists of systems with a constant value for N, V and E. The second ensemble to be mentioned is the *canonical* ensemble, also called NVT-ensemble. Its systems also have a constant value for N and V, however other than the NVE-ensemble it is not isolated, but able to exchange energy with a reservoir. Since the canonical system and its reservoir are in thermal equilibrium, the system's own energy may fluctuate and its temperature T is maintained. This ensemble represents an experimental assembly that consists of a closed system in a heat bath. Further relevant

ensembles are the *grandcanonical* or μVT-ensemble, the *isobaric-isothermal* or NPT-ensemble and the *isoenthalpic-isobaric* or NPH-ensemble, where H is the enthalpy with $H = E + PV$. The first describes an open system with fixed volume, the second a closed system exposed to pressure, and the last a closed, insulated system exposed to pressure.

2.1.2 Energy and Force

There are several forms of energy, for example potential, kinetic, electric or chemical energy, which can be converted into each other. Albeit, in classical mechanics (see *Chapter 2.1.3*) the equations of motion only consider the potential (Equation (1)) and kinetic (Equation (2)) energy of the system.

$$V(r_1, \ ..., \ r_N) = \sum_{i<j} u(r_{ij}) \tag{1}$$

$$T = \sum_i \frac{1}{2} m_i \cdot \dot{r_i}^2 \tag{2}$$

Where $r_1, \ ..., \ r_N$ describe the position, $\dot{r_1}, \ ..., \ \dot{r_N}$ the velocity and $m_1, \ ..., \ m_N$ the mass of each of the system's particles. Furthermore $u(r_{ij})$ defines the *inter-particle potential*, a function of the distance r_{ij} between particle i and j. The inter-particle potential also depends on the types of the interacting particles. It contains for example *binding energy*, *van der Waals* (vdW) *interaction* and *electrostatic interaction* between the two particles. The total energy E of a system equals the sum of potential and kinetic energy (Equation (3)).

$$E = T + V(r). \tag{3}$$

For closed, insulated systems, i.e. systems of the microcanonical ensemble, the result of Equation (3) has to be conserved over time, while both of its components may vary. This is required due to the energy conservation law.

To calculate the trajectory of a particle, its potential force is needed. As stated in Equation (4), the potential force introduced onto a particle is usually a three-dimensional vector which can be derived from Equation (1) by taking its negative gradient, i.e. its negative derivative with respect to the particle's position r_i.

$$F_i = -\nabla V_i = -\frac{\delta V_i}{\delta r_i}. \tag{4}$$

2.1.3 Classical Mechanics

Molecular dynamics relies on several physical models and systems. Since MD is supposed to simulate trajectories of various particles, the most central underlying concepts concern the equations of motion, which have to be applied on each particle

of the system. The equation of motion for the microcanonical ensemble is based on *Newton's three laws of motion* which apply to inertial systems:

1. Unless an atom is influenced by a force introduced onto it, it moves either at a constant velocity or rests.

2. If a force is applied on a body of constant mass, the body is accelerated in the direction of this force.

3. For two atoms i and j, if $F_i(j)$ is the force introduced onto atom i by atom j. Then the force introduced onto atom j by atom i has the same magnitude but takes the opposite direction ($F_j(i) = -F_i(j)$).

Assuming these laws to be true, Newton's second law can be formulated as in equation Equation (5), where m is an atom's constant mass and $\ddot{r}$ its acceleration at the current point in time. An equation of motion can be inferred from this equation by shifting it to $\ddot{r}$ (Equation (6)):

$$F = m\ddot{r} \tag{5}$$

$$\ddot{r} = \frac{F}{m} \tag{6}$$

According to Equation (6), the acceleration of a particle is determined by the force introduced onto it divided by its constant mass. The force F is usually a three-dimensional vector given by Equation (4) determining the direction of acceleration.

It may be proven that Equation (6) meets the requirement of microcanonical ensembles to preserve the energy of the system. Therefor it has to be transformed into another equation of motion, derived from a function called the *Hamiltonian* [Tuckerman and Martyna (2000)]. It is a transformation of the *Lagrangian* function which is also a function of motion.

$$L(r,\dot{r}) = T - V = \sum_i \frac{1}{2} m_i \dot{r_i}^2 - \sum_{i<j} u(r_{ij}). \tag{7}$$

$$H(r,p;V) = \sum_i \dot{r}_i \cdot p_i - L, \tag{8}$$

$$p_i = \frac{\delta L(r,\dot{r})}{\delta \dot{r}_i} = m_i \dot{r}_i. \tag{9}$$

$$H(r,p;V) = \sum_i \frac{{p_i}^2}{2m_i} + \sum_{i<j} u(r_{ij}) = T + V = E. \tag{10}$$

$$\dot{r}_i = \frac{\delta H}{\delta p_i} = \frac{p_i}{m_i}, \tag{11}$$

$$\dot{p_i} = -\frac{\delta H}{\delta r_i} = -\frac{\delta V}{\delta r_i} = F_i(r_1,\ ...,\ r_N). \tag{12}$$

Equation (7) is the Lagrangian for an N-particle system, while Equation (8) is its Hamiltonian, which is dependent on the Lagrangian. Inserting Equation (7) into Equation (8) yields Equation (10), from which the two Equations (11) and (12) can be derived, the Hamiltonian equations of motion. For these equations, $r = r_1,\ ...,\ r_N$, $\dot{r} = \dot{r_1},\ ...,\ \dot{r_N}$ and $p = p_1,\ ...,\ p_N$ and Equation (9) specifies the momenta of the system. As shown in Equation (10) the Hamiltonian is equal to the energy of the system. An important characteristic of Equations (11) and (12) is that they conserve the Hamiltonian [Tuckerman and Martyna (2000)], meaning they conserve the total energy of the system. This can be proven by taking the time derivative of the Hamiltonian, which is zero. Additionally the Hamiltonian equations of motion have the quality of being *time reversible* [Tuckerman and Martyna (2000)], i.e. they can be applied in the same way, if the time t is transformed to $-t$.

To show that the Hamiltonian equations of motion are a transformation of Newton's equation of motion and thus prove that the latter is applicable for the microcanonical ensemble, one may take the time derivative of both sides of Equation (11). The derivative of the left side can be seen in Equation (13). For the right side, applying the quotient rule and substituting with Equation (12) at the end of the calculation delivers the derivative stated in Equation (14). Equating these two yields Equation (6).

$$(\dot{r_i})' = \ddot{r_i} \tag{13}$$

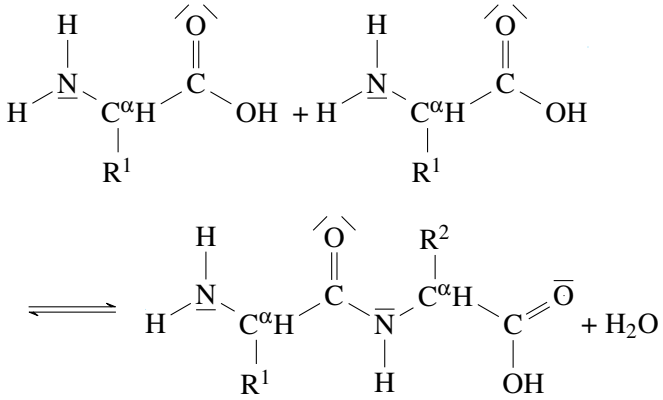
$$\left(\frac{p_i}{m_i}\right)' = \frac{m_i \cdot p_i' - p_i \cdot m_i'}{m_i^2} = \frac{m_i \cdot \dot{p_i} - p_i \cdot 0}{m_i^2} = \frac{m_i \cdot \dot{p_i}}{m_i^2} = \frac{\dot{p_i}}{m_i} = \frac{F_i}{m_i} \tag{14}$$

As mentioned earlier, Newton's and Hamilton's equation of motion (see Equations (6), (11) and (12)) are used for experiments in the microcanonical ensemble. Other ensembles require different equations of motion. Observing for instance the NVT-ensemble, energy fluctuations have to be produced, because they are needed to keep the temperature constant. One possibility to achieve this, is to append a collision term to the Hamiltonian equation of motion as an additional, stochastic force acting on a particle as stated by [Andersen (1980)].

At last, a mathematical integration has to be performed to solve the equations of motion for several points in time. This means, that the new position and velocity for each moment in time are derived by integrating those of the previous moment in time. For this purpose, a time step must be chosen; the shorter the time step, the more precise the integration.

2.2 Proteins

Proteins are crystal-like macromolecular compounds consisting of amino acids which contain the two functional groups COOH, the carboxyl group, and NH_2, the amino group. Amino acid molecules are zwitterions, i.e. they always have a positive and a negative charge. The charged functional groups are the carboxylate group COO^- and the ammonium group NH_3^+. Several amino acids can be bound to each other by forming peptide bonds which originate from the condensation of one amino acid's carboxyl group with another amino acid's amino group [Branden and Tooze (2012)]. The following chemical equation expresses the peptide bonding of two glycine amino acids.



A peptide consisting of two amino acid molecules is called dipeptide, whereas peptides with 2-9 amino acid components are called oligo- and those with more than 9 are called polypeptides. Proteins are peptides built of more than 100 amino acids.

The description of a protein's spatial structure is organized in primary, secondary, tertiary and quaternary structure [Branden and Tooze (2012)], as can be seen in Figure 2. The primary structure of a protein is the sequence of its amino acids. Regular peptide bonds and eventual hydrogen bridge bonds between close peptide bonds define the secondary structure, where two common examples are the α-helix and the β-sheet structure. Tertiary structure is the structure generated by intermolecular forces between side chains of different amino acids. If a protein consists of several polypeptide chains they form a quaternary structure.

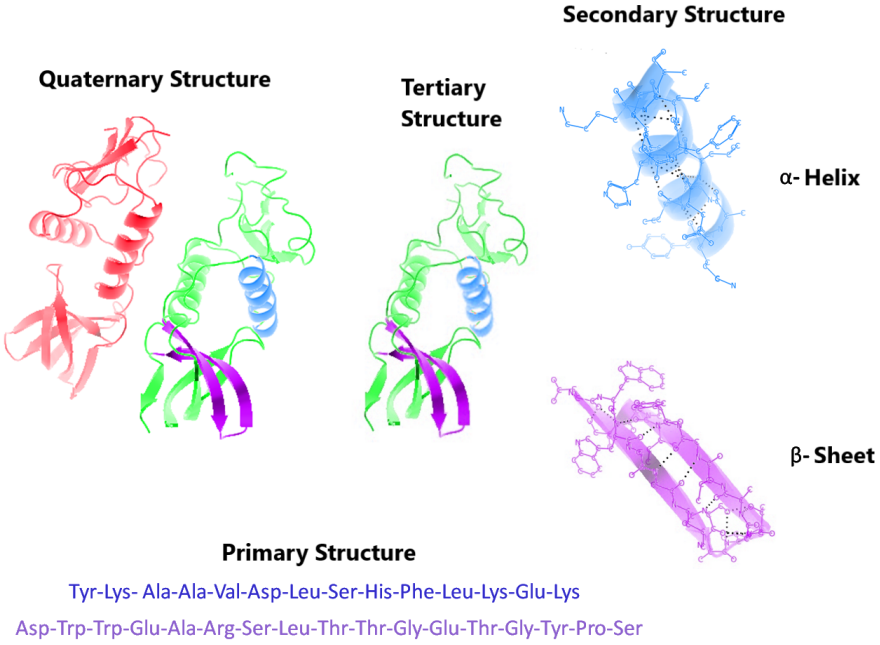


Figure 2: Protein Structures (Figure derived from [Holger87 (2012)] and then translated into English)

2.3 Molecular Dynamics

The various MD methods are associated with *molecular modelling*, which is the set of all methods used to model the behavior of molecules. The field of molecular dynamics simulations in particular deals with the computational, graphical representation of the dynamical behavior of molecules. Its fields of application are *computational chemistry* and *computational biology*, which focus on the resolution of chemical and biological problems through computer simulation, as well as *material science*, which deals with the analysis and construction of new materials, and *computer-aided drug design*, which is concerned with the observation of the intermolecular interaction between a designed drug and its biological target.

As mentioned at a few points in this thesis, there are various methods associated with molecular dynamics simulations depending on the particular interest. Some of those are briefly described in *Chapter 2.3.1*. The input data needed to start an MD application is stated in *Chapter 2.3.2*. After preparations have been made, the molecular dynamics algorithm starts. The common algorithm is described in the following lines. At first, the forces introduced onto each atom by any other of the system's atoms have to be calculated. The forces are derived from the energy functions (see *Chapters 2.1.2 and 2.3.3*), which are functions of the types and the distance between the two considered particles. The complete force introduced onto an atom is then used to calculate its acceleration by using Equation (6). Afterwards

a mathematical integration is performed to solve the equation of motion for a sequence of time steps by integrating the position and velocity of the previous time step to yield those of the current one. For force calculation either *molecular mechanics* or *quantum chemistry* are utilized. Since this thesis is focused on molecular mechanics, it is described in *Chapter 2.3.3*.

2.3.1 Approaches

There are various methods associated with MD simulations depending on the particular interest. Some of these are briefly described in this chapter.

To begin with, MD programs differ in their level of detail regarding the atoms of the simulated molecule and its surroundings. Molecules can be modeled in an *atomistic* or also called *all-atom* way where each atom is represented as one particle, as for example in [Cornell et al. (1995)] whose force field was utilized for the purpose of this thesis. All-atom simulations are computationally expensive since each atom has its own distinct position, a type, radius, and so forth and has to be involved in force calculations. [Kmiecik et al. (2016)] says, that even if a supercomputer designed for the purpose of all-atom MD simulations is used, only for small and fast folding protein's the folding processes can be simulated. In [Zwier and Chong (2010)] several methods for all-atom simulations are discussed. Alternatively molecules can be modeled in a *coarse-grained* manner, where several atoms are merged to atom groups to be handled as one particle. Since the trajectories of fewer particles have to be calculate, the coarse-grained approach is more affordable than the atomistic one considering computational cost and the computing power which may be availed [Kmiecik et al. (2016)].

MD simulations further make use of *quantum mechanics* or alternatively *classical mechanics* (*Chapter 2.1.3*) in determining the potential energy function for the MD algorithm. Using quantum mechanics for molecular dynamics simulations or for molecular modelling in general is called quantum chemistry or alternatively *molecular quantum mechanics* (QM). [Micha and Burghardt (2007)] gives a broad overview of established methods and approaches in development. With the QM method, the forces are calculated by precisely treating the electronic behavior of each atom using formulas derived from theoretical knowledge and considerations. Such a theoretical approach is also called an ab initio approach and thus using quantum chemistry techniques for the derivation of an energy function in the context of MD is also known as *ab initio molecular dynamics* (AIMD). This approach usually utilizes the electronic molecular Hamiltonian to solve the time-dependent Schrödinger equation which describes the atom's wave function [Linderberg and Öhrn (2004)]. The Schrödinger equation in quantum mechanics is analog to Newton's second law (Equation (6)) in classical mechanics (*Chapter 2.1.3*). Assigning energy functions using classical mechanics is also referred to as molecular mechanics (MM). In this method the energy function is usually called a "force field" which is empirically fitted to calculate the approximate behavior of a class of molecules, e.g. proteins. Common force fields are the AMBER (**A**ssisted **M**odel **B**uilding

with **E**nergy **R**efinement) [Weiner et al. (1986)] [Cornell et al. (1995)], CHARMM (**C**hemistry at **Har**vard **M**acromolecular **M**echanics) [Brooks et al. (1983)] and GROMACS (**Gro**ningen **Ma**chine for **C**hemical **S**imulations) [Van Der Spoel et al. (2005)] force fields. In comparison to the QM method which always calculates the energy of the specific molecule being simulated, MM saves computational cost and is an efficient and sufficiently accurate technique in many cases. The parameters needed as input for this energy function are derived either empirically or using ab initio techniques. Since this thesis is focused on MM, the method is described further in *Chapter 2.3.3*. Another popular approach is deploying a hybrid of both aforementioned approaches which is called QM/MM molecular dynamics. It can be a considerable solution for simulations where MM is sufficient for most of the simulated system but where a small region of interest in the molecule needs to be observed more accurately. [Senn and Thiel (2009)] gives an overview of the basics, optimization possibilities and applications of QM/MM simulations. All variants may be applied for both all-atom and coarse-grained simulations. MM and QM are also subsets of molecular modelling.

Another important factor is the surrounding of the observed molecule. One possibility is situating the molecule in solvent. This may be done with *explicit solvent* where a water model is needed to simulate the water molecules as individual particles too, in an either atomistic or coarse-grained way. An alternative method is establishing a mathematical formula so that the influence of the surrounding water can be derived implicitly which is called an *implicit solvent simulation*. Water molecules are not modeled as particles, but are incorporated by adding a formula to the energy function.

Further parameters in which MD algorithms differ are for instance the boundary conditions, e.g. cut-off boundary or periodic boundary, the mathematical integration method, for example the Verlet algorithm or the leapfrog algorithm [Leimkuhler et al. (1996)] may be applied, or the time step Δt chosen for the integration may be adjusted where the results are more accurate the smaller Δt is.

2.3.2 Inputs

To launch any MD programm, the type, initial position and initial velocity of each particle of the system to be simulated are required as input, where by system is meant the entirety of both the observed molecule and, if applicable, its surroundings. For instance, the molecule can be encompassed by a gas or fluid. This data is the basis to specify all other inputs and to perform every partial calculation to finally yield the trajectory of the system.

The atomic structure and consequently the positions and atom types are derived experimentally. The structures of proteins for instance are obtained from a *crystal structure analysis* which can be carried out using different techniques, for example *X-ray crystallography*. Coordinates are typically represented in ångström (Å). Instead of autonomously performing such an analysis, online databases, e.g.

the RCSB website[1] or the ChemSpider website[2], can be accessed, which provide data sets containing all data required to simulate certain molecules. The particles' types are then used to specify several of the particles' properties, for example their covalent radii, masses or bonds between particles.

For the calculation of the initial velocities, in many cases an *ideal gas* is assumed. According to the kinetic theory of gases, whose principles are for example stated in [Loeb (2004)], the average kinetic energy of a particle directly depends on temperature in an ideal gas (Equation (17)). Moreover, kinetic energy is a function of velocity (Equation (2)). The normally distributed *Maxwell-Boltzmann distribution*, whose principles are summarized by for instance [Hernandez (2017)], utilizes these relations, describing the spreading of the magnitudes of the velocities' components in an ideal gas. Its mean and standard deviation can be obtained from Equations (15) and (16).

$$\mu = 0 \tag{15}$$

$$\sigma = \sqrt{\frac{k_B T}{m}} = \sqrt{\frac{RT}{M}} \tag{16}$$

In these two equations $k_B = 1.380649 * 10^{-23} \frac{J}{K}$ defines the Boltzmann constant, $R = 8.314462618 \frac{J}{mol \cdot K}$ the ideal gas constant, m the particle mass and M the molar mass. The temperature T is yielded by Equation (17). If the mass is given in molar mass instead of particle mass, k_B is substituted by R.

$$T_{\text{current}} = \frac{2}{3} \cdot \frac{\overline{E_{\text{kin}}}}{k_B} = \frac{2}{3} \cdot \frac{E_{\text{kin}}}{N k_B} \tag{17}$$

A value from this distribution is generated for all three components of each atom's velocity vector. The velocity is represented in ångström per seconds (Å/s).

2.3.3 Molecular Mechanics

Molecular dynamics and molecular modelling in general use different methods to determine the calculation of energy and force and accordingly motion. Assigning energy functions using classical mechanics is also referred to as molecular mechanics. MM energy functions are usually referred to as force fields. Such a force field consists of several terms. Let Equation (18) be the potential energy of the considered system. Then Equation (19) represents the covalent or also called bonded energy of the system, whereas Equation (20) represents the non-bonded energy of the system [Rappe and Casewit (1997)]. The bonded energy term (Equation (19)) consists of a bond and angle term as well as a term named dihedral or also torsion term. The first

[1] **R**esearch **C**ollaboratory for **S**tructural **B**ioinformatics: `https://www.rcsb.org`

[2] ChemSpider by the Royal Society of Chemistry: `http://www.chemspider.com`

two are principally computed as harmonic oscillators [Rappe and Casewit (1997)]. Some force fields also add a separate term for out-of-plane angles. The non-bonded energy term (Equation (20)) is composed of one term each for van der Waals and electrostatic energies. The first is usually modelled by the *Lennard-Jones potential*, while the latter is based on *Coloumb's law* [Rappe and Casewit (1997)]. To finally receive a particle's force, the negative gradient of its energy function has to be calculated, as mentioned in *Chapter 2.1.2*. Finally, the trajectory is yielded using Newton's equation of motion, see Equation (6), and performing a mathematical integration.

$$V(r) = E_{\text{pot}} = E_{\text{bonded}} + E_{\text{nonbonded}} \tag{18}$$

$$E_{\text{bonded}} = E_{\text{bond}} + E_{\text{angle}} + E_{\text{dihedral}} \left(+ E_{\text{out of plane}}\right) \tag{19}$$

$$E_{\text{nonbonded}} = E_{\text{van der Waals}} + E_{\text{electrostatic}} \tag{20}$$

The concrete formula depends on the chosen *force field*. A force field is the unity of an energy function and its parameters. Often either of them alone is also referred to as force field. Force fields are often specialized, e.g. to a certain group of molecules. The parameters are either derived empirically by fitting them to certain molecules observed in laboratory or by utilizing ab initio techniques. These and more information about the derivation of molecular mechanics parameters can be extracted from [Allinger et al. (1994)]. Furthermore force fields differ in the atom types for which parameters are calculated. An atom type is usually defined by the element of the atom and by the atom's neighbor atoms in the molecule. The energy function and parameters of distinct force fields are usually not compatible, because force fields are only internally consistent.

3 Method and Implementation

In the scope of this thesis it is to be shown which impact distinct terms of a molecular mechanics inter-particle potential have on the trajectory of a molecule. For this purpose a version of the molecular mechanics based AMBER force field for proteins, nucleic acids and organic molecules, presented by [Cornell et al. (1995)] was utilized. I simplified the energy function and parameters of this force field and spared implementing a water model, although the force field is suitable for explicit solvent simulations. This was done to limit the scope due to the time constraints of a bachelor thesis. The molecule whose trajectory is to be shown is an oligopeptide called oxytocin. It was chosen because its small size facilitates data preparation and reduces computational cost. The implementation is performed in OpenGL and OpenGL Shading Language (GLSL). Furthermore the CVK 3 from the "AG Computergrafik" at Universität Koblenz-Landau is utilized.

Inputs and their designation in the implementation are stated in *Chapter 3.1*. The complete force field, its parameters and the simplifications generated here are outlined in *Chapter 3.2*. That chapter also describes the force calculation derived from the AMBER energy function, and the implementation of the force field, including the GPU specific features of the implementation. *Chapter 3.3* completes the description of the MD implementation by explaining the leapfrog integration algorithm which I utilized for the time step integration, and finally *Chapter 3.4* illustrates the drawing procedure. It is noteworthy that for the MD implementation some of the constants as well as some calculation results have to be converted into different units. This is due to the fact that the S.I. unit for length is *m* while the length unit used in molecular modelling is ångström. Furthermore, data like for instance energy is given per mol instead of per atom. This topic will not be explained further in this thesis.

3.1 Inputs

To begin with, the simulation parameters utilized by the implementation are stated in *Chapter 3.1.1*. Next, the initial position and type data of the considered peptide are derived from a .mol file, downloaded from the ChemSpider website. The ChemSpider ID of that peptide is "388434" [Royal Society of Chemistry (RSC) (nd)]. The file then was converted to a format similar to the XYZ format, to load only the data required for this project into the OpenGL environment. Initial positions and types as well as file conversion and loading are described in *Chapter 3.1.2*. Afterwards, *Chapter 3.1.3* describes the calculation of initial velocities, derived from the Maxwell-Boltzmann distribution. Finally, inference of other inputs from the initial positions and types is elucidated in *Chapter 3.1.4*. All created input arrays are loaded into buffers and serve as input for the shaders.

3.1.1 Simulation Parameters

The number of atoms N belonging to the system corresponds to the number of atoms of oxytocin, since only this molecule is simulated, without any encompassing substance. For the initial temperature T_{initial} I choose a value of $309.75K$, because it is near the average body temperature of a human being and therefore in the temperature range of a protein's or peptide's natural environment. The bond threshold k used here has a value of 1.2, whereas the time step Δt for numerical integration has a magnitude of 0.01fs. The latter is chosen smaller than it would be ordinarily, to compensate for some of the inaccuracies induced by the simplifications made on the force field. Because of the time limitation of a bachelor thesis no bounding volume was created, although the energy and force calculations are performed for a microcanonical ensemble. Since the system has no spacial boundary, the energy also fluctuates and therefore N is the only determined ensemble property. The simulation parameters chosen may also be extracted from Table 2.

Parameter	Value
N	135
T_{initial}	$309.65K$
$k_{\text{threshold}}$	1.2
Δt	$0.01fs$

Table 2: Simulation Parameters

3.1.2 Initial Positions and Types

To prepare the .mol file for extracting the position and type data, first all lines that do not contain coordinate information are deleted from the file. Then the spaces in front of the first column are deleted and the spaces seperating the columns are changed to tabulators by using pearl-style regular expressions. The regular expression "^\s+" is replaced by "" and "\h+" by "\t". This makes the file readable for the following Bash command which extracts the file's 1st, 2nd, 3rd and 4th column and writes them to a new file.

```
cut -f1,2,3,4 ChemSpiderID.mol > ChemSpiderID_tmp.mol
```

After this adjustment the types are replaced by integers. This enables to generate buffers containing information about atoms of certain elements, and accessing this information with the integer value of the particular element as index. The replacement is performed by the following Bash command.

```
cat ChemSpiderID_tmp.mol | sed "s/H$/0/" |
sed "s/C$/1/" | sed "s/N$/2/" | sed "s/O$/3/" |
sed "s/S$/4/" > ChemSpiderID.xyze
```

We call the file ChemSpiderID.xyze, with x, y and z standing for the coordinates and e for the element. The integer values represent the corresponding element in

the MD simulation of this thesis, see Table 3.

String	Integer
H	0
C	1
N	2
O	3
S	4

Table 3: Type Replacement

To finally load the .xyze file into the OpenGL environment and extract the information, an `std::ifstream` is used. It is assumed that every line consists of four entries: the atom's x-, y- and z-coordinate, represented in Å, and its type, an integer. Each line is read token by token, writing its contents into a position- and type-array respectively. Each entry of the position array consists of a four element vector containing floating point numbers, whereas the type array contains integers. Each atom of the imported molecule has its own position and type and is therefore simulated as a distinct particle, since an all-atom MD simulation is performed in this thesis. As a consequence, both arrays have a length of N, the number of atoms in the peptide to be simulated. To serve as input for the compute shader, they are each stored in an SSBO and the position is furthermore stored in an array buffer to be accessed in the vertex shader.

3.1.3 Initial Velocities

The initial velocities, as mentioned in *Chapter 2.3.2*, are derived from the Maxwell-Boltzmann distribution. While assuming an ideal gas, this distribution utilizes the direct dependence of kinetic energy and temperature, whereby kinetic energy is a function of velocity as can be seen in Equation (2).

To generate random values of the Maxwell-Boltzmann velocity distribution the following functions are used. The first function, `std::normal_distribution`, creates a normal distribution which is constructed by giving a mean and standard deviation as input. In this case, the mean and standard deviation of the Maxwell-Boltzmann velocity distribution are its input (*Chapter 2.3.2*). Uniformly-distributed, non-deterministic integer random numbers are then produced by the function `std::random_device` as seed for the third function, `std::mt19937`, which is the random number engine. Finally, for all three components of each atom's velocity vector one random value of the normal distribution is fetched.

The temperature in σ is not the fixed temperature of the system, but rather its initialization temperature, since we are exhibiting a simulation in the microcanonical ensemble where energy is fixed but temperature fluctuates. In the end, the final values are stored in an array of N elements filled with four element vectors which is then stored in an SSBO to serve as input for the compute shader.

3.1.4 Other Inputs

As mentioned before, several additional inputs can be inferred from the types and initial coordinates. First, an atom's mass may be derived from its type. It is chosen as it can be seen in the second column of Table 4, taking the values of [Wieser et al. (2013)]. Where intervals are given, I choose the lowest values of these intervals. The masses of the five elements used in this thesis' simulation are stored in an array of five elements containing float values. They are represented in unified atomic mass units. Subsequently another array of N elements is filled with the atomic mass of each atom by accessing the five-element array with the particular type value of the current atom as index. To serve as input for the compute shader this information is then stored in a shader storage buffer object (SSBO) with read-only access, since the masses stay constant throughout the algorithm.

Next, the covalent radius of an atom may also be deduced from its type as stated in the third column of Table 4. The values are derived from [Cordero et al. (2008)]. For simplicity, I choose the value of sp3-carbon for all carbon atoms. The covalent radii are given in ångström and a five-element array of float values is filled with the covalent radius of each element. As the covalent radii are only used in the main program and not in the shaders no buffer array has to be created for them.

The third property to be derived from the types are the colors per element which are presented in the fourth column of Table 4. The colors proposed by [Koltun (1965)] are used for the elements hydrogen, carbon, nitrogen, oxygen and sulfur. A five element array where each entry consists of a four element vector is initialized with the r-, g- and b-components of the color of each element as its first three values and 1 as its fourth value. The access for the initialization of the N-element array is arranged as for the masses. The color information is stored in an array buffer to be accessed in the vertex shader.

Element	Atomic Mass (in u)	Atomic Radius (in Å)	Color
H	1.0078	0.31	
C	12.009	0.76	
N	14.006	0.71	
O	15.999	0.66	
S	32.059	1.05	

Table 4: Atomic Masses, Atomic Radii, Color

The last input inferred from the .xyze file are the bonds which are established between pairs of atoms. Since the peptide used is already in equilibrium state, the bonds are not supposed to change during simulation. Therefore they can be calculated once before the MD algorithm starts and then be used as a constant input for the compute shader. To calculate if there is a bond between a particular pair of atoms, the distance between their coordinates has to be determined. Afterwards the resulting value is compared to the sum of their covalent radii multiplied by a threshold factor. The distance has to be smaller or equal than the product. Here,

a threshold value of $k_{\text{Threshold}} = 1.2$ is chosen. The bond calculation is stated in Equation (21) where r_{ij} is the distance between two atoms i and j, and rad_i and rad_j are their covalent radii. Atoms i and j are bonded, if this equation is true.

$$r_{ij} <= k_{\text{Threshold}} * (\text{rad}_i + \text{rad}_j) \tag{21}$$

To store the bond information, an array of N elements is declared, where each element is filled with another array of 4 elements. Each element is initialized with -1. This form is chosen, because there are N atoms, where each atom can only have up to four bonds. Subsequently, the array is filled anew, now with the bonds that are calculated. Each atom is compared to every other atom but itself. If a bond exists, the index of the bonded atom is inserted into the next element of the current atom. The four element array for one certain atom now contains the indices of the atoms bound to it and, if it has less than four bonds, the remaining elements are filled up with -1. This makes it possible to optimize the algorithm later on by calculating the concrete number of bonds for each atom. The generated array is stored in an SSBO to serve as input for the compute shader.

Additionally, one array for energy and force respectively are initialized with zeros. These are also each stored in an SSBO to be accessed in the compute shader.

3.2 Force Field

After all preparations have been made the MD algorithm starts. At first, the forces have to be calculated. The AMBER energy function utilized for that purpose and its parameters are described in *Chapter 3.2.1*, whereas the simplifications that I apply on this force field to limit the scope of this bachelor thesis are mentioned in *Chapter 3.2.2*. The calculation of the force, i.e. the negative gradient of the energy function supplied by the force field, is presented in *Chapter 3.2.3*. *Chapter 3.2.4* explains the implementation of the force field, including the GPU specific features of the implementation.

3.2.1 Energy Function and Parameters

The force field on which this thesis builds on is an AMBER force field which was published by [Cornell et al. (1995)] (Equation (22)) which again is build upon the force fields introduced in [Weiner et al. (1984)] and [Weiner et al. (1986)]. The values of the force field parameters were derived part empirically part ab initio.

$$\begin{aligned} E_{\text{total}} = & \sum_{\text{bonds}} K_r (r - r_{\text{eq}})^2 + \sum_{\text{angles}} K_\theta (\theta - \theta_{\text{eq}})^2 \\ & + \sum_{\text{dihedrals}} \frac{V_n}{2} [1 + cos(n\phi - \gamma)] + \sum_{i<j} \left[\frac{A_{ij}}{{R_{ij}}^{12}} - \frac{B_{ij}}{{R_{ij}}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}} \right] \end{aligned} \tag{22}$$

Since it is based on molecular mechanics, it has the same structure as shown in

Chapter 2.3.3 (Equations (18) to (20)). Equation (22) defines the total energy of the observed system. It consists of five partial energy functions each of which are described hereafter. The first three partial energy functions are functions for covalent, i.e. bonded, energies and the last two are functions for non-bonded energies. The non-bonded interactions are mainly intermolecular interactions and thus inside of one molecule they are only calculated between atoms separated by at least three bonds. If separated by exactly three bonds they are scaled down by a factor of 0.5. The force field parameters mentioned in the below paragraphs can be derived from [Cornell et al. (1995)].

Equation (23) describes the sum of the energies between each pair of directly bonded atoms. In this formula K_r is the bond force constant, r_{eq} the equilibrium bond length and r the current length of the considered bond. A spring is created by this term, always dragging the atoms into the direction where the particular bond has the equilibrium bond length for this pair of atoms. This means that the atoms oscillate around the positions which yield the equilibrium bond length. r_{eq} is *not* necessarily the equilibrium bond length of the particularly observed bond, since it is a constant value over the entire force field. Figure 3 shows the force generated by the bond energy, where f_a is the force introduced onto atom a by atom b, and $-f_a$ the force introduced onto atom b by atom a. This is according to Newton's third law, see *Chapter 2.1.3*.

$$E_{\text{bonds}} = \sum_{\text{bonds}} K_r (r - r_{\text{eq}})^2 \tag{23}$$

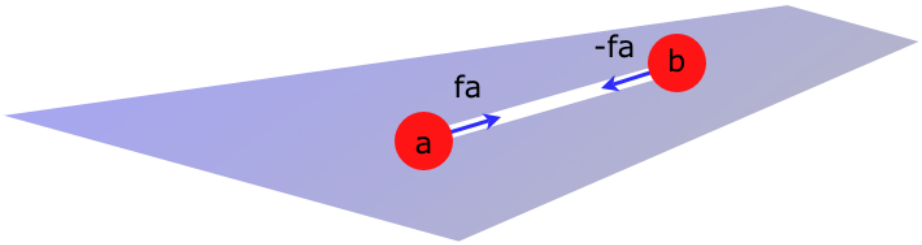


Figure 3: Bond Force [Monasse and Boussinot (2014)]

Equation (24) describes the sum of the energies for each bond angle. In this formula K_θ is the angle force constant, θ_{eq} the equilibrium bond angle and θ the current angle of the considered bond angle. As the previous formula, this one creates a spring, letting the atoms oscillate around the positions which yield the equilibrium angle for the particular bond angle between this combination of atoms. As above, θ_{eq} is constant over the force field and *not* necessarily the equilibrium bond angle of the particularly observed bond. Figure 4 shows the bond angle force with f_a, f_c and $f_b = -(f_a + f_c)$ describing the forces introduced on atoms a, c and b by the bond angle. The bond angle is the angle between the two bonds symbolized by bars between the atoms.

$$E_{\text{angles}} = \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2 \tag{24}$$

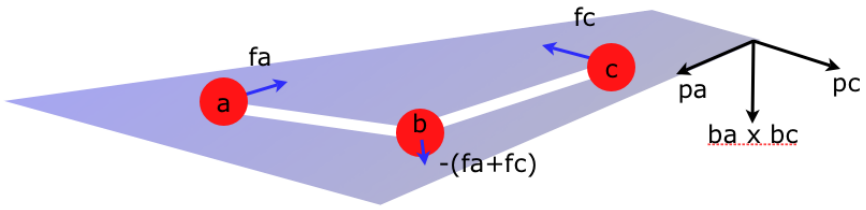


Figure 4: Bond Angle Force [Monasse and Boussinot (2014)]

Equation (25) describes the sum of the energies for each dihedral which may also be labeled as E_{torsions}. It also includes energies generated by out of plane angles. In this formula $V_n/2$ describes the magnitude of the torsion, γ the phase offset, n the periodicity of the torsion and ϕ the current angle of the considered torsion angle. Figure 5 shows the torsion angle forces f_a, f_b, f_c and f_d introduced on atoms a, b, c and d by the torsion angle. The bonds from b to a and b to c span a plane and those from c to b and c to d span another plane. The torsion angle is the angle between these two planes which is stated as θ in the figure, but equals ϕ in Equation (25).

$$E_{\text{dihedrals}} = \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + cos(n\phi - \gamma)] \tag{25}$$

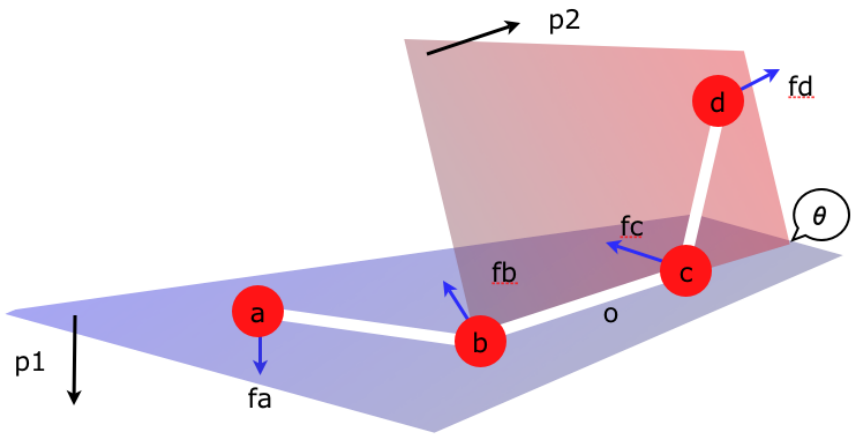


Figure 5: Torsion Angle Force [Monasse and Boussinot (2014)]

Equation (26) describes the sum of the van der Waals energies between each pair of non-bonded atoms. It is a potential called "6-12 potential" with the van der

Waals radius R^* and van der Waals well depth ε as parameters for each of the two considered atoms. R_{ij} is the distance between them. Furthermore, the following substitutions apply: $A_{ij} = \varepsilon_{ij}({R_{ij}}^*)^{12}$, $B_{ij} = 2\varepsilon_{ij}({R_{ij}}^*)^6$, ${R_{ij}}^* = {R_i}^* + {R_j}^*$ and $\varepsilon_{ij} = \sqrt{\varepsilon_i \varepsilon_j}$ for an interaction between atoms i and j.

$$\begin{aligned} E_{\text{van der Waals}'} &= \sum_{i<j} \left[\frac{A_{ij}}{{R_{ij}}^{12}} - \frac{B_{ij}}{{R_{ij}}^6} \right] \\ &= \sum_{i<j} \left[\frac{\varepsilon_{ij} \cdot ({R_{ij}}^*)^{12}}{{R_{ij}}^{12}} - \frac{2 \cdot \varepsilon_{ij} \cdot ({R_{ij}}^*)^6}{{R_{ij}}^6} \right] \end{aligned} \tag{26}$$

Equation (27) describes the sum of the electrostatic energies between each pair of non-bonded atoms. It is a Columbic interaction potential, where the parameters are the electrostatic charges q, and the dielectric constant ε. The charges were derived from an electrostatic model also created by Cornell et al.

$$E_{\text{electrostatics}} = \sum_{i<j} \left[\frac{q_i q_j}{\varepsilon R_{ij}} \right] \tag{27}$$

Each atom type has its own parameters for these functions. The said paper has 13 atom types for the element C, e.g. CA, CB, CT, seven atom types for N, e.g. N and NA, 5 for O, e.g O and OH, 12 for H, e.g. HA and H1 and two for S, that are S and SH. It also has several atom types for elements that are not covered here.

3.2.2 Modifications

In this thesis only the energy terms for bond energies (Equation (23)), angle energies (Equation (24)) and van der Waals energies (Equation (26)) were applied. Additionally, to further reduce complexity I calculated simplified force field parameters only for the elements H, C, N, O and S and not for more specific atom types, i.e. the atom types equal their elements. This was done using the original parameters from Cornell et al. and simplifying them. To calculate one parameter for a certain element, the corresponding original parameters for each atom type belonging to this element were averaged with an arithmetic mean. The atom types OW and HW are left out in this calculation, since they are values for water model atoms which do not exist in this thesis. E.g. for a C — S bond, the parameters of the CT — S bond and the CT — SH bond in the paper from Cornell et al. were averaged. In Table 5 we see the parameters defined by Cornell et al. Thus, in this thesis the parameters for a bond between C and S are $K_r = 232.0$ and $r_{\text{eq}} = 1.810$.

Bond	K_r	r_{eq}
CT-S	227.0	1.810
CT-SH	237.0	1.810

Table 5: Bond force parameters used for bonds between C and S considering atom types by [Cornell et al. (1995)]

The averaged bond force, angle force and van der Waals force parameters can be derived from Tables 6 to 8. The bond force parameters are rounded up to the whole number for K_r and to three decimal places for r_{eq}, the angle force parameters are rounded up to the whole number for K_θ and to two decimal places for θ_{eq} and the van der Waals force parameters are rounded to four decimal places for R^* and ε.

Angle	K_θ	θ_{eq}
H — C — H	35	109.50
H — C — N	41	116.01
H — C — O	50	109.50
H — C — S	50	109.50
H — N — H	35	116.50
H — S — H	35	92.07
C — C — C	64	118.44
C — C — H	41	115.77
C — C — N	71	117.28
C — C — O	69	118.64
C — C — S	50	111.65
C — N — C	67	118.42
C — N — H	32	119.92
C — O — C	60	109.50
C — O — H	45	110.75
C — S — C	62	98.90
C — S — H	43	96.00
C — S — S	68	103.70
N — C — N	70	120.16
N — C — O	74	119.28
O — C — O	80	126.00

Table 6: Angle Force Parameters

Bond	K_r	r_{eq}
H — C	359	1.083
H — N	434	1.010
H — O	553	0.960
H — S	274	1.336
C — C	427	1.434
C — N	434	1.379
C — O	463	1.333
C — S	232	1.810
S — S	166	2.038

Table 7: Bond Force Parameters

Element	R^*	ε
H	1.0796	0.0509
C	2.2054	0.0735
N	1.8495	0.1700
O	1.6818	0.2001
S	2.0000	0.2500

Table 8: Van der Waals Force Parameters

Same as the inputs mentioned in *Chapter 3.1* they are stored in buffers to be accessed by the compute shader and same as for example for the mass buffer array, the access is accomplished by using the atom types. The buffer array for the van der Waals parameters has a length of five, i.e. one entry per element, where each entry consists of a two element vector containing R^* and ε for the particular element. The buffer array for the bond parameters consists of a five element array where each

entry is again a five element array which finally contains the two element vectors for K_r and r_{eq}, i.e. it is a 5x5 array containing two-dimensional vectors. According to the same scheme, the buffer for the angle parameters is a 5x5x5 array containing two-dimensional vectors.

3.2.3 Force Calculation

Since Equation (22) is an energy function, it only calculates the energy of the system. To calculate the forces between each pair of atoms the equation's negative gradient has to be calculated (see *Chapter 2.1.2*). Here, as I only apply Equations (23), (24) and (26), only the force calculation for those three partial energy functions is shown (Equations (28) to (30)). Other than the energy that is calculated for the whole system the force is calculated per atom. For that reason the sigma signs are left out in the following equations. Equations (28) and (30) calculate the bond and van der Waals force between a pair of atoms i and j with positions r_i and r_j. ${F_{\text{bond}}}^i$ and ${F_{\text{bond}}}^j = -{F_{\text{bond}}}^i$ are the forces introduced onto atoms i and j by the bond energy, and ${F_{\text{vdW}}}^i$ and ${F_{\text{vdW}}}^j = -{F_{\text{vdW}}}^i$ the forces introduced onto atoms by the van der Waals energy. Equation (29) calculates the angle force for an angle of atoms i, j and k, with positions r_i, r_j and r_k, where j is the atom in the center of the angle, i.e. the atom which is bound to i and k. For the angle forces ${F_{\text{angle}}}^i$, ${F_{\text{angle}}}^j$ and ${F_{\text{angle}}}^k$ introduced onto atoms i, j and k, the direction for each force is different from the other two forces' directions. Note that for all equations r_{ij} is the distance between atoms i and j. This was done for simplicity, instead of distinguishing between r, the bond length, and R_{ij}, the distance between two not necessarily bound atoms i and j. Further, $\hat{r}_{ji} = \frac{r_i - r_j}{r_{ij}}$ is the normalized direction from j to i.

$$
\begin{aligned}
{F_{\text{bond}}}^i &= -\frac{\delta E_{\text{bond}}}{\delta r_i} \\
&= -\frac{\delta}{\delta r_i} K_r \cdot (r_{ij} - r_{\text{eq}})^2 \\
&= -2 \cdot K_r \cdot (r_{ij} - r_{\text{eq}}) \cdot \frac{\delta}{\delta r_i} (r_{ij} - r_{\text{eq}}) \\
&= -2 \cdot K_r \cdot (r_{ij} - r_{\text{eq}}) \cdot \frac{\delta}{\delta r_i} r_{ij} \\
&= -2 \cdot K_r \cdot (r_{ij} - r_{\text{eq}}) \cdot \frac{\delta}{\delta r_i} \sqrt{(r_{j_x} - r_{i_x})^2 + (r_{j_y} - r_{i_y})^2 + (r_{j_z} - r_{i_z})^2} \\
&= -2 \cdot K_r \cdot (r_{ij} - r_{\text{eq}}) \cdot \hat{r}_{ji}
\end{aligned}
\tag{28}
$$

$$
\begin{aligned}
{F_{\text{angle}}}^{i} &= -\frac{\delta E_{\text{angle}}}{\delta r_i} \\
&= -\frac{\delta}{\delta r_i} K_\theta \cdot (\theta - \theta_{\text{eq}})^2 \\
&= -2 \cdot K_\theta \cdot (\theta - \theta_{\text{eq}}) \cdot \frac{\delta}{\delta r_i} (\theta - \theta_{\text{eq}}) \\
&= -2 \cdot K_\theta \cdot (\theta - \theta_{\text{eq}}) \cdot \frac{\delta}{\delta r_i} \theta \\
&= -2 \cdot K_\theta \cdot (\theta - \theta_{\text{eq}}) \cdot \frac{\delta}{\delta r_i} \arccos(\hat{r}_{ji} \cdot \hat{r}_{jk}) \\
&= -2 \cdot K_\theta \cdot (\theta - \theta_{\text{eq}}) \cdot \frac{\hat{r}_{ji} \times (\hat{r}_{ji} \times \hat{r}_{jk})}{|\hat{r}_{ji} \times (\hat{r}_{ji} \times \hat{r}_{jk})|} \frac{1}{r_{ij}}
\end{aligned}
\tag{29a}
$$

$$
\begin{aligned}
{F_{\text{angle}}}^{k} &= -\frac{\delta E_{\text{angle}}}{\delta r_k} \\
&= -2 \cdot K_\theta \cdot (\theta - \theta_{\text{eq}}) \cdot \frac{(\hat{r}_{ji} \times \hat{r}_{jk}) \times \hat{r}_{jk}}{|(\hat{r}_{ji} \times \hat{r}_{jk}) \times \hat{r}_{jk}|} \frac{1}{r_{jk}}
\end{aligned}
\tag{29b}
$$

$$
{F_{\text{angle}}}^{j} = -({F_{\text{angle}}}^{i} + {F_{\text{angle}}}^{k}) \tag{29c}
$$

$$
\begin{aligned}
{F_{\text{vdW}}}^i &= -\frac{\delta E_{\text{vdW}}}{\delta r_i} \\
&= -\frac{\delta}{\delta r_i}\left[\frac{A_{ij}}{{r_{ij}}^{12}} - \frac{B_{ij}}{{r_{ij}}^{6}}\right] \\
&= -\frac{\delta}{\delta r_i}\left[\frac{\varepsilon_{ij}\cdot({R_{ij}}^*)^{12}}{{r_{ij}}^{12}} - \frac{2\cdot\varepsilon_{ij}\cdot({R_{ij}}^*)^{6}}{{r_{ij}}^{6}}\right] \\
&= -\varepsilon_{ij}\left(({R_{ij}}^*)^{12}\cdot\frac{\delta}{\delta r_i}\left[\frac{1}{{r_{ij}}^{12}}\right] - 2\cdot({R_{ij}}^*)^{6}\cdot\frac{\delta}{\delta r_i}\left[\frac{1}{{r_{ij}}^{6}}\right]\right) \\
&= -\varepsilon_{ij}\left(({R_{ij}}^*)^{12}\cdot\frac{-12}{{r_{ij}}^{13}}\cdot\frac{\delta}{\delta r_i}r_{ij} - 2\cdot({R_{ij}}^*)^{6}\cdot\frac{-6}{{r_{ij}}^{7}}\frac{\delta}{\delta r_i}r_{ij}\right) \\
&= 12\cdot\varepsilon_{ij}\left(\frac{({R_{ij}}^*)^{12}}{{r_{ij}}^{13}} - \frac{({R_{ij}}^*)^{6}}{{r_{ij}}^{7}}\right)\cdot\frac{\delta}{\delta r_i}r_{ij} \\
&= 12\cdot\frac{\varepsilon_{ij}}{{R_{ij}}^*}\left(\left(\frac{{R_{ij}}^*}{r_{ij}}\right)^{13} - \left(\frac{{R_{ij}}^*}{r_{ij}}\right)^{7}\right)\cdot\frac{\delta}{\delta r_i}r_{ij} \\
&= 12\cdot\frac{\varepsilon_{ij}}{{R_{ij}}^*}\left(\left(\frac{{R_{ij}}^*}{r_{ij}}\right)^{13} - \left(\frac{{R_{ij}}^*}{r_{ij}}\right)^{7}\right)\cdot\hat{r}_{ij}
\end{aligned}
\tag{30}
$$

In Equations (28) and (29) the partial terms $-2\cdot K_r\cdot(r_{ij}-r_{\text{eq}})$ and $-2\cdot K_\theta\cdot(\theta-\theta_{\text{eq}})$ are the respective magnitudes of the force, whereas the rest of the considered force term is its direction. For ${F_{\text{bond}}}^i$, the direction is simply the normalized vector from j to i. For the angle force ${F_{\text{angle}}}^i$, $\hat{r}_{ji}\times\hat{r}_{jk}$ is a vector which is orthogonal to the plane spanned by the two bonds from j to i and from j to k. Then $\hat{r}_{ji}\times(\hat{r}_{ji}\times\hat{r}_{jk})$ is a vector which is orthogonal to the bond from j to i and to the previously mentioned orthogonal vector. The resulting vector is normalized and then divided by r_{ij}, the bond length between i and j. This is the direction of the angle force introduced onto atom i. If we look at Figure 4, the resulting direction vectors are the blue vectors originating from atom a, b and c which are equal to i, j and k in this example. For the van der Waals force, $\hat{r}_{ij}$ is the direction of the force whereas the rest of the term is its magnitude.

3.2.4 Implementation

In this chapter the implementation of the force calculation in the compute shader is described looking at one shader invocation. The atom processed by that invocation is henceforth referred to as A_{thread}. Implementing the compute shader, at first all buffers are included as mentioned in *Chapters 3.1 and 3.2.2*. Then all constant values concerning A_{thread}, i.e. mass, type and bonds, plus its position and velocity

are stored locally on the GPU because they are frequently accessed. Subsequently, the force-buffer entry of A_{thread}, i.e. the force introduced onto this atom in the last time step, is stored to be reused for the time step integration (*Chapter 3.3*). To make memory access secure without using a floating point atomic add function, each instance of the compute shader first stores its calculated forces in a local array and all threads' results are summarized after all of them have completed their force calculations. This local force array has as many elements as there are atoms in the considered molecule. Each entry consists of a three dimensional force vector which is initialized with zeros and will be filled with the force introduced onto that atom in the current run of the compute shader. This procedure is described more detailed later in this chapter. To implement the simplified force field as described in *Chapters 3.2.2 and 3.2.3*, the bonded and non-bonded energies and forces are each calculated in a loop which both are described below. In the following I refer to the atom that A_{thread} is currently compared to as $A_{\text{c\_vdW}}$ or $A_{\text{c\_bond}}$ respectively, with c standing for "compare".

For the first loop several remarks have to be made. As mentioned above, for this force field the non-bonded interactions only should be calculated between atoms separated by at least three bonds and non-bonded interactions separated by exactly three bonds should be downscaled. But considering, that the bonds have an equilibrium bond length around roughly 1 Å to 1.5 Å, appropriate to the force field parameters, it was first contemplated to simply calculate the non-bonded interactions only for distances above 4 Å in this paper. However, I finally decided to make both options available in the program by a customizable boolean variable that is set in the simulation parameters of the compute shader, and to show both results in *Chapter 4*.

The loop for the calculation of the non-bonded, i.e. the van der Waals energies and forces iterates over all atoms that have a higher index than A_{thread}. This may be done, since the force introduced onto $A_{\text{c\_vdW}}$ by A_{thread} is the negative value of the force introduced onto A_{thread} by $A_{\text{c\_vdW}}$ (see *Chapter 3.2.3*) and therefore it is sufficient to sample each pair of atoms in only one direction and directly calculate the force introduced onto both atoms by each other. At the beginning of the loop it is inspected if the calculation should be done only for distances over 4 Å or for all pairs of atoms separated by at least three bonds. In the first case, it is simply calculated if the distance between A_{thread} and $A_{\text{c\_vdW}}$ is smaller or equal to 4 Å and if $A_{\text{c\_vdW}}$ is a direct bond of A_{thread}. If any of those two conditions is true, $A_{\text{c\_vdW}}$ is ignored. If not, the van der Waals parameters for both are derived from the buffer and the energy between those two atoms and the forces that A_{thread} and $A_{\text{c\_vdW}}$ exert on each other are calculated using Equations (26) and (30). Otherwise, if the calculation should be done for atoms separated by at least three bonds, it is examined if $A_{\text{c\_vdW}}$ is a direct of bond A_{thread} or of the bonds of A_{thread}, or of the bonds of those, meaning that both atoms are separated by only one, two or three bonds respectively. If one of the first two is the case, the energy and force calculation is discarded. If they are separated by exactly three bonds, the energy and force calculation is performed with a scale factor of 0.5. If the atoms are separated by more than three bonds the energy and

force calculation is performed normally, with a scale factor of 1. Since each thread calculates the force for two atoms, more than one thread might be accessing the same buffer entry if we wrote the forces directly into the buffer at this point. For that reason the forces are first stored in the local force array mentioned above.

The loop for the calculation of the bonded energies and forces, i.e. for the bonds and angles, iterates over all atoms $A_{\text{c\_bond}_1}$ that are bound to A_{thread}. First the bond length, i.e. the distance between the two atoms' centers, as well as the normalized direction from A_{thread} to $A_{\text{c\_bond}_1}$ is calculated. If $A_{\text{c\_bond}_1}$ has a higher index than A_{thread} the bond force parameters are read from the corresponding buffer and subsequently the energy of the bond as well as the force introduced onto A_{thread} and that introduced onto $A_{\text{c\_bond}_1}$ by the bond are calculated. This is done by using Equations (23) and (28), applying the distance and normalized direction as input. Same as in the loop of the previous paragraph, the forces are then stored in the local force array. Afterwards another loop starts which iterates over all atoms $A_{\text{c\_bond}_2}$ that have not yet been processed by the superordinate loop. This means that each combination of two bonds of A_{thread} providing an angle with A_{thread} as the angle's vertex is observed in only one direction. In this loop the bond length of this second bond and the normalized direction from A_{thread} to $A_{\text{c\_bond}_2}$ is calculated and the angle force parameters are derived from the corresponding buffer. Then the bond lengths and normalized directions calculated for $A_{\text{c\_bond}_1}$ and $A_{\text{c\_bond}_2}$ as well as the parameters are used as input for the calculation of the bond angle energy and force which is processed using Equations (24) and (29).

After all energies and forces have been calculated, first the potential, kinetic and total energy of the system are calculated as described in *Chapter 2.1.2*. Afterwards, the calculated forces are written to the force buffer. Therefor the force buffer entries are set to zero and a memory barrier is inserted, to ensure that all threads are at the same position in the program. To assure that the accessed force buffer entry is individual for each thread, I created two loops to be executed consecutively where one of them iterates from the global index of A_{thread}, `gl_WorkGroupID.x`, down to index zero and the second of them iterates from one index above the thread ID, i.e. `gl_WorkGroupID.x + 1`, up to index $N-1$. Using an access index dependent on the index of A_{thread} makes the buffer access unique for each thread. Between those two loops a memory barrier has to be positioned, because the number of iterations differ and it has to be assured that the second loop is not entered until all threads have completed the first loop. Inside of the loops, the force buffer is read at the index provided by the iteration variable, to obtain the current value which is then written to a temporary variable F_{temp}. Afterwards, the value in the local force array at the same index is added to F_{temp}. The resulting value is then written back to the force buffer. Memory barriers are inserted after all read and write operations on the buffer.

3.3 Integration

This chapter explains the leapfrog integration algorithm which I use to perform the time step integration. First, to be able to perform the numerical integration the thread-atom's acceleration of the last time step has to be derived from the stored last time step's force and that of the current time step from the current time step's force by applying Equation (6). Then the leapfrog integration algorithm as presented in [Leimkuhler et al. (1996)] is performed. It provides three formulas to calculate the position and velocity of the current time step, using those of the previous time step. In Equation (31), p_i is the impuls, r_i the position and F_i the force at time step i, whereas m is the mass and Δt the time step. The original Equations (31b) and (31c) are transformed by inserting Equation (31a) and substituting $p_i = m \cdot \dot{r}_i$ (Equation (9)) and $\frac{F_i}{m} = \ddot{r}$ (Equation (6)).

$$p_{i+\frac{1}{2}} = p_i + \frac{1}{2} F_i \cdot \Delta t \tag{31a}$$

$$\begin{aligned}
r_{i+1} &= r_i + \frac{p_{i+\frac{1}{2}}}{m} \cdot \Delta t && \mid \textit{insert Equation (31a)} \\
&= r_i + \frac{p_i + \frac{1}{2} F_i \cdot \Delta t}{m} \cdot \Delta t \\
&= r_i + \frac{p_i}{m} \cdot \Delta t + \frac{1}{2} \frac{F_i}{m} \cdot \Delta t^2 && \mid \textit{insert Equations (6) and (9)} \\
&= r_i + \frac{m \cdot \dot{r}_i}{m} \cdot \Delta t + \frac{1}{2} \ddot{r}_i \cdot \Delta t^2 \\
&= r_i + \dot{r}_i \cdot \Delta t + \frac{1}{2} \ddot{r}_i \cdot \Delta t^2
\end{aligned} \tag{31b}$$

$$\begin{aligned}
p_{i+1} &= p_{i+\frac{1}{2}} + \frac{1}{2} F_{i+1} \cdot \Delta t && \mid \textit{insert Equation (31a)} \\
p_{i+1} &= p_i + \frac{1}{2} F_i \cdot \Delta t + \frac{1}{2} F_{i+1} \cdot \Delta t && \mid \textit{insert Equation (9)} \\
\Leftrightarrow m \cdot \dot{r}_{i+1} &= m \cdot \dot{r}_i + \frac{1}{2} F_i \cdot \Delta t + \frac{1}{2} F_{i+1} \cdot \Delta t && \mid \textit{:m} \\
\Leftrightarrow \dot{r}_{i+1} &= \dot{r} + \frac{1}{2} \frac{F_i}{m} \cdot \Delta t + \frac{1}{2} \frac{F_{i+1}}{m} \cdot \Delta t \\
&= \dot{r} + \frac{1}{2} \ddot{r}_i \cdot \Delta t + \frac{1}{2} \ddot{r}_{i+1} \cdot \Delta t \\
&= \dot{r} + \frac{1}{2} \cdot (\ddot{r}_i + \ddot{r}_{i+1}) \cdot \Delta t
\end{aligned} \tag{31c}$$

The results of Equations (31b) and (31c) are implemented using the named data as input, i.e. the current, not yet changed position of the thread-atom r_i, its current velocity $\dot{r}_i$, the acceleration calculated in this iteration $\ddot{r}_{i+1}$ and that of the last iteration $\ddot{r}_i$, and the time step Δt. Then the new values for the thread-atom's position and velocity are written into the corresponding buffers.

3.4 Drawing

Each atom of the considered molecule is drawn as a sphere at the particular atom's position and with the color of its element (*Chapter 3.1.4*). For this thesis' purpose I extracted the required functions of `CVK::Spheres` and `CVK::Geometry` from the CVK 3 and simplified them by excluding some buffers that are redundant for the aim of this thesis. A radius of 0.7 ångström is chosen for each sphere which is roughly the average radius of the five included chemical elements.

In the render loop, these spheres are drawn using instanced drawing and then illuminated using a simple variant of phong shading. The vertex shader receives the positions and normals of the sphere model as well as the position and color of the currently processed instance as input. Further inputs are the view and projection matrix. To later calculate the fragment color, a light vector is defined in the vertex shader. Then the x-, y- and z-component of the atom's position serve as the translation coordinates of the model matrix and subsequently `gl_Position` is defined by transforming the currently processed position of the sphere with the model-, view- and projection matrix. The position and normal of the current vertex as well as the light position are then calculated in their view space coordinates and those plus the vertex color are provided for the fragment shader. The fragment shader then receives the fragment's position, normal, unlighted color and the light position. The colors of ambient, specular and of the point light as well as the shininess are defined and afterwards the fragments are illuminated by relating the light colors and the unlighted fragment color to the position and normal of the fragment. The resulting fragment color is provided to the fragment buffer.

4 Results

The simulation was executed six times enabling different combinations of forces:

1. bond forces
2. bond and bond angle forces, i.e. all bonded forces implemented in this thesis
3. van der Waals forces, i.e. the non-bonded force implemented in this thesis, calculated for pairs of atoms which are separated by at least three bonds
4. van der Waals forces calculated for pairs of atoms which have a distance of at least 4 Å
5. bond, bond angle and van der Waals forces, i.e. all forces implemented in this thesis, with the van der Waals forces calculated for pairs of atoms which are separated by at least three bonds
6. bond, bond angle and van der Waals forces, with the van der Waals forces calculated for pairs of atoms which have a distance of at least 4 Å

The goal of these simulations is to show how the structure of the chosen molecule, oxytocin, changes throughout these simulations beginning with its start configuration (Figure 6) which is oxytocin in thermodynamic equilibrium, and ending after a particular number of time steps.

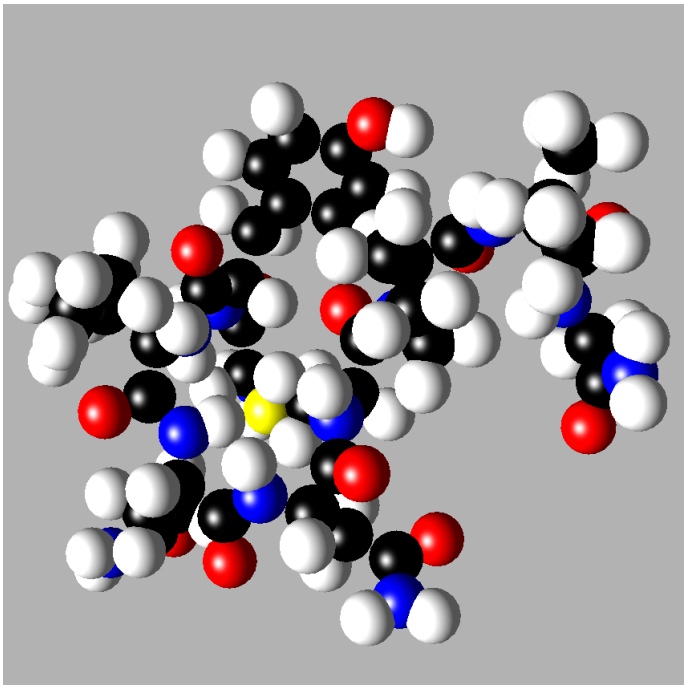


Figure 6: Start configuration

Each of the simulations was run for 105,000 time steps which, considering the chosen time step of 0.01 femtoseconds, equals a simulation time of 1.05 picoseconds. The execution time for those 1.05 ps ranges from about 12 minutes and 10 seconds to 12 minutes and 20 seconds on the used hardware. The start configuration of the molecule which applies for all six simulations can be seen in Figure 6. Subsequently, the observations made for each of these simulations are outlined in the following paragraphs and supplied by excerpts of the simulation. To make it possible to show the trajectories of the molecule the camera had to be zoomed out at some points. However, for the purpose of this thesis it is more important to observe the changes in the molecule's structure rather than its motion in simulation space.

Figure 7 shows the state of the molecule after 0.3, 0.75 and 1.05 picoseconds of simulation time, applying only bond forces. As it can be seen the whole structure of the molecule is compacted throughout the simulation. This first happens within several regions of the molecule, and later also those regions move closer together. The reasons for this might be that the bond angle and torsion angle forces which would usually pull the atoms apart where those angles should be formed are missing. Also missing are the van der Waals forces which would cause atoms to repel each other strongly if they are located too close to one another. Without those forces the bond length between each pair of bound atoms indeed stays around its equilibrium bond length, but non-bonded atoms are also coming closer and closer together since there are too few repulsive forces.

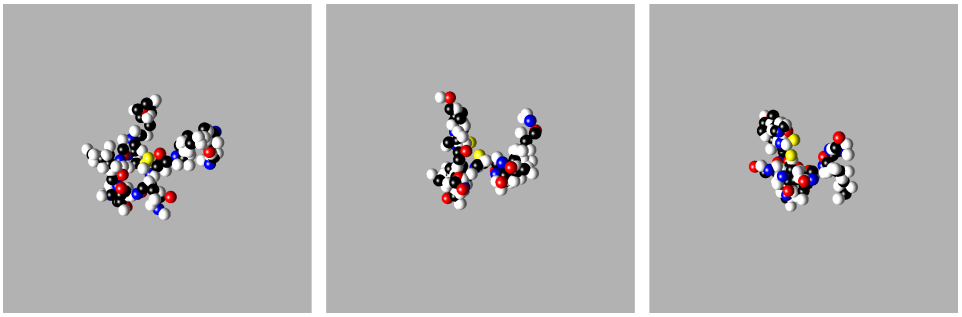


Figure 7: Trajectory only with bond force, after 0.3, 0.75 and 1.05 picoseconds of simulation time (from left to right)

Figure 8 shows the state of the molecule after 0.3, 0.75 and 1.05 picoseconds of simulation time, applying only the bonded forces calculated within the scope of this thesis, i.e. bond forces and bond angle forces. In this simulation, first some regions are straightened out, then the molecule is compacted similar as with only the bond forces and finally the structure of the molecule loosens up again. This could be explained by the fact that now with the bond angle forces another factor for the fluctuation of each atom's position comes into effect. However, since there are still, especially repulsive, forces missing, the molecule also has the tendency to be compacted again within the limits of possible variations.

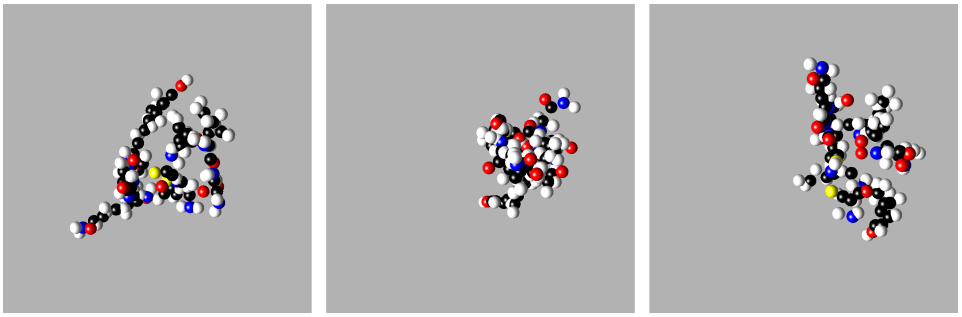


Figure 8: Trajectory only with bond and bond angle force, after 0.3, 0.75 and 1.05 picoseconds of simulation time (from left to right)

Figure 9 shows the state of the molecule after 0.3, 0.75 and 1.05 picoseconds of simulation time, applying only the vdW forces for pairs of atoms which are separated by at least three bonds. It can be seen that the atoms simply move outwards. This happens because there are no bonded energies keeping the molecule together. However the decomposition is a slow process, since the vdW forces are weak compared to other forces.

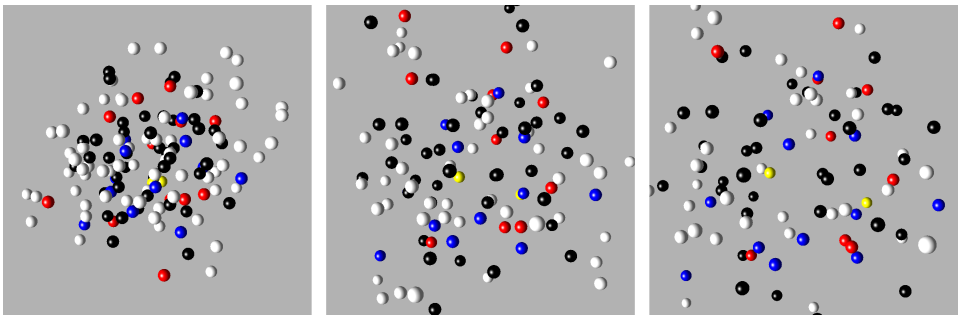


Figure 9: Trajectory only with vdW force calculated for pairs of atoms which are separated by at least three bonds, after 0.3, 0.75 and 1.05 picoseconds of simulation time (from left to right)

Figure 10 shows the state of the molecule after 0.3, 0.75 and 1.05 picoseconds of simulation time, applying only the vdW forces for pairs of atoms which are at least 4 Å apart. It can be seen that especially the atoms at the outer part of the molecule, i.e. the hydrogen atoms, move away from the rest of the molecule. This is due to the fact that they have the highest number of atoms compared to them which have a distance over 4 Å. Combined with the fact that they have the lowest mass, they accelerate the fastest. The core of the molecule in contrast nearly stays at the same position. It is clear to see, that vdW forces are calculated between fewer particles as in the other version which happens because pairs of atoms separated by at least three bonds but with a distance of less than 4 Å do not exert vdW forces onto each other in this version. This automatically leads to fewer forces.

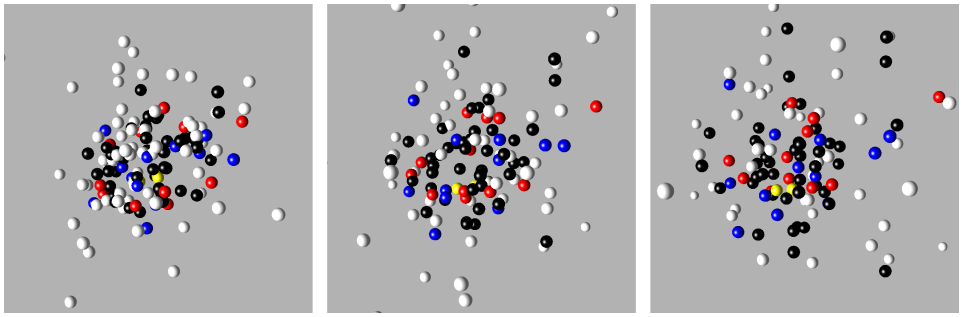


Figure 10: Trajectory only with vdW force calculated for pairs of atoms which are at least 4 Å apart, after 0.3, 0.75 and 1.05 picoseconds of simulation time (from left to right)

Figure 11 shows the state of the molecule after 0.3, 0.75 and 1.05 picoseconds of simulation time, applying all implemented forces with the vdW forces calculated for pairs of atoms which are separated by at least three bonds. After 0.3 ps it can be seen, that the molecule begins to break down in several groups. These groups are moving further and further apart but still can be recognized as groups of atoms. A possible reason for this behavior is on one hand, that two of the partial energy functions were not implemented, particularly the torsion angle forces. Without this third formula for bonded energies which should hold the molecule together, the molecule is driven apart by the van der Waals forces. On the other hand without a water model surrounding the molecule and without a boundary volume the atoms' motions are not guided and restricted by the molecule's environment but rather deflect in any direction.

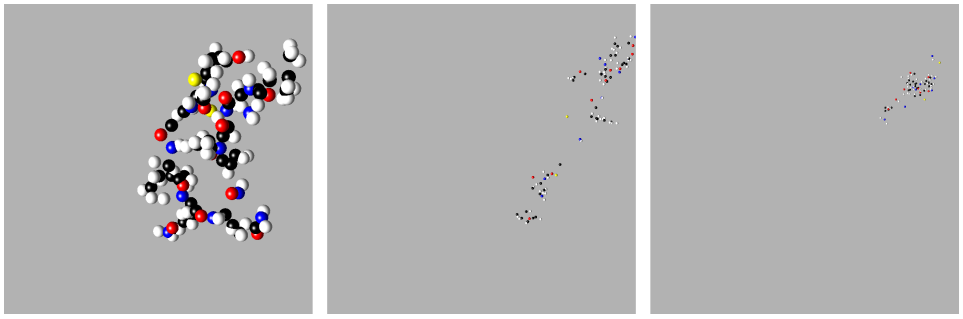


Figure 11: Trajectory with all implemented forces and with the vdW forces calculated for pairs of atoms which are separated by at least three bonds, after 0.3, 0.75 and 1.05 picoseconds of simulation time (from left to right)

Figure 12 shows the state of the molecule after 0.3, 0.75 and 1.05 picoseconds of simulation time, applying all implemented forces with the vdW forces calculated for pairs of atoms which are at least 4 Å apart. Similar to the first execution of the simulation, where only bond forces are considered, the molecule is first compacted within several regions. In contrast to prior version where all implemented forces

are applied but the vdW forces are calculated differently, these groups are not detached from each other but the molecule strongly deforms. However, it has the same restrictions as mentioned before.

Figure 12: Trajectory with all implemented forces and with the vdW forces calculated for pairs of atoms which are at least 4 Å apart, after 0.3, 0.75 and 1.05 picoseconds of simulation time (from left to right)

In all of these versions the molecule's structure cannot be maintained. As mentioned above, this is due to several reasons. First, other than the original force field parameters which are determined per atom type, the simplified force field parameters used for this simulation are inaccurate, since they are only chosen per element. Another reason are the missing energy functions for the torsion and electrostatic energies and forces, and the simulation lacks a water model as well as a boundary volume. Despite all of these limitations it was possible to show the impact that distinct forces have onto the motion behavior of a molecule and to therefore illustrate the functionality of the utilized force field.

5 Conclusion

In this thesis the basics of molecular dynamics simulations were discussed and its application was demonstrated by implementing a small MD program. The resulting simulations was used to show how individual terms of a force field affect the behavior of a molecule with respect to time. For future work it would be interesting to extend the approach of this thesis, adding the missing energy terms, i.e. the torsion force term and that for the electrostatic interactions, applying a water model and implementing a boundary volume, to generate more realistic demonstrations of the various energy terms. Additionally an algorithm optimized for more atoms and consequently more threads could be implemented. Furthermore, it is most certainly desirable to become acquainted with existing molecular dynamics programs to acquire a deeper knowledge of molecular dynamics simulations. This last idea and a collaboration with physicists, chemists or biologists would be advisable, since MD is a highly interdisciplinary field. Though the implementation of MD programs seems to be performed rather by natural scientists than by computer scientists, as literature research suggests, it is for sure an interesting and promising field of research worth of support from both scientific sectors.

References

Allinger, N. L., Zhou, X., and Bergsma, J. (1994). Molecular mechanics parameters. *Journal of Molecular Structure: THEOCHEM*, 312(1):69–83.

Andersen, H. C. (1980). Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of chemical physics*, 72(4):2384–2393.

Branden, C. I. and Tooze, J. (2012). *Introduction to protein structure*. Garland Science.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S. a., and Karplus, M. (1983). CHARMM: a program for macromolecular energy, minimization, and dynamics calculations. *Journal of computational chemistry*, 4(2):187–217.

Cordero, B., Gómez, V., Platero-Prats, A. E., Revés, M., Echeverr\'\ia, J., Cremades, E., Barragán, F., and Alvarez, S. (2008). Covalent radii revisited. *Dalton Transactions*, (21):2832–2838.

Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M., Ferguson, D. M., Spellmeyer, D. C., Fox, T., Caldwell, J. W., and Kollman, P. A. (1995). A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *Journal of the American Chemical Society*, 117(19):5179–5197.

Hernandez, H. (2017). Standard Maxwell-Boltzmann distribution: Definition and properties. *ForsChem Research Reports*, 2.

Hill, T. L. (1986). *An introduction to statistical thermodynamics*. Courier Corporation.

Holger87 (2012). Protein-Struktur. `https://commons.wikimedia.org/wiki/File:Protein-Struktur.png`. [Holger87 [CC BY-SA (`https://creativecommons.org/licenses/by-sa/3.0`)]] (Online; accessed 13-February-2020)].

Kmiecik, S., Gront, D., Kolinski, M., Wieteska, L., Dawid, A. E., and Kolinski, A. (2016). Coarse-grained protein models and their applications. *Chemical reviews*, 116(14):7898–7936.

Koltun, W. L. (1965). Space filling atomic units and connectors for molecular models.

Leimkuhler, B. J., Reich, S., and Skeel, R. D. (1996). Integration methods for molecular dynamics. In *Mathematical Approaches to biomolecular structure and dynamics*, pages 161–185. Springer.

Linderberg, J. and Öhrn, Y. (2004). *Propagators in quantum chemistry*. John Wiley & Sons.

Loeb, L. B. (2004). *The kinetic theory of gases*. Courier Corporation.

Micha, D. A. and Burghardt, I. (2007). *Quantum dynamics of complex molecular systems*, volume 83. Springer.

Monasse, B. and Boussinot, F. (2014). Determination of forces from a potential in molecular dynamics. *arXiv preprint arXiv:1401.1181*.

Nzjacobmartin (2017). Statistical Ensembles. `https://commons.wikimedia.org/wiki/File:Statistical_Ensembles.png`. [Nzjacobmartin [CC BY-SA (`https://creativecommons.org/licenses/by-sa/4.0`)] (Online; accessed 13-February-2020)].

Rappe, A. K. and Casewit, C. J. (1997). *Molecular mechanics across chemistry*. University Science Books.

Rovigatti, L., Sulc, P., Reguly, I. Z., and Romano, F. (2015). A comparison between parallelization approaches in molecular dynamics simulations on GPUs. *Journal of Computational Chemistry*, 36(1):1–8.

Royal Society of Chemistry (RSC) (n.d.). Oxytocin. CSID:388434, `http://www.chemspider.com/Chemical-Structure.388434.html` (accessed 10:45, Jan 29, 2020) [Royal Society of Chemistry (RSC)].

Senn, H. M. and Thiel, W. (2009). QM/MM methods for biomolecular systems. *Angewandte Chemie International Edition*, 48(7):1198–1229.

Tuckerman, M. E. and Martyna, G. J. (2000). Understanding modern molecular dynamics: techniques and applications.

Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., and Berendsen, H. J. C. (2005). GROMACS: fast, flexible, and free. *Journal of computational chemistry*, 26(16):1701–1718.

Weiner, S. J., Kollman, P. A., Case, D. A., Singh, U. C., Ghio, C., Alagona, G., Profeta, S., and Weiner, P. (1984). A new force field for molecular mechanical simulation of nucleic acids and proteins. *Journal of the American Chemical Society*, 106(3):765–784.

Weiner, S. J., Kollman, P. A., Nguyen, D. T., and Case, D. A. (1986). An all atom force field for simulations of proteins and nucleic acids. *Journal of computational chemistry*, 7(2):230–252.

Wieser, M. E., Holden, N., Coplen, T. B., Böhlke, J. K., Berglund, M., Brand, W. A., De Bièvre, P., Gröning, M., Loss, R. D., Meija, J., and Others (2013). Atomic weights of the elements 2011 (IUPAC Technical Report). *Pure and Applied Chemistry*, 85(5):1047–1078.

Zwier, M. C. and Chong, L. T. (2010). Reaching biological timescales with all-atom molecular dynamics simulations.