universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Implementation and Testing of CHARMM as Backend to the Free Energy package Transformato

verfasst von / submitted by

Benedict Johannes Braunsfeld, B.Sc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 862
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Chemie
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Stefan Boresch

"You shall not pass!"

Gandalf the Grey

UNIVERSITY OF VIENNA

Abstract

Faculty of Chemistry
Department of Computational Biological Chemistry

Master of Science

Implementation and Testing of CHARMM as Backend to the Free Energy package Transformato

by Benedict BRAUNSFELD

In recent years free energy calculations have become a popular method in academia and the pharmaceutical industry, due to the possibility to calculate the thermodynamic properties of a system. In free energy calculations, it is possible to calculate free energy differences along non-physical pathways by transforming small parts of the system, i.e., changing a functional group into another. This transformation is done over a series of intermediate states. A new approach inserts a common core state between the initial and the final state and calculates the free energy difference between the initial state and the common core, as well as between the final state and the common core to obtain the free energy difference between initial and final state. The `TRANSFORMATO` package is an implementation of this approach, automatically setting up the various intermediate states needed. In this thesis, a `CHARMM` backend was successfully implemented into the `TRANSFORMATO` package. This implementation was used to show that `TRANSFORMATO` successfully implemented the common core approach for both `CHARMM` and `openMM` backend by validating calculations done with `TRANSFORMATO` against reference calculations done with the established `PERT` module of `CHARMM`.

Acknowledgements

I would like to thank the Austrian Science Fund (FWF) for funding project P31024 and supporting this work. Also, I would like to thank the people making this work possible. First, I would like to thank Univ.-Prof. Mag. Dr. Stefan Boresch. As my supervisor, he made my participation in this project possible and provided me with all resources needed for my work. Also, Stefan, even though I know he is a very busy man, made meetings on short notice possible to answer my endless questions and helped me to improve my knowledge in molecular dynamics and especially in free energy calculations. Secondly, I want to thank Dr. Marcus Wieder for all his guidance and for a feeling of discussing on an equal level even though my knowledge in Python programming and free energy calculations is still lacking. I am very thankful for all our scientific discussions every day and night time. Working with both Stefan and Marcus is very enjoyable and motivating. Further, I want to thank the University of Vienna and especially the Department of Computational Biological Chemistry to make this work possible. Lastly, I want to thank my family and friends for their support.

Contents

List of Figures

List of Tables

List of Abbreviations

ABNR	**A**dopted **B**asis **N**ewton **R**aphson method
BAR	**B**ennett **A**cceptance **R**atio
CC	**C**ommon **C**ore
CMM	**C**har**MM**
CPI	**C**yclo**P**entyl**I**ndole
FEP	**F**ree **E**nergy **P**erturbation
FES	**F**ree **E**nergy **S**imulation
GPU	**G**raphics **P**rocessing **U**nit
LJ	**L**ennard **J**ones
MBAR	**M**ultistate **B**ennett **A**cceptance **R**atio
MC	**M**onte **C**arlo
MCS	**M**aximum **C**ommon **S**ubstructure
MD	**M**olecular **D**ynamics
OMM	**O**pen**MM**
PBC	**P**eriodic **B**oundary **C**ondition
PME	**P**article **M**esh **E**wald summation
QM	**Q**uantum **M**echanic
SAI	**S**erial **A**tom **I**nsertion
SD	**S**teepest **D**escent
TDC	**T**hermo**D**ynamic **C**ycle
TF	**T**rans**F**ormati
TI	**T**hermodynamic **I**ntegration
WHAM	**W**eighted **H**istogram **A**nalysis **M**ethod

List of Symbols

a or $\ddot{\vec{r}}$	acceleration	ms^{-2}
θ	angle	$\circ$
l	bond length	Å
$\sigma_i j$	collision diameter	Å
$\langle A \rangle$	ensemble average	
$\mathscr{F}$	force	$kgms^{-2}$
ω	frequency	s^{-1}
$\mathscr{H}$	Hamiltonian	kcal/mol
F	Helmholtz free energy	kcal/mol
ΔF	Helmholtz free energy difference	kcal/mol
$\Delta\Delta F$	relative Helmholtz free energy difference	kcal/mol
m	mass	kg
D_e	potential depths	kcal/mol
$\mathscr{U}$ or $\mathscr{V}$ or v	potential energy	kcal/mol
ρ	probability density	
t	time	s
ϵ_0	vacuum permittivity	kgs^{-2}
$\vec{v}$ or $\dot{\vec{r}}$	velocity	ms^{-1}
$\epsilon_i j$	well depth	kcal/mol

Chapter 1

Introduction

Molecular simulations are utilized to study natural systems by mathematically modeling them. The interest in this field of research has been growing rapidly over the last decades. Due to the increasing power and decreasing costs of computational resources, molecular simulations are broadly used for computational studies (Figure 1.1).

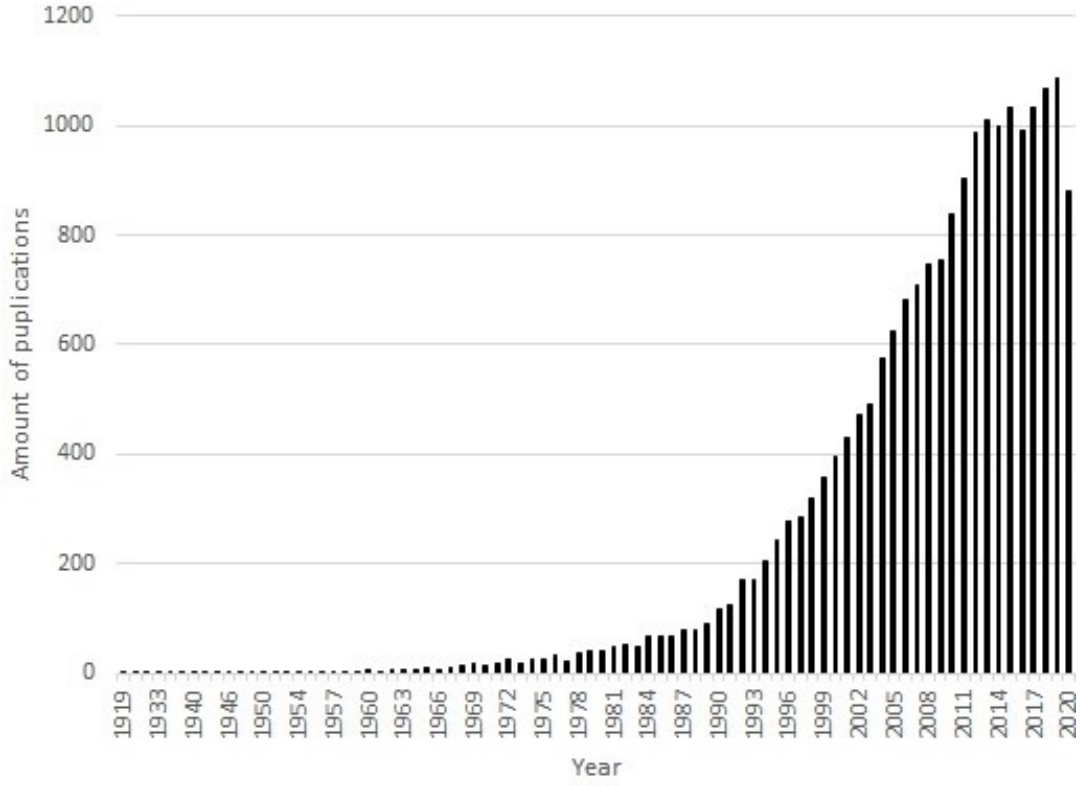

FIGURE 1.1: Number of publications including molecular simulations per year.[1]

Molecular simulations are used to support and visualize experimental findings. Even though quantum mechanics-based methods provide the most accurate results, they are expensive and limited to small systems (at best a few hundred atoms) and short time scales (ps). Methods based on classical mechanics (molecular mechanics) apply to larger systems (millions of atoms) and much longer time scales (up to µs). Therefore, molecular mechanics are the tool of choice to study biological phenomena and properties.[2]

One family of computational methods are the so-called free energy simulations (FES). This methodology can be used to calculate "absolute" free energies (i.e. solvation free energy differences or binding free energy differences). However, in many cases, it is more efficient to compute relative free energy differences.[3–5] To calculate relative free energy differences, the principle of thermodynamic circles is utilized, allowing the calculation of a non-physical pathway by transforming a molecule into another. These transformations are done step-wise in a series of intermediate states.

Free energy simulations are used in theoretical drug design and discovery since the free energy indicates whether a process is thermodynamically favorable. As an example, the binding free energy of a small molecule, e.g. a drug, to a receptor indicates the binding affinity between drug and receptor. Consequently, free energy calculations are used to predict the most suitable derivatives of lead structures[1].[3–5] This prediction aids in the decision of which molecules to prioritize for synthesis. To be useful on an industrial level, these calculations have to be stable and fast enough to compete with an experimental synthesis.

Free energy methods are supported by various molecular simulation packages, such as AMBER[6], CHARMM[7], GROMACS[8], and SOMD[9]. Results from different simulation packages should be comparable for a given molecular configuration and force field within statistical significance. A recent study compared results obtained with four MD programs; while the overall results agreed, differences between the four programs certainly were larger than expected based on statistical error estimates.[10] Therefore, it is difficult to directly compare results obtained with different MD programs. Another difficulty is the calculation procedure in itself. In many simulation packages, free energy calculations need extensive manual adjustments at all stages of the procedure. Especially the choice of intermediate states is still one of the most complicated aspects of free energy calculations.[11] Setting them up manually remains challenging even for experts. Therefore, supporting software packages are needed to standardize and automate free energy calculations.[12, 13]

One of these tools is TRANSFORMATO.[14] The TRANSFORMATO framework is the first implementation of the Common Core approach.[15] In this approach, a common core state is inserted between the initial and the final state. The common core makes it possible to calculate the relative free energy between multiple systems with a common substructure by calculating the free energy difference between the initial states of these systems and the common core. In established approaches a complete transformation for each pair of systems is required. TRANSFORMATO automatically sets up the various intermediate states needed. At the current state of development TRANSFORMATO provides an openMM[16] backend, which has not been fully validated yet.

In this work, a CHARMM[7] backend is added to the TRANSFORMATO framework. The CHARMM backend was validated by comparing to calculations of the common core approach set up manually. These calculations set up manually were independently validated with the help of the existing PERT module of CHARMM. Since the PERT module

[1] A lead structure is a molecule with therapeutically useful biological activity. Often the lead structure is not the optimal structure for the target of interest and needs to be further modified.

provides only the possibility of VSWItch[17] as switching function for the Lennard-Jones interaction, the calculations set up manually and with TRANSFORMATO also utilized a VSWItch as switching function. To validate the openMM backend, which uses a VFSWitch[7] as switching function, it was compared to calculations done with TRANSFORMATO with the CHARMM backend using a VFSWitch as switching function.

Chapter 2

Theory

2.1 Molecular Dynamics Simulations

Classical molecular dynamics simulations are based on Newton's laws of motion. By solving Newton's second law (Equation 2.1) the force of a particle i can be obtained.[2]

$$\overrightarrow{\mathscr{F}_i} = m_i \ddot{\vec{r}}_i \tag{2.1}$$

$$\ddot{\vec{r}}_i = \frac{d^2\vec{r}_i}{dt^2} \tag{2.2}$$

Here m_i is the mass of the particle i, $\ddot{\vec{r}}_i$ is the second derivative of the position $\vec{r}_i$ of i with respect to the time t (Equation 2.2) and $\overrightarrow{\mathscr{F}_i}$ denotes the force on i. In a system of N interacting particles described through the potential $\mathscr{U}(\vec{r}_i)$ with $i = 1, ..., N$ the forces are expressed as shown in Equation 2.3.[2]

$$\overrightarrow{\mathscr{F}_i} = -\frac{\partial \mathscr{U}(\vec{r}_i)}{\partial \vec{r}_i} \tag{2.3}$$

When combining Equation 2.2 and Equation 2.3 and applying it to a system of N atoms, a set of 3N second order differential equations must be solved at every time step. In a realistic system, the motion of one particle is coupled to all other particles in the system. This many-body problem can only be solved numerically. The integrator has to be consistent. Therefore, within range of the step size $\Delta t \longrightarrow 0$ the original differential equations have to be reproduced. The integrator has to be stable and accurate, giving a numerical result close to the solution, regardless of the simulation length. Since the equation of motion is time-reversible, the algorithm has to be symplectic. This means that the phase space volume and total energy of the system are conserved between two consecutive time steps. At last, the algorithm should be cost-efficient, producing results in a reasonable time with adequate time steps and simulation lengths. The most popular algorithms are the Verlet type algorithms.([2, 18]; [19], pp. 353-409)

2.1.1 The Verlet Algorithm

The Verlet algorithm, described by Loup Verlet[20], uses the position $\vec{r}_i$ and force $\overrightarrow{\mathscr{F}_i}$ of a particle i at time t and the position of the same particle at time $(t - \Delta t)$ to calculate the position of the particle at time $(t + \Delta t)$. To obtain the Verlet algorithm (Equation 2.6) the Taylor expansions of $\vec{r}_i(t + \Delta t)$ (Equation 2.4) and $\vec{r}_i(t - \Delta t)$

(Equation 2.5) are combined.[2] The dependency of time and position $\vec{r}_i$ is illustrated in Figure 2.1.

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \frac{1}{2m_i}\vec{\mathcal{F}}_i(t)\Delta t^2 + ... \tag{2.4}$$

$$\vec{r}_i(t-\Delta t) = \vec{r}_i(t) - \vec{v}_i(t)\Delta t + \frac{1}{2m_i}\vec{\mathcal{F}}_i(t)\Delta t^2 + ... \tag{2.5}$$

$$\vec{r}_i(t+\Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t-\Delta t) + \frac{1}{m_i}\vec{\mathcal{F}}_i(t)\Delta t^2 + ... \tag{2.6}$$

Since the Verlet algorithm does not explicitly include the velocity $\vec{v}_i$, it can be obtained by taking the mean value of positions $\vec{r}_i(t+\Delta t)$ and $\vec{r}_i(t-\Delta t)$ (Equation 2.7). Thus, the velocity is always one step behind the position term.[2]

$$\vec{v}_i(t) = \frac{(\vec{r}_i(t+\Delta t) - \vec{r}_i(t-\Delta t))}{2\Delta t} \tag{2.7}$$

Regarding Equation 2.6 the position at time $(t-\Delta t)$ is required to obtain the position at time $(t+\Delta t)$ for $t = 0$. Hence, the Verlet algorithm is not self-starting. Thus, the position at time $(-\Delta t)$ is not available and needs to be calculated with a Taylor expansion over $\vec{r}_i(t)$ or another algorithm.([19], pp. 353-409)

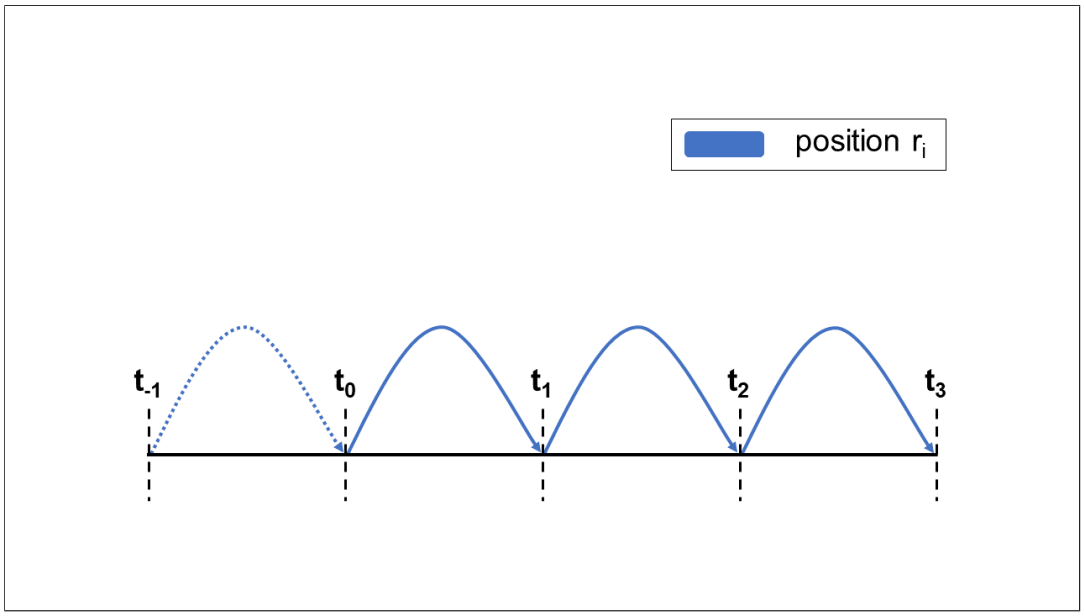

FIGURE 2.1: Schematic illustration of the dependency of position $\vec{r}_i$ and time t in the Verlet algorithm.

2.1.2 The Velocity Verlet Algorithm

There are several variations of the Verlet algorithm, one of them being the Velocity Verlet algorithm.[21] Starting from Equation 2.4 the position at time $t+\Delta t$ is calculated by using the velocity $\vec{v}_i$ and force $\frac{1}{m_i}\vec{\mathcal{F}}_i$ at time t. Next the velocity at time $t+\frac{1}{2}\Delta t$ is calculated according to Equation 2.8. Calculating the force $\vec{\mathcal{F}}_i$ with Equation 2.1 at the current position of $t+\Delta t$ results in the acceleration through force

$\frac{1}{m_i}\vec{\mathscr{F}}_i(t+\Delta t)$. In the final step the new velocity $\vec{v}_i(t+\Delta t)$ is determined as shown in Equation 2.9.([19], pp. 353-409; [21])

$$\vec{v}_i(t+\frac{1}{2}\Delta t) = \vec{v}_i(t) + \frac{1}{2m_i}\vec{\mathscr{F}}_i(t)\Delta t \tag{2.8}$$

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t+\frac{1}{2}\Delta t) + \frac{1}{2m_i}\vec{\mathscr{F}}_i(t+\Delta t)\Delta t \tag{2.9}$$

By inserting Equation 2.8 into Equation 2.9 the final expression of the Velocity Verlet algorithm is obtained (Equation 2.10).([19], pp. 353-409; [21])

$$\vec{v}_i(t+\Delta t) = \vec{v}_i(t) + \frac{1}{2m_i}\vec{\mathscr{F}}_i(t)\Delta t + \frac{1}{2m_i}\vec{\mathscr{F}}_i(t+\Delta t)\Delta t \tag{2.10}$$

The Velocity Verlet algorithm is using the velocity and the forces at time t to start; hence, it is self-starting. A visual representation of the Velocity Verlet algorithm is shown in Figure 2.2. After the calculation of each step in Equations 2.4, 2.8 and 2.9 the position and velocity are simultaneously at the same time step (Figure 2.2).

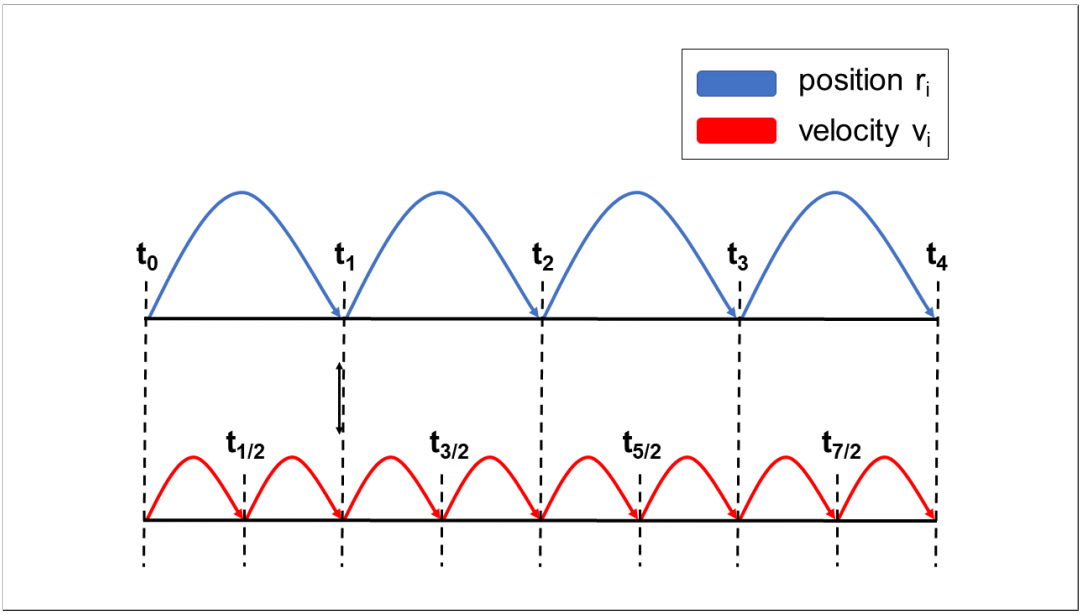

FIGURE 2.2: Schematic illustration of the dependencies of position $\vec{r}_i$, velocity $\vec{v}_i$ and time t in the Velocity Verlet algorithm.

2.1.3 The Leapfrog Algorithm

Another modification of the Verlet algorithm is the Leapfrog algorithm.[22] To obtain the velocity at time $t+\frac{1}{2}\Delta t$, the position and force at time t and the velocity at time $t-\frac{1}{2}\Delta t$ are required (Equation 2.12). The velocity at time $t+\frac{1}{2}\Delta t$ is then used to update the position to time $t+\Delta t$. Due to the alternating half step difference between position and velocity, the algorithm is named after the well known children's game (Figure 2.3). The velocity at time t is calculated with Equation 2.13.[2]

$$\vec{r}_i(t+\Delta t) = \vec{r}_i(t) + \vec{v}_i(t+\frac{1}{2}\Delta t)\Delta t \tag{2.11}$$

$$\vec{v}_i(t + \frac{1}{2}\Delta t) = \vec{v}_i(t - \frac{1}{2}\Delta t) + \frac{1}{m_i}\vec{\mathscr{F}}_i(t)\Delta t \tag{2.12}$$

$$\vec{v}_i(t) = \frac{\vec{v}_i(t + \frac{1}{2}\Delta t) + \vec{v}_i(t - \frac{1}{2}\Delta t)}{2} \tag{2.13}$$

Compared to the Verlet algorithm, the Leapfrog algorithm explicitly includes the velocity. The general idea of the Leapfrog algorithm is sketched in Figure 2.3. A major drawback is that the position and velocity are shifted by half a time step. Therefore, it is not easily possible to calculate the kinetic energy contribution to the total energy for the position at the current time step. In this regard the Velocity Verlet algorithm is preferable.([2, 22]; [19], pp. 353-409)

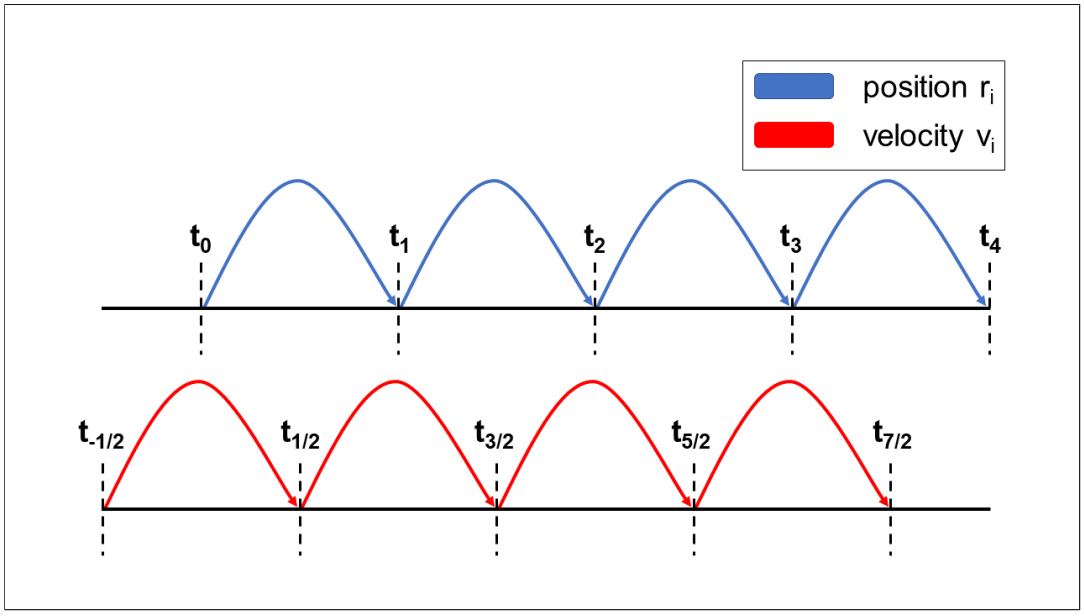

FIGURE 2.3: Schematic illustration of the dependencies of position $\vec{r}_i$, velocity $\vec{v}_i$ and time t in the Leapfrog algorithm.

2.2 Force Fields

In classical Molecular Dynamics simulations, electrons are not explicitly treated while solving the equation of motion. The prerequisite is that nuclei and electrons are decoupled. This decoupling of the electronic motion and the nuclear motion is justified by the Born-Oppenheimer approximation[1]. In MD simulations forces between atoms are calculated, using a force field which is an empirically derived collection of functions and associated parameters describing the interaction of different atoms in a defined environment. The contribution of these interactions to

[1]The Born-Oppenheimer approximation considers the great difference in masses between a nucleus and an electron. Therefore, electronic and nuclear motion operate on a different time scale. This means an electron adjusts instantly to the motion of a nuclei, while the motion of the nuclei from an electronic perspective is fixed.([2]; [19], pp. 35-36)

the potential energy of a system is commonly divided into bonded interactions (intramolecular; bond stretching, angle bending, bond rotation) and non-bonded interactions (intra- and intermolecular; electrostatic and van der Waals interactions) (Equation 2.14).([2]; [19], pp. 165-199)

$$\mathscr{V}(r^N) = \underbrace{\sum_{bonds} \frac{k_i}{2}(l_i - l_{i,0})^2 + \sum_{angles} \frac{k_i}{2}(\theta_i - \theta_{i,0})^2 + \sum_{torsions} \frac{V_n}{2}(1 + \cos(n\omega - \gamma))}_{= bonded\ interactions} + \underbrace{\sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \right)}_{= non-bonded\ interactions} \tag{2.14}$$

2.2.1 Bonded Interactions

The potential energy of bond stretching can be described by a Morse potential:

$$v(l) = D_e\{1 - \exp[-a(l - l_0)]\} \tag{2.15}$$

with D_e as the depth of the potential energy minimum and $a = \omega\sqrt{\frac{\mu}{2D_e}}$, where ω is the frequency of the bond vibration and μ the reduced mass. The Morse potential describes the behavior of a bond from equilibrium distance to dissociation. In molecular dynamics simulations, the bond lengths are mostly close to their equilibrium values. Hence, the Morse potential is replaced by a harmonic potential following Hooke's law (Equation 2.16), since the region close to the equilibrium is sufficiently modeled by the harmonic potential as illustrated in Figure 2.4.

$$v(l) = \frac{k}{2}(l - l_0)^2 \tag{2.16}$$

In Equation 2.16 l is the bond length, l_0 is the bond length at the equilibrium and k is the stretching constant which is related to the frequency of the bond vibration ω by $k = \frac{\omega^2}{\mu}$.([19], pp. 170-173)

Similarly to the bond stretching, the angle bending is also described by a harmonic potential:

$$v(\theta) = \frac{k}{2}(\theta - \theta_0)^2 \tag{2.17}$$

which depends on an angle θ in respect to the angle θ_0 at the equilibrium.([19], p. 173) Bond stretching and angle stretching are illustrated in Figure 2.5 (I & II).

The third part of the bonded interaction are the bond rotations addressing the steric interactions between two sets of three atoms, having two atoms in common. The angle between the two intersecting planes that are created by the two sets is called the dihedral angle. Figure 2.5 (III) shows the sets $\{A, B, C\}$ and $\{B, C, D\}$ on the corresponding planes α and β and the dihedral angle ω. The periodic behavior of the bond rotation is modeled by a torsional potential as a Fourier series:

$$v(\omega) = \frac{V_n}{2}(1 + \cos(n\omega - \gamma)) \tag{2.18}$$

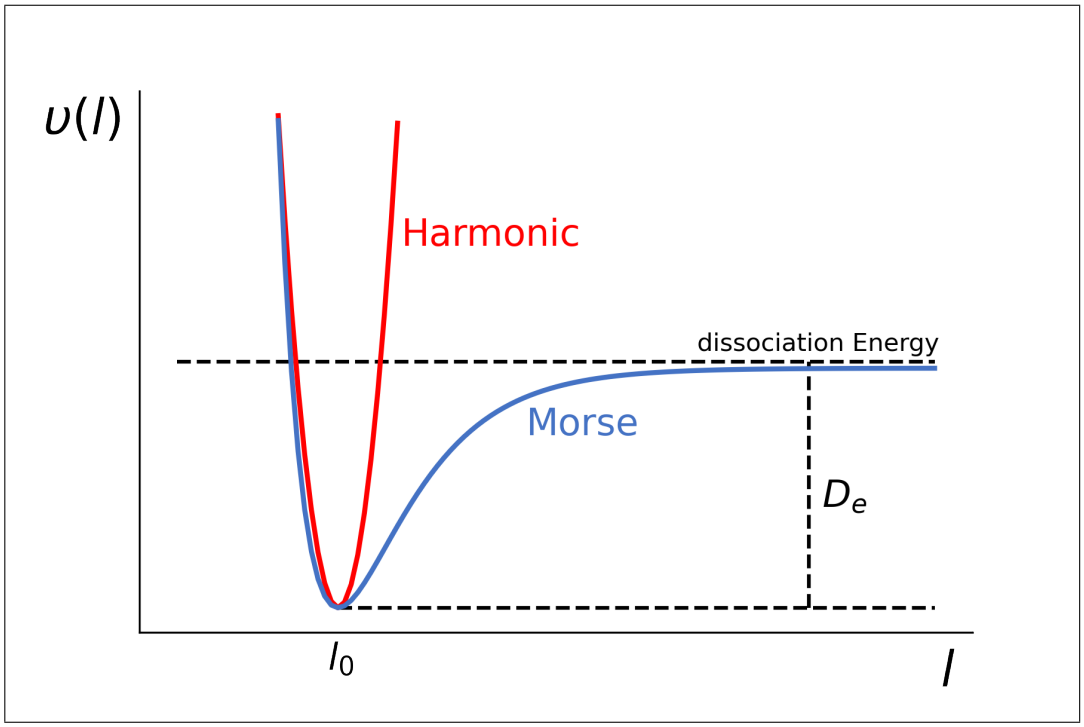

FIGURE 2.4: Sketched comparison between a Morse potential (blue) and a harmonic potential (red).

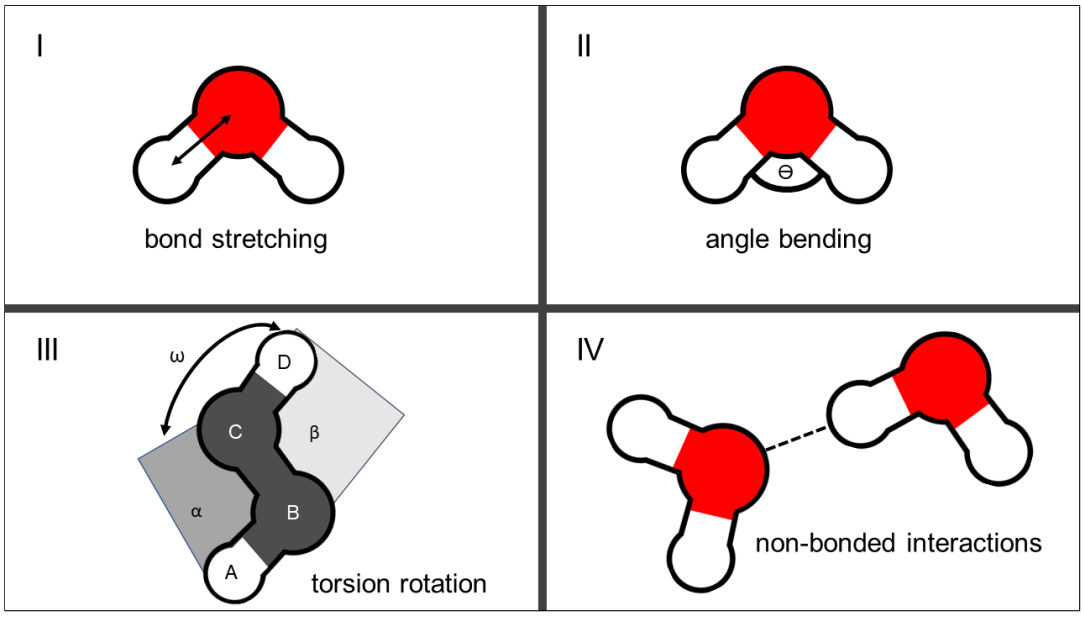

FIGURE 2.5: I) Illustration of the bond stretching along l. II) Illustration of the angle bending at angle θ. III) Illustration of a dihedral torsion between atoms A, B, C, D. The dihedral angle ω is defined between the two planes α generated by atoms A, B, C and plane β generated by atoms B, C, D. IV) Schematic representation of intermolecular interactions.

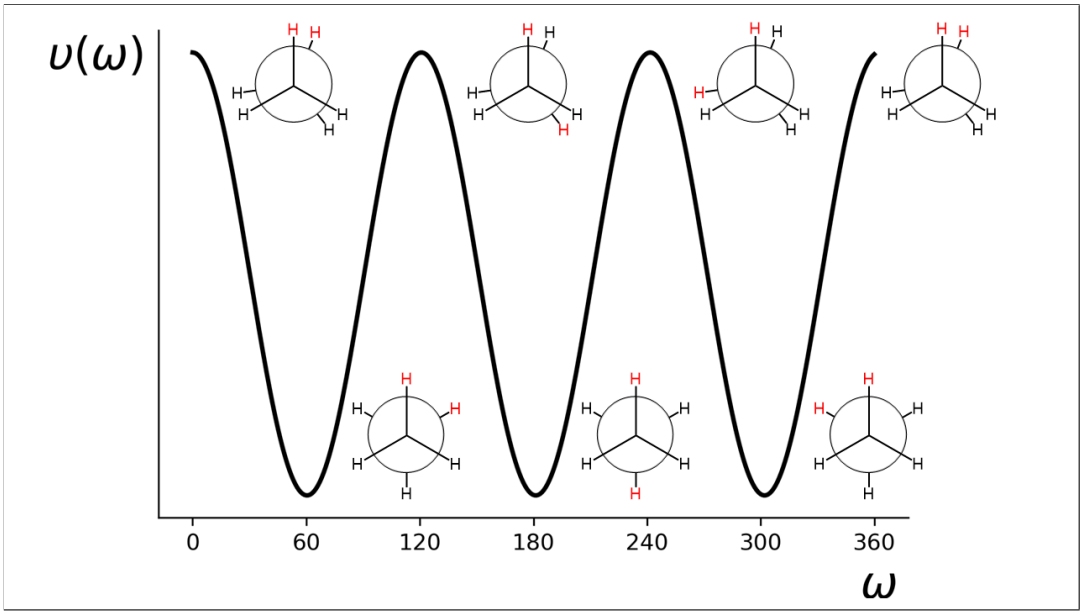

FIGURE 2.6: Schematic rotational profile of ethane.

where V_n gives a qualitative indication of the relative barrier height to rotation, n gives the multiplicity as number of minima in the function for a rotation of 360° and γ is the phase shift factor, providing the angular distance between every minimum.([19], pp. 173-176) In Figure 2.6 a schematic rotational profile for the torsion angle is shown with $V_n = 3$, $n = 3$ and $\gamma = 0$.

2.2.2 Non-bonded Interactions

Interactions between independent molecules or atoms are non-bonded interactions. These interactions are grouped into the electrostatic interactions and the van der Waals interactions. The electrostatic Interactions are calculated as a summation of interactions with Coulomb's law:

$$v(r) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}. \tag{2.19}$$

In this approach, the charge distribution is assumed as an arrangement of point charges q_i, restricted to the nuclear center of a molecule. Therefore, N is the number of point charges, and $\frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$ the interaction between two of these point charges. Here ϵ_0 is the vacuum permittivity and r_{ij} is the distance between the point charges.([19], pp. 181-203)

The van der Waals interactions are modeled as a Lennard-Jones-12-6 (LJ) potential[2]. The LJ-12-6 potential (Equation 2.20) consists of an attractive contribution and a repulsive contribution. The attractive contribution of the LJ-12-6 potential is a result of the dispersive forces. These dispersive interactions are due to the instantaneous dipoles arising from fluctuations. This instantaneous dipole then induces dipoles to neighboring atoms, starting a chain reaction. The repulsive contribution is due to the exchange forces. Following the Pauli principle, it is not allowed for two electrons in

[2]The 12th and 6th power are chosen by the power-law relationship. There is no theoretical reason to favor these powers besides that it is found in many terms to calculate the dispersion energy.

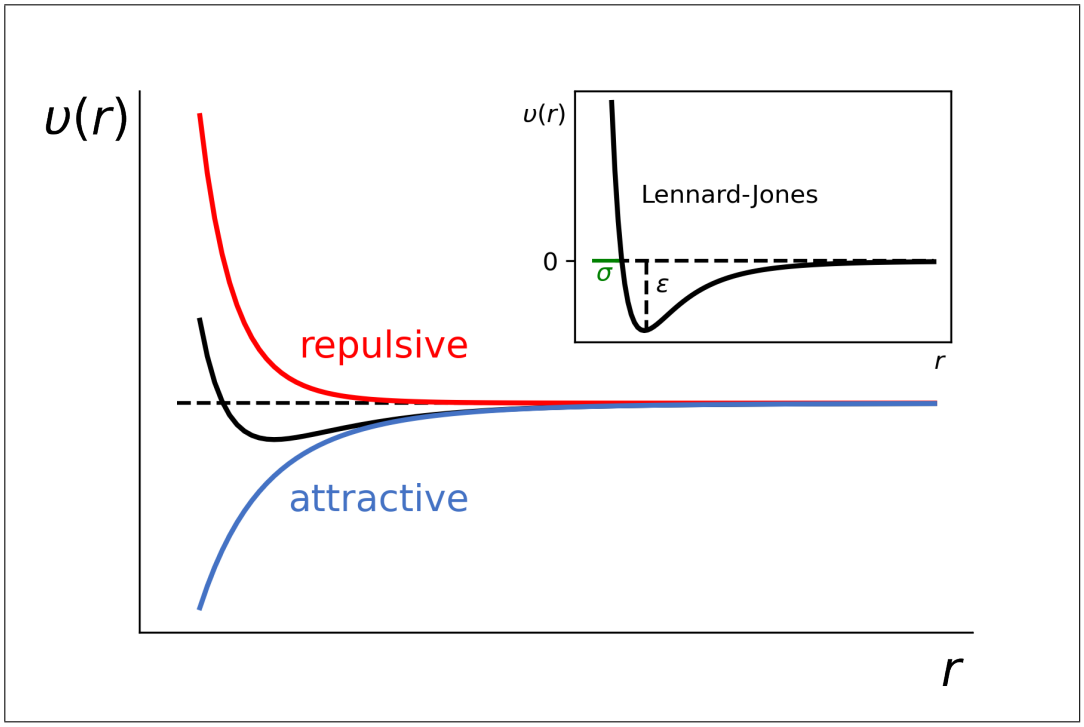

FIGURE 2.7: Lennard Jones potential (black) with attractive contribution (blue), repulsive contribution (red), well depth ϵ and collision diameter σ (green).

a system to have the same set of quantum numbers.[23] Systems with a separation of 3 Å and below violate the Pauli principle, resulting in a large increase in energy.

$$v(r) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} 4\epsilon_{ij} \left[\underbrace{\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12}}_{repulsive} - \underbrace{\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}}_{attractive} \right] \tag{2.20}$$

In Equation 2.20 σ_{ij} is the collision diameter[3] and ϵ_{ij}[4] the well depth. An illustration of the attractive contribution and the repulsive contribution to a Lennard-Jones potential is shown in Figure 2.7. The calculation of the non-bonded terms is computationally expensive, since all particles interact with each other. Therefore, a cut-off is introduced to the non-bonded terms, limiting the range of interaction. A commonly used distance is a cut-off of length 12 Å.([19], pp. 204-214; [2, 24, 25]) The resulting problems with the slow decay of the electrostatic interactions are treated with the Particle Mesh Ewald (PME) summation.[25, 26]

[3]The collision diameter σ_{ij} gives the separation r between two particles for which the energy is zero. In CHARMM σ_{ij} is calculated as the arithmetic mean $\frac{\sigma_{ii}+\sigma_{jj}}{2}$.

[4]In CHARMM ϵ_{ij} is calculated as the geometric mean $\sqrt{\epsilon_{ii} \cdot \epsilon_{jj}}$.

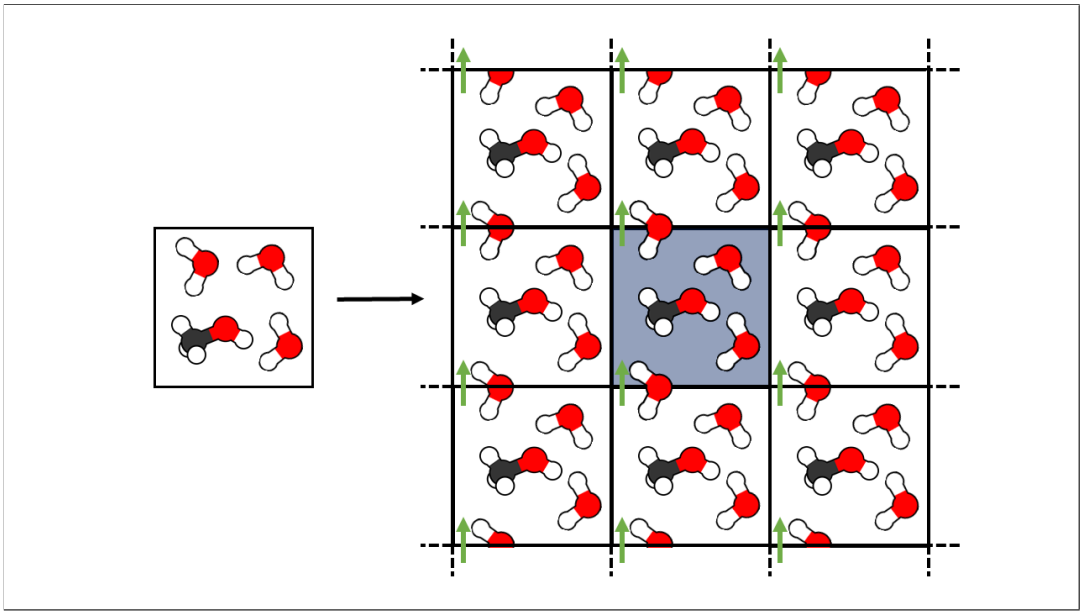

FIGURE 2.8: Schematic illustration of cubic periodic boundary conditions.

2.3 Additional Aspects

2.3.1 Boundary Conditions

Handling Boundary Conditions and the resulting surface interactions correctly is essential to calculate macroscopic properties[5] from molecular simulations. Depending on the properties of interest, handling boundaries differs a lot. For example, a simulation in a cubic box where most of the molecules interact with the box walls would not be appropriate setup to study bulk phenomena but instead surface interactions; e.g. in liquid drops. To study bulk properties only a fraction of the molecules should be influenced by the boundary effects. Having the same cubic box as before a multiple of the used molecules would be needed.([19], p. 317)

Periodic boundary condition (PBC) is a commonly used method to handle boundary effects without using large numbers of molecules. Here, a simulation box of choice is replicated in all directions. Therefore, the forces acting upon the particles in the central box are as they would be in a bulk fluid. The coordinates of the particles in the replicated boxes are calculated as integral multiples of the box side length and added or subtracted to the coordinates of the resembling particle in the central box. Whenever a particle leaves the central box it is replaced by a replicated particle entering the box from the opposite side. Therefore, the number of particles in the box remains constant. The simplest way to illustrate such a periodic boundary condition is illustrated in Figure 2.8. Here the cubic central box with the grey background is mirrored in a two-dimensional periodic array. If the water molecule in the top left corner of the central box would leave the box in direction of the green arrow it would be replaced by the water molecule in the bottom middle box. The cubic geometry is not essential. Other geometries like the rhombic dodecahedron and the truncated octahedron can be used instead of the cubic geometry.([2]; [19], pp. 317-319)

[5]Macroscopic quantities are, e.g., energy (E), enthalpy (H), entropy (S), Helmholtz free energy (F), Gibbs free energy (G), pressure (P), volume (V), temperature (T) and the number of particles (N)

2.3.2 Thermodynamic Ensembles

Statistical mechanics connect macroscopic quantities defined by the laws of thermodynamics with microscopic quantities[6] on the molecular level. A statistical ensemble consists of a large number of replicas of a system, taking all possible states into account simultaneously. A thermodynamic ensemble is a statistical ensemble in a statistical equilibrium, meaning that the probability distribution ρ of states is constant over time. Properties of interests are then calculated as the ensemble average $\langle A \rangle$ as sum of the property A_i over all states with respect to the probability density ρ_i (Equation 2.21).

$$\langle A \rangle = \sum_i \rho_i A_i \tag{2.21}$$

Different ensembles describe how the system is separated from its surroundings, from completely isolated to completely open to the surrounding of the system. The three most important thermodynamic ensembles in molecular simulations are the microcanonical, canonical, and isothermal-isobaric ensemble.([2]; [27], pp. 255-256)

In the microcanonical ensemble (NVE) the system is completely isolated from its surroundings. Therefore, the energy, the volume, and the number of particles are constant within the system. Since the system has no interactions with the surroundings, the temperature of the system is difficult to control. The microcanonical ensemble is produced by the direct integration of Newton's law of motion. In molecular simulations, this ensemble is used to represent systems with a specific total energy.([2]; [27], pp. 255-256)

In the canonical ensemble (NVT) the volume and number of particles are constant, while the total energy of the system is allowed to interact with the surroundings to keep the temperature of the system constant. The free energy of a system is called Helmholtz free energy F if the simulation is run with a thermostat modeling a canonical ensemble. This is explained in more detail in Section 2.5.([2]; [27], pp. 62-92)

The isothermal-isobaric ensemble (NPT) exchanges, similarly to the canonical ensemble, energy with its surroundings to keep the temperature constant. In contrast to the canonical ensemble, the isothermal-isobaric ensemble is also allowed to change the volume of the system. Due to the volume change, the pressure is now constant. Exchange of particles are not allowed resulting in a constant N. The free energy of a system is called Gibbs free energy G if the simulation is run with a thermostat and barostat modeling an isothermal-isobaric ensemble.([2]; [19], p. 307)

2.3.3 Thermostats in Molecular Simulation

The interaction of a system with a heat bath can be modeled through different approaches. These approaches are called thermostats. Generally, thermostats modify the equation of motion to apply to an ensemble with constant temperature. Global thermostats are acting on all the particles in one system with the same strength at once.[28] Examples for global thermostats are the Nosé-Hoover thermostat[29, 30] and the Berendsen thermostat[31]. The other group is called local thermostats, acting locally like the Langevin thermostat[32] and Andersen thermostat[33]. In the

[6] e.g. the interactions between the atoms and molecules forming the system

following, only the Langevin thermostat will be described because it is the only thermostat used in this thesis.

The Langevin dynamics[34], often referred to as the Langevin thermostat, modifies Newton's equation of motion by adding a friction and a noise term, resulting in

$$m_i\ddot{\vec{r}}_i = \overrightarrow{\mathscr{F}_i} - m_i\Gamma_i\dot{\vec{r}}_i + \vec{\xi}_i(t) \tag{2.22}$$

where $\mathscr{F}_i$ denotes the force on i, m_i the mass of i, Γ_i is the friction coefficient and $\vec{\xi}_i(t)$ is a Gaussian noise. The friction coefficient determines the relaxation time of the temperature. The noise term models the random collision of a particle with a solvent molecule according to the Brownian motion. In a physical system, the momentum inside the system is conserved and can not vanish. The Langevin thermostat does not conserve the momentum of the system, hence long-range hydrodynamic interactions are modeled incorrectly. Therefore, it is important to consider choosing the Langevin thermostat only for systems where hydrodynamics is not of importance.[2]

2.4 Monte Carlo Simulations

The Monte Carlo simulation is an alternative molecular simulation method. Monte Carlo simulation is a statistical approach using repeated random sampling solved numerically. In a chemical context, random changes are applied to a configuration, generating several different configurations of a system. For each of these configurations the potential energy $\mathscr{V}(r^N)$ is calculated and weighted with a Boltzmann factor $\exp(\mathscr{V}(r^N)/k_BT)$. Since it has been proven that these random samples of configurations are connected to thermodynamic properties, it is possible to calculate properties of interest using Equation 2.21. Here i is the random sample, ρ_i is the probability calculate with the Boltzmann factor and A_i the property of interest in the random sample. Random samples with non-physical configuration often have high energies which contribute greatly to the total potential energy of the system, resulting in poor convergence. Therefore, the Metropolis Method was introduced to the naive Monte Carlo algorithm.[35] The Metropolis Method generates a Markov chain[7] of states. Instead of creating random configurations and taking all of them into account, only configurations with a large contribution to the total potential energy $\mathscr{V}(r^N)$ are created. Therefore, every new configuration $i+1$ depends on the previous configuration i by calculating an acceptance ratio that is based on the Boltzmann factor. Configurations that are energetically close to the previous configuration have a higher probability of acceptance. In accordance with this assumption, starting with an equilibrated system results in an ensemble that is close to the equilibrium. Due to the Metropolis Method high-energy contributions of configurations, that are not of interest, are eliminated resulting in a modified Monte Carlo simulation that can calculate thermodynamic properties.([19], pp. 410-456)

[7] Markov chain is a stochastic model describing possible states $i+1$ in which the probability of each possible state, stored in the transition matrix π, depends only on the state i attained in the iteration before. The transition matrix π is of size NxN where N equals the number of possible states.([27], p. 259)

2.5 Free Energy Calculations

The free energy describes the stability of a system with regard to its environment. In a canonical ensemble (NVT) the free energy F is related to the partition function Q (Equation 2.24) as shown in Equation 2.23.([19], pp. 303-305 & pp. 410-412; [27], p. 17)

$$F = -k_B T \ln Q \tag{2.23}$$

$$Q_{NVT} = \frac{1}{N!}\frac{1}{h^{3N}} \iint dp^N dr^N \exp\left[-\frac{\mathscr{H}(\vec{p},\vec{r})}{k_B T}\right] \tag{2.24}$$

Properties in a thermodynamic ensemble are calculated as an ensemble average $\langle A \rangle$ (see Section 2.3.2) which is the average over the states of the systems. (Equation 2.25). A is the property of interest, which depends on the momenta p^N and the positions r^N. ρ is the probability density for a configuration having the momenta p^N and the positions r^N.([19], pp. 303-305; [27], pp. 255-256)

$$\langle A \rangle_{NVT} = \iint dp^N dr^N A(\vec{r}) \rho_{NVT}(\vec{p},\vec{r}) \tag{2.25}$$

The probability density ρ in the canonical ensemble is given by Equation 2.26. Here $\mathscr{H}(\vec{p}_i,\vec{r}_i)$ is the Hamiltonian, i.e. the total energy, Q_{NVT} the partition function (Equation 2.24) and $\beta = (k_B T)^{-1}$ with k_B the Boltzmann´s constant and T the temperature.([27], pp. 255-256)

$$\rho_{NVT}(\vec{p},\vec{r}) = \frac{\exp(-\beta \mathscr{H}(\vec{p},\vec{r}))}{Q} \tag{2.26}$$

or

$$\rho_{NVT}(\vec{p},\vec{r}) = \frac{\exp(-\beta \mathscr{H}(\vec{p},\vec{r}))}{\iint dp^N dr^N \exp(-\beta \mathscr{H}(\vec{p},\vec{r}))} \tag{2.27}$$

When using Cartesian coordinates, the total energy $\mathscr{H}(\vec{p}_i,\vec{r}_i)$ can be split into the kinetic energy $\mathscr{K}(\vec{p}_i)$ and potential energy $\mathscr{U}(\vec{r}_i)$ (Equation 2.28).([27], pp. 255-256)

$$\mathscr{H}(\vec{p},\vec{r}) = \mathscr{K}(\vec{p}) + \mathscr{U}(\vec{r}) \tag{2.28}$$

The probability density ρ can be factored as shown in Equation 2.29.

$$\rho_{NVT}(\vec{p},\vec{r}) = \overset{(p)}{\rho}_{NVT}(\vec{p}) \overset{(r)}{\rho}_{NVT}(\vec{r}) \tag{2.29}$$

with

$$\overset{(p)}{\rho}_{NVT}(\vec{p}) = \frac{\exp(-\beta \mathscr{K}(\vec{p}))}{\int dp^N \exp(-\beta \mathscr{K}(\vec{p}))} \tag{2.30}$$

and

$$\overset{(r)}{\rho}_{NVT}(\vec{r}) = \frac{\exp(-\beta \mathscr{U}(\vec{r}))}{\int dr^N \exp(-\beta \mathscr{U}(\vec{r}))}. \tag{2.31}$$

As shown in Equation 2.30 and 2.31 the probability density $\overset{(r)}{\rho}_{NVT}(\vec{r})$ for a certain configuration is independent of the probability density $\overset{(p)}{\rho}_{NVT}(\vec{p})$ of the momenta.([27],

pp. 255-256) If the quantity of interest depends only on the coordinates, it is possible to simplify Equation 2.25 to

$$\langle A \rangle_{NVT} = \int dr^N A(\vec{r}) \rho^{(r)}_{NVT}(\vec{r}) \underbrace{\int dp^N \rho^{(p)}_{NVT}(\vec{p})}_{=1}. \tag{2.32}$$

The denominator in Equation 2.31 is called the configurational partition function Z (Equation 2.33).

$$Z = \int dr^N \exp(-\beta \mathscr{U}(\vec{r})). \tag{2.33}$$

In Monte Carlo simulations and Molecular Dynamics simulations it is not possible to calculate the partition functions, e.g. Q or Z, directly from a simulation. Therefore, free energy calculations commonly calculate free energy differences between two states i and j (Equation 2.34).[11]

$$-\beta^{-1} \ln \frac{Z_j}{Z_i} = -k_B T \ln \frac{\int dr_j^N \exp(-\beta \mathscr{U}_j(\vec{r}))}{\int dr_i^N \exp(-\beta \mathscr{U}_i(\vec{r}))} = \Delta F_{ij} \tag{2.34}$$

States in free energy calculations are often referred to as λ states with the initial state $\lambda = 0$ and the end state $\lambda = 1$. The states i and j are two λ states with the potential energies $\mathscr{U}_i = \mathscr{U}(\lambda_i)$ and $\mathscr{U}_j = \mathscr{U}(\lambda_j)$ with

$$\mathscr{U}(\lambda) = (1-\lambda)\mathscr{U}(\lambda = 0) + \lambda \mathscr{U}(\lambda = 1) \quad 0 \leq \lambda \leq 1. \tag{2.35}$$

Common methods to calculate the free energy differences from simulation data are the Zwanzig Relationship, Thermodynamic Integration (TI), Bennett Acceptance Ratio (BAR), Multistate Bennett Acceptance Ratio (MBAR), and Weighted Histogram Analysis Method (WHAM).

2.5.1 Alchemical Sampling Methods

Zwanzig Relationship

The Zwanzig relationship, also known as Free Energy Perturbation or Exponential Averaging (EXP), relates the different potential energies of slightly distinct states with their thermodynamic properties.[36] Given two well defined states i and j, the free energy difference ΔF between $\mathscr{U}_i(\vec{r})$ and $\mathscr{U}_j(\vec{r})$ can be calculated as:

$$\begin{aligned} \Delta F &= F_j - F_i \\ &= -\beta^{-1} \ln \frac{Z_j}{Z_i} \\ &\dots \\ &= -\beta^{-1} \ln \langle \exp(-\beta(\mathscr{U}_j(\vec{r}) - \mathscr{U}_i(\vec{r}))) \rangle_i \\ &= -\beta^{-1} \ln \langle \exp(-\beta \Delta \mathscr{U}_{ij}(\vec{r})) \rangle_i. \end{aligned} \tag{2.36}$$

To indicate which initial state is used for averaging over the ensemble of configurations, the subscripts i or j are used. Subscript i refers to initial state i and j to initial

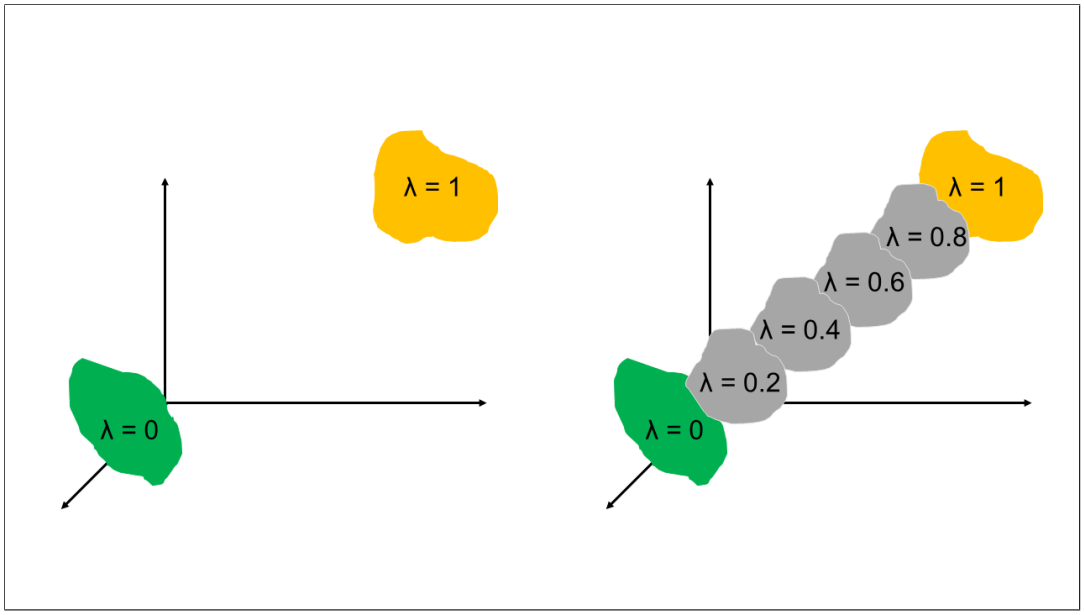

FIGURE 2.9: Overlap in phase space can improve through a series of intermediate states. Overlap between initial state ($\lambda = 0$) and final state ($\lambda = 1$) on the right side. A series of intermediate states to improve overlap on the left side.

state j.([19], pp. 564-566; [11]) For an initial state j the free energy difference ΔF can be calculated as:

$$\Delta F = -\beta^{-1} \ln\langle \exp(-\beta(\mathscr{U}_i(\vec{r}) - \mathscr{U}_j(\vec{r})))\rangle_j = -\beta^{-1} \ln\langle \exp(-\beta\Delta\mathscr{U}_{ji}(\vec{r}))\rangle_j. \tag{2.37}$$

Since this method converges very poorly if states i and j have little or no phase space overlap[8], a series of intermediate states between $\lambda = 0$ and $\lambda = 1$ are introduced (Figure 2.9).([19], pp. 564-566; [39, 40]) An example with one intermediate state I ($\lambda = 0.5$) with the potential $\mathscr{U}_I(\vec{r})$, the initial state i ($\lambda = 0$) and the end state j ($\lambda = 1$) is shown in Equation 2.38.

$$\begin{aligned} \Delta F &= (F_j - F_i) + (F_I - F_i) \\ &= -\beta^{-1} \ln\left[\frac{Z(j)}{Z(I)} \cdot \frac{Z(I)}{Z(i)}\right] \\ &\dots \\ &= -\beta^{-1} \ln\langle \exp(-\beta(\mathscr{U}_j(\vec{r}) - \mathscr{U}_I(\vec{r})))\rangle_I - \langle \exp(-\beta(\mathscr{U}_I(\vec{r}) - \mathscr{U}_i(\vec{r})))\rangle_i \end{aligned} \tag{2.38}$$

This can be extended to the usage of multiple intermediate states to connect two states that do not overlap in phase space. The transformation from the initial to the final state is split up into a series of intermediate states with sufficient overlap between each neighboring state. Since only the end states are of importance, the

[8] In the canonical ensemble, the weight of configuration depends on its energy.[37] Thus, it suffices to consider the energy of a configuration when using Hamiltonian A to that of Hamiltonian B. The overlap in configurations (phase space) between two systems A and B can, therefore, be projected on the probability distributions of $\Delta\mathscr{U}(A \to B)$ for configurations sampled using Hamiltonian A and $\Delta\mathscr{U}(B \to A)$ for configurations sampled with Hamiltonian B, and in the context of free energy simulations, the overlap between these two distributions is used as a measure of phase space overlap.[38]

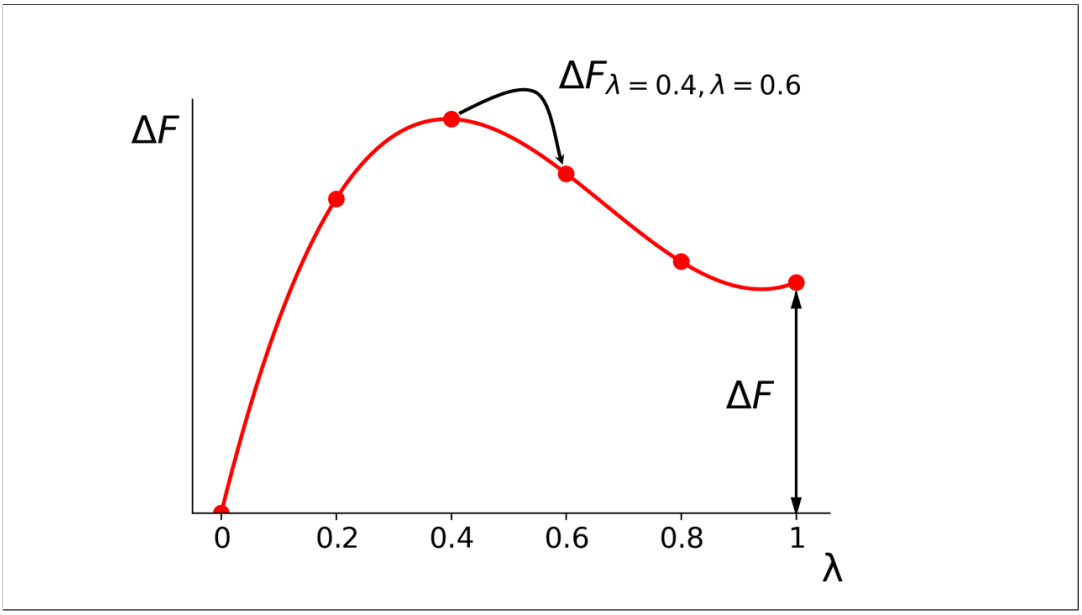

FIGURE 2.10: The schematic representation of the Zwanzig approach. To obtain the total free energy difference between $\lambda = 0$ and $\lambda = 1$, the free energy difference between all neighboring states is calculate as indicate for $\lambda = 0.4$ and $\lambda = 0.6$.

intermediate states can have non-physical forms as long as states are overlapping in terms of phase space. Common examples for non-physical states are atoms without charges, van der Waals, or Lennard Jones potentials.[11] In a series of K overlapping states, where $\lambda_i = 0$ is the initial state and $\lambda_j = 1$ is the end state, the total free energy difference ΔF is the sum over all free energy differences $\Delta F_{K,K+1}$ (Equation 2.39).

$$\Delta F_{i,j} = \sum_{K=i}^{K=j} \Delta F_{K,K+1} \tag{2.39}$$

Hence, the potential energy of the system is dependent on the λ state and the corresponding configuration $\vec{r}$. Simulations of $\mathscr{U}(\lambda, \vec{r})$ over a series of λ states generate the samples which are necessary to calculate $\Delta F_{K,K+1}$ for each overlapping state (Figure 2.10).[11]

Thermodynamic Integration

A different approach to compute free energy differences is Thermodynamic Integration (TI). This method was established by John G. Kirkwood in 1935.[41] The free energy difference ΔF is the integral over the free energy F as a continuous function of λ (Equation 2.40).

$$\Delta F = \int_{\lambda=0}^{\lambda=1} \frac{dF(\lambda)}{d\lambda} d\lambda \tag{2.40}$$

The free energy as function of λ is calculated as:

$$F(\lambda) = -\beta^{-1} \ln Z(\lambda). \tag{2.41}$$

Inserting Equation 2.41 into Equation 2.40:

$$\Delta F = -\beta^{-1} \int_{\lambda=0}^{\lambda=1} \frac{\partial \ln Z(\lambda)}{\partial \lambda} d\lambda = \int_{\lambda=0}^{\lambda=1} \frac{-\beta^{-1}}{Z(\lambda)} \frac{\partial Z(\lambda)}{\partial \lambda} d\lambda. \tag{2.42}$$

With the definition of Z from Equation 2.33, $\frac{\partial Z(\lambda)}{\partial \lambda}$ can be written as:

$$\frac{\partial Z(\lambda)}{\partial \lambda} = \int dr^N \frac{\partial}{\partial \lambda} \exp(-\beta \mathscr{U}(\vec{r}, \lambda)) = \beta \int dr^N \frac{\partial \mathscr{U}(\vec{r}, \lambda)}{\partial \lambda} \exp(-\beta \mathscr{U}(\vec{r}, \lambda)). \tag{2.43}$$

Substituting Equation 2.43 back into Equation 2.42 gives the formula for free energy difference in Thermodynamic Integration (Equation 2.44).([19], p. 568 & pp. 630-631; [11])

$$\Delta F = \int_{\lambda=0}^{\lambda=1} \left\langle \frac{\partial \mathscr{U}(\vec{r}, \lambda)}{\partial \lambda} \right\rangle_{\lambda} d\lambda \tag{2.44}$$

To solve the integral from Equation 2.44 a series of simulations of different values of λ are performed. For each value of λ the average of

$$\left\langle \frac{\partial \mathscr{U}(\vec{r}, \lambda)}{\partial \lambda} \right\rangle_{\lambda} d\lambda \tag{2.45}$$

is determined. The total free energy difference is the area under the graph of this average overall λ values (Figure 2.11). The discussion of the efficiency of the Thermodynamic Integration in comparison to the Zwanzig approach is still ongoing.[11, 42]

$$\Delta F = \sum_{k=1}^{K} w_k \left\langle \frac{\partial \mathscr{U}(\vec{r}, \lambda)}{\partial \lambda} \right\rangle_{k} \tag{2.46}$$

Since the number of intermediate states that can be simulated is limited, numerical integration is used to solve the integral (Equation 2.46). Here w_k are the corresponding weights depending on the choices of the integration method. Some different integration methods have been studied.[43–45]

In this case one commonly known example for a simple, but flexible and robust method is the trapezoid rule. The weights for equally distributed points are $w_k = \frac{1}{K-1}$ except for the start and end points. The weights for the initial state and the end state are $w_1 = w_K = \frac{1}{2(K-1)}$. For unequal spacing between the points $w_1 = \frac{\lambda_2 - \lambda_1}{2}$ and $w_K = \frac{\lambda_K - \lambda_{K-1}}{2}$ holds true. For points $k \neq 1, K$ the weight is determined through $w_k = \lambda_{k+1} - \lambda_{k-1}$. TI is fairly simple to use for linear pathways, i.e. if Equation 2.47 holds true.

$$\frac{\partial \mathscr{U}(\vec{r}, \lambda)}{\partial \lambda} = \frac{\partial}{\partial \lambda}[(1-\lambda)\mathscr{U}_0(\vec{r}, \lambda) + \lambda \mathscr{U}_1(\vec{r}, \lambda)] = \mathscr{U}_1(\vec{r}, \lambda) - \mathscr{U}_0(\vec{r}, \lambda) \tag{2.47}$$

The method for non-linear pathways is more complex, due to the need for an analytical calculation of the numerical derivatives. Also, the number of intermediate states must be large enough to reduce the bias in the integration below the statistical noise.[11, 42]

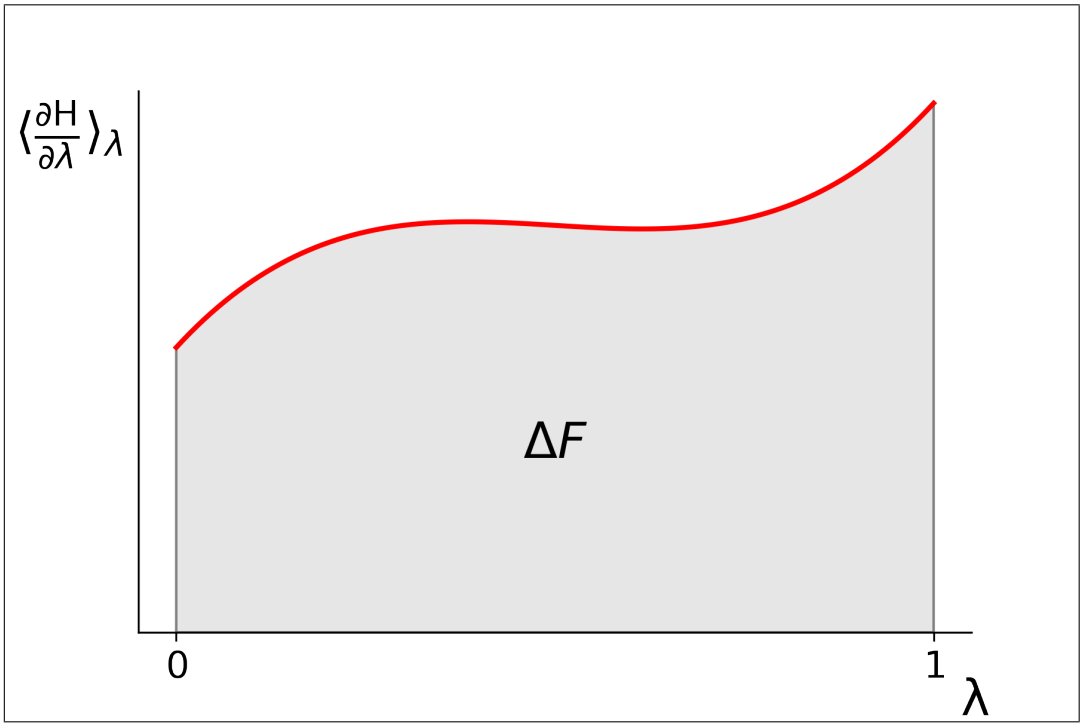

FIGURE 2.11: Calculation of free energy difference by Thermodynamic Integration.

Bennett Acceptance Ratio

The Bennett Acceptance Ratio (BAR) first developed by Charles H. Bennett in 1976 uses a double-ended approach to calculate the free energy difference between two states.[46] Therefore, two potentials $\mathscr{U}_i(\vec{r})$ and $\mathscr{U}_j(\vec{r})$ acting on the same configuration space $\vec{r}$ are set in relation through the difference in potential energies $\Delta\mathscr{U}_{ij}(\vec{r})$ along the pathway between two states $\lambda_i = 0$ and $\lambda_j = 1$ (Equation 2.48). To obtain ΔF_{ij} two simulations are needed, one of state $\lambda_i = 0$ and one of state $\lambda_j = 1$.[47]

$$\Delta F_{ij} = -\beta^{-1} \ln \frac{Z_j}{Z_i} = \beta^{-1} \ln \frac{\langle \alpha(\vec{r}) \exp[-\beta \mathscr{U}_i(\vec{r})] \rangle_j}{\langle \alpha(\vec{r}) \exp[-\beta \mathscr{U}_j(\vec{r})] \rangle_i} \tag{2.48}$$

Bennett's approach holds true for all $\alpha(\vec{r}) > 0$. Since Equation 2.48 is an exact function of distributions, the optimal pathway can be obtained by minimizing the variance of the free energy through a weighting function $\alpha(\vec{r})$. This results in an implicit function of ΔF which is numerical solvable (Equation 2.50).[11, 46]

$$\sum_{i=1}^{n_i} \frac{1}{1 + \exp(\ln(n_i/n_j) + \beta\Delta\mathscr{U}_{ij} - \beta\Delta Fij))} \tag{2.49}$$

$$-\sum_{j=1}^{n_j} \frac{1}{1 + \exp(\ln(n_j/n_i) - \beta\Delta\mathscr{U}_{ji} + \beta\Delta F_{ji}))} = 0 \tag{2.50}$$

Here n_i and n_j refer to the number of samples from each state. $\Delta\mathscr{U}_{ij}$ refers to the difference in potential energy while sampled from state j and $\Delta\mathscr{U}_{ji}$ to the difference in potential energy while sampled from state i. Comparing BAR with the FEP method

various studies have shown the superiority of BAR.[39, 40, 46, 48] Since FEP estimates the potential energy of the end state as an extrapolation of the start state it requires significantly more overlap between the configurational space of each state to converge.[48] Using potentials in both directions BAR is more robust in the estimation of free energies. Even though BAR is less dependent on overlaps than FEP, an overlap still must exist between the states.[11, 39, 40, 46, 48] The direct comparison of TI and BAR is hardly possible, due to the theoretical difference in the two methods using different information to calculate the free energy difference. On a practical level BAR is able to outperform TI, since it needs fewer intermediate states for an equivalent level of precision.[11, 39, 40, 49, 50]

Multistate Bennett Acceptance Ratio

The Multistate Bennett Acceptance Ratio (MBAR) is based on BAR and uses a multistate approach. A MBAR approach with two sample states would be equivalent to BAR. Instead of a single weighting function as in BAR (Equation 2.48), MBAR operates on a set of $K \times K$ weighting functions $\alpha_{ij}(\vec{r})$ with K as the number of states.[51] To minimize the variance, the weighting functions $\alpha_{ij}(\vec{r})$ are expressed as:

$$\alpha_{ij}(\vec{r}) = \frac{N_j \widehat{c}_j^{-1}}{\sum_{k=1}^{K} N_k \widehat{c}_k^{-1} \exp\left(-\beta \mathscr{U}_k(\vec{r})\right)}. \tag{2.51}$$

Here $\widehat{c}_j$ and $\widehat{c}_k$ are normalized constants and N_K is the number of uncorrelated equilibrium samples from each state K.[52]

Weighted Histogram Analysis Method

Another not so commonly used approach in alchemical calculations is the Weighted Histogram Analysis Method (WHAM) first introduced by Alan M. Ferrenberg and Robert H. Swendsen in 1989. This method uses the information of all intermediate states along a pathway and analyses this information at once.[53] The underlying principle is that given a discrete number of states, it is possible to create a histogram with discrete bins[9] which provides a relative probability of observing the states of interest. Instead of trying to calculate the ratios between partition functions as shown in Equation 2.34, the partition function Q_i is redefined. An unknown density of states for each of the simulated thermodynamic states Ω_i is added to the partition function Z_i (Equation 2.52).[54]

$$Z_i = \int dr^N \Omega_i(\vec{r}) \exp(-\beta \mathscr{U}(\vec{r})) \tag{2.52}$$

The density of states Ω_i for a simulation run i is defined as:

$$\Omega_i(\vec{r}, \lambda_i) = B_i(\vec{r}, \lambda_i) \exp\left[\left(\sum_{j=0}^{K} \beta \lambda_{j,i} \mathscr{U}(\vec{r}_j)\right) - \beta F_i\right]. \tag{2.53}$$

[9] Data binning is a form of quantization. The original data set is grouped in smaller intervals called 'bin'. Afterward, each bin is replaced by a representative value which is depending on the context (i.e. the number of residues of each bin; the mean over the bin)

Here B_i is the representative value of the bin for simulation i evaluated at $\vec{r}$ and λ_i. λ_i as available set of states during simulation i with $\{\lambda\}_i = \{\lambda_1, \lambda_2, \ldots \lambda_K\}_i$. The density of states can be estimated over a number of simulation runs S as:

$$\Omega_i(\vec{r}) = \sum_{i=1}^{S} \omega_i(\vec{r}) B_i(\vec{r}, \lambda_i) \tag{2.54}$$

with

$$\Omega_i(\vec{r}) = \sum_{i=1}^{S} \omega_i(\vec{r}) = 1. \tag{2.55}$$

The best weighting function $\omega_i(\vec{r})$ minimizes the error of B_i over all simulation runs S. The error $\delta^2 B_i$ for a single B_i is

$$\delta^2 B_i = g_i \langle B_i \rangle \tag{2.56}$$

with $g_i = 1 + 2\tau_i$. τ_i is the correlation time of a given simulation run i. The expectation value $\langle B_i \rangle$

$$\langle B_i \rangle = N_i \Omega \exp\left(\beta F_i - \beta \sum_{j=0}^{K} \lambda_{j,i} \mathscr{U}(\vec{r}_j)\right) \tag{2.57}$$

Inserting these assumptions back into Equation 2.52 and taking a numerical integration approach similar to the one shown in Equation 2.46 results in the final WHAM partition function Z_i^{WHAM}:

$$Z_i^{WHAM} = \sum_{j=1}^{K} \frac{\sum_{x=1}^{S} g_x^{-1} B_x(\vec{r}, \lambda_j) \exp\left[-\beta\lambda_0\vec{r}_0 - \beta \sum_{l=1}^{K} \lambda_l \mathscr{U}_l(\vec{r})\right]}{\sum_{y=1}^{S} g_y^{-1} N_y \exp\left[\beta F_y - \beta\lambda_0\vec{r}_0 - \beta \sum_{m=1}^{K} \lambda_{m,y} \mathscr{U}_m(\vec{r})\right]}. \tag{2.58}$$

The subscript 0 refers to the unchanged initial state. The variables x and y are pointing to a single run in different sets of simulation runs. Additionally, variables l and m are referring to states within the respective simulation run. To reduce the final form of the WHAM equation to a simpler form, it is possible to set the bin width to zero. This results in a simpler expressing for the free energy F_i of each state with a total of K states (Equation 2.59).[11, 54, 55]

$$F_i = -\beta^{-1} \ln \sum_{k=1}^{K} \sum_{n=1}^{N_k} \frac{\exp[-\beta \mathscr{U}_i(\vec{r}_{kn})]}{\sum_{k'=1}^{K} N_{k'} \exp[\beta F_{k'} - \beta \mathscr{U}_{k'}(\vec{r}_{kn})]} \tag{2.59}$$

Here is $i = 1, ..., K$ and $\vec{r}_{kn}$ the nth sample of the kth with their correlated potential energy $\mathscr{U}_i$. To calculate the free energy difference of a system one of the free energies F_i must be fixated. Since WHAM is using all information of every state simultaneously, it reduces the number of computational cycles and loops. This results in overall increased efficiency.[11, 54, 55]

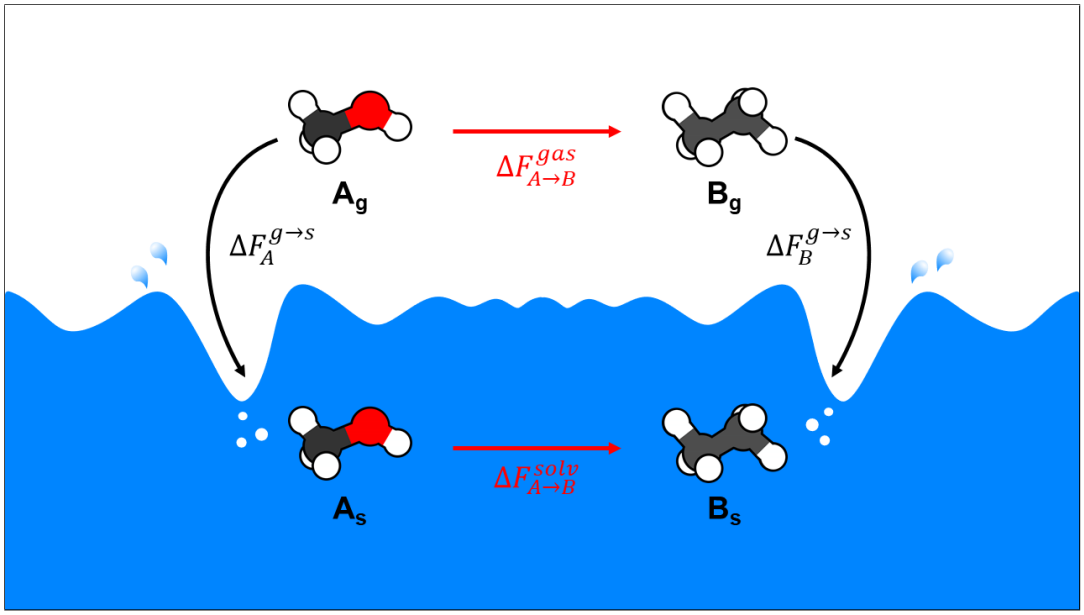

FIGURE 2.12: Thermodynamic cycle for the relative solvation free energy between molecule A and B.

2.5.2 Thermodynamic Cycles

The free energy is a state function. Therefore, any pathway[10] between two end states gives the same free energy. This principle is used to calculate relative free energies by substituting physical pathways with non-physical pathways to avoid cost-intensive calculations. The solvation free energy difference $\Delta\Delta F_{solv}$ for two solutes A and B could be calculated as the difference between the absolute solvation free energy of A and that of B (Figure 2.12 & Equation 2.60 in black), as would be done in an experimental approach. Since the solvation of a molecule requires a lot of reorganization of the solvent, long simulation times are needed. In contrast to physical experiments, simulations have the freedom to use a non-physical pathway (alchemical path). By transforming molecule A into molecule B only small parts of the system are changed. Calculating the free energy difference of molecule A and molecule B in the gas phase and solution and subtracting these from each other (Figure 2.12 & Equation 2.60 in red) results in the same relative free energy $\Delta\Delta F_{solv}$ as for the physical pathway. These four states form a thermodynamic cycle. For a closed thermodynamic cycle the energy around the circle has to be zero and $\Delta F_B^{g\to s} - \Delta F_A^{g\to s} = \Delta F_{A\to B}^{solv} - \Delta F_{A\to B}^{gas}$ holds true.[56]

$$\Delta\Delta F_{solv} = \Delta F_B^{g\to s} - \Delta F_A^{g\to s} = \Delta F_{A\to B}^{solv} - \Delta F_{A\to B}^{gas}. \tag{2.60}$$

2.5.3 Intermediate States

As discussed in Section 2.5.1, intermediate states are required for most free energy calculations and their construction is one of the critical aspects of free energy calculations. Since the initial state and end state may differ in the number of atoms, it is necessary to turn excess atoms into dummy atoms. Dummy atoms are placeholders that do not have non-bonded interactions with the system. The bonded terms are

[10] Series of transformations between two end states.

fully intact and attach the dummy atom to the molecule. To avoid the so-called van der Waals endpoint problem[57] (described below), methods like soft-core potentials[58, 59] or the serial atom insertion (SAI)[60] are used.

Van der Waals End Point Problem

When creating or annihilating particles in the presence of solvent, a sampling problem, known as the van der Waals endpoint problem, occurs. The space close to a particle (its van der Waals radius) is normally inaccessible to solvent particles. If, however, a particle is "decoupled"[11], this space becomes completely accessible as soon as the Lennard-Jones interactions are completely turned off, resulting in poor convergence. This leads to numeric instabilities in the calculation near the end state.[57–59]

Soft-core Potentials

The standard method to avoid the van der Waals end point problem is the introduction of a soft-core potential. Here the interatomic distance r_{ij} in the Lennard-Jones potential (Equation 2.20) is coupled to a shifting parameter δ, which is scaled depending on λ.[58, 59] Therefore, the following expression (as in [58]) for the van der Waals interactions between the particles i and j is obtained:

$$v(r_{ij}, \lambda) = \lambda 4\epsilon_{ij} \left(\frac{\sigma_{ij}^{12}}{[r_{ij}^2 + \delta(1-\lambda)]^6} - \frac{\sigma_{ij}^{6}}{[r_{ij}^2 + \delta(1-\lambda)]^3} \right) \Bigg] \tag{2.61}$$

Equation 2.61 is written for the creation of a particle. At $\lambda = 0$ the van der Waals interaction between the particles i and j are completely turned off. At $\lambda = 1$ the van der Waals interaction is described through a fully intact Lennard-Jones potential. For $0 \leq \lambda < 1$ the van der Waals interaction are described through a soft-core potential. The shifting parameter δ contributes more to the denominator term as λ gets closer to zero.

Soft-core potentials are mainly used in free energy calculations. Therefore, the dedicated code is often interlaced with the FES code, which has no GPU support most of the time.

Serial Atom Insertion

If a suitable soft-core implementation is unavailable or if the GPU acceleration is missing, Serial Atom Insertion (SAI) can be used instead.[60] Since the numerical instabilities only appear near the end states $\lambda = 0$ or $\lambda = 1$, but not exactly at these states, the Lennard-Jones interaction of a particle is either fully intact or fully turned off, without scaling. This can be done atom by atom or in batches of two atoms at a time. The SAI approach can be used with plain molecular dynamics and does not require specialized code like the soft-core potential. Thus, if the molecular dynamics code used is GPU accelerated, so will SAI. As described in detail in [60], free energy simulations utilizing SAI require (m)BAR as an alchemical method.

[11] Decoupling an atom from the system refers to the process of turning off all intermolecular interactions.

2.5.4 Common Core Approach

The SAI approach as just described serves to turn on or off interactions with one part of the system, a ligand or solute as a whole, i.e., it is suited to compute absolute free energy differences. In many applications, however, one is interested primarily in relative free energy differences, exploiting thermodynamic cycles (cf. Section 2.5.2 and Figure 2.12). The central step is alchemical transformations $X \rightarrow Y$, such as transforming methanol to ethane in Figure 2.12. Often, free energy differences for a series of solutes or ligands need to be computed, which share a common substructure. In such cases, the generic $X \rightarrow Y$ can be written as

$$CC - R_1 \rightarrow CC - R_i \quad (i = 2, \ldots, N) \tag{2.62}$$

In the ethane/methanol example, the common substructure "CC" is the methyl group. In search and prediction algorithms used in drug design, maximum common substructures (MCS) are frequently used.[61, 62] In alchemical free energy calculations the maximum common substructure approach is used to group structurally related compounds before calculating the relative free energies of the compounds to one compound of the group.[63] Yet, each alchemical transformation in Equation 2.62, changing R_1 to R_i $(i = 2, ..., N)$ needs to be set up separately with attention to detail. The so-called common core (CC) approach developed by S. Boresch and co-workers determines the maximum common substructure between two molecules A and B (or a series of ligands) and inserts it as an additional endpoint in the free energy calculations.[15] Rather than carrying out free energy simulations according to Equation 2.62, one considers transformations of the form

$$CC - R_i \rightarrow CC - DR_i \quad (i = 1, \ldots, N) \tag{2.63}$$

where DR_i indicates that the group has been transformed into dummy atoms. If dummy atoms are treated correctly (M.Fleck, M.Fleck and S.Boresch, manuscript submitted[64]), any contribution to the partition function they may have is identical along parallel paths of a thermodynamic cycle; in other words, the presence of dummy atoms has no influence on the desired relative free energy difference. Specifically,

$$\begin{aligned} \Delta\Delta F(CC - R_1 \rightarrow CC - R_2) = \Delta\Delta F(CC - R_2 \rightarrow CC - DR_2) \\ -\Delta\Delta F(CC - R_1 \rightarrow CC - DR_1). \end{aligned} \tag{2.64}$$

Since the maximum common substructure is based on elements, it can differ in bonded and non-bonded interactions for each molecule. These different maximum common substructures are further referred to as CC_A and CC_B. In such cases, as an additional step "common core" CC_A needs to by transformed into CC_B. To sum up, the free energy differences for the transformation in gas phase ($\Delta F^{gas}_{A\rightarrow B}$) and in solvent ($\Delta F^{solv}_{A\rightarrow B}$) are obtained by

$$\Delta F^{gas}_{A\rightarrow B} = \Delta F^{gas}_{B\rightarrow CC_B} - \Delta F^{gas}_{A\rightarrow CC_A} - \Delta F^{gas}_{CC_A\rightarrow CC_B} \tag{2.65}$$

and

$$\Delta F^{solv}_{A\rightarrow B} = \Delta F^{solv}_{B\rightarrow CC_B} - \Delta F^{solv}_{A\rightarrow CC_A} - \Delta F^{solv}_{CC_A\rightarrow CC_B}. \tag{2.66}$$

As an example of how the common core approach is used in this work, consider the modified thermodynamic circle to compute the relative solvation free energy calculation between methanol (A) and ethane (B) in Figure 2.13. The common core

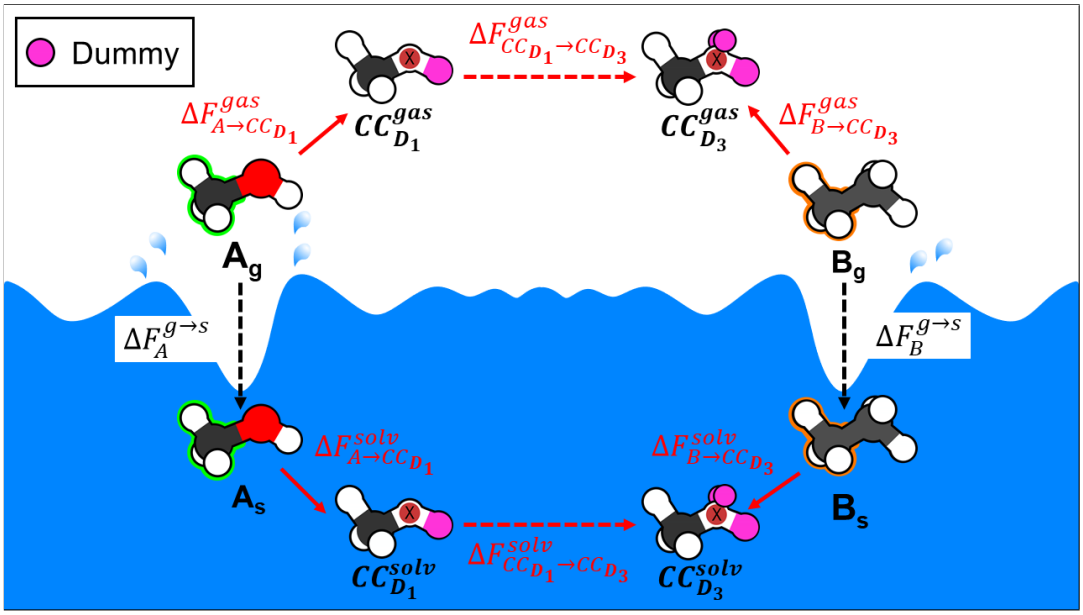

FIGURE 2.13: Thermodynamic cycle for the relative solvation free energy between molecule A and B. The CC_A is highlighted in green and the CC_B in orange. Dummy atoms are shown in a lightpurple.

is not a methyl group, but a hypothetical moiety methyl-X (CH_3X), where X is a Lennard-Jones particle without charge. One needs to compute the free energy difference $\Delta F_{A\rightarrow CC_{D_1}}$ between methanol and the common core methyl-X (green) with one dummy atom (CC_{D_1}), as well as the free energy difference $\Delta F_{B\rightarrow CC_{D_3}}$ between ethane and CH_3X (orange) with three dummy atoms (CC_{D_3}). The two common cores CC_{D_1} and CC_{D_3} only differ by the number of dummy atoms. The calculation of the free energy difference $\Delta F_{CC_{D_1}\rightarrow CC_{D_3}}$ is not necessary since in a relative free energy calculation $\Delta\Delta F_{CC} = \Delta F^{solv}_{CC_{D_1}\rightarrow CC_{D_3}} - \Delta F^{gas}_{CC_{D_1}\rightarrow CC_{D_3}} = 0$ has to be true.[64] The relative solvation free energy is then obtained by inserting Equation 2.65 and 2.66 into Equation 2.60. Changing of a chemical moiety to dummy atoms, $CC - R_1 \rightarrow CC - DR_1$ can be done either via soft-cores or via SAI. If a tranformation of the type $CC_A \rightarrow CC_B$ is required, intermediate states can be generated by linearly scaling the bond and non-bonded energy terms differing between the two end states.

Chapter 3

Methods

In general, free energy calculation protocols extract energetic information from equilibrium samples of all states needed to connect the configurations of interest. The samples are generated by molecular dynamics simulations. To make free energy simulations more stable and easier to use, supporting software packages have been developed for various program packages (e.g. [12]). Since the CHARMM community is lacking such a tool, a CHARMM[7] backend was implemented into the TRANSFORMATO[14] package.

3.1 Transformato

The TRANSFORMATO[14] package implements the Common Core approach described in Section 2.5.4. SAI is used to avoid the van der Waals endpoint problem when transforming atoms into dummy atoms. Therefore, all molecular dynamics simulations can be run as plain CHARMM MD jobs.

To calculate free energy differences in TRANSFORMATO, two transformations are needed to create a thermodynamic cycle(i.e. the transformation of methanol to ethane in vacuum and the transformation of methanol to ethane in solution (cf. Section 2.5.4 and Figure 2.13)). The needed initial states[1] are typically generated with CHARMM-GUI[7, 65, 66]. CHARMM-GUI provides tools to set up molecular dynamics calculations starting from PDB or mol2 files. TRANSFORMATO uses the output files from CHARMM-GUI to set up the transformation from the initial state to the common core as follows. First, the charges of all atoms not part of the common core are linearly scaled to zero. Then, the Lennard-Jones interactions of all hydrogens not part of the common core are turned off. Next, the Lennard-Jones interactions of the heavy atoms not part of the common core are turned off using the SAI approach (cf. Section 2.5.3). The only exception is the atom connecting the common core to the chemical moiety which has been transformed into dummy atoms. This atom is transformed into a particle with Lennard-Jones parameters between the atom types it is replacing in an additional intermediate step. At this stage, the common core of molecule A (CC_A) and the common core of molecule B (CC_B) only differ by the number of dummy atoms. If needed, CC_A is mutated to CC_B (cf. Section 2.5.4). This is repeated for every molecule to the common core.

For a user, the general workflow consists out of the following steps. Starting from a PDB or mol2 file the user generates input files for TRANSFORMATO with CHARMM-GUI.

[1] In the case of relative solvation free energy differences one state with the molecule in vacuum and one in solution is needed for each of the molecules. For example, the relative solvation free energy difference of methanol to ethane is computed from four initial states(methanol in vacuum, methanol in a waterbox, ethane in vacuum and ethane in a waterbox

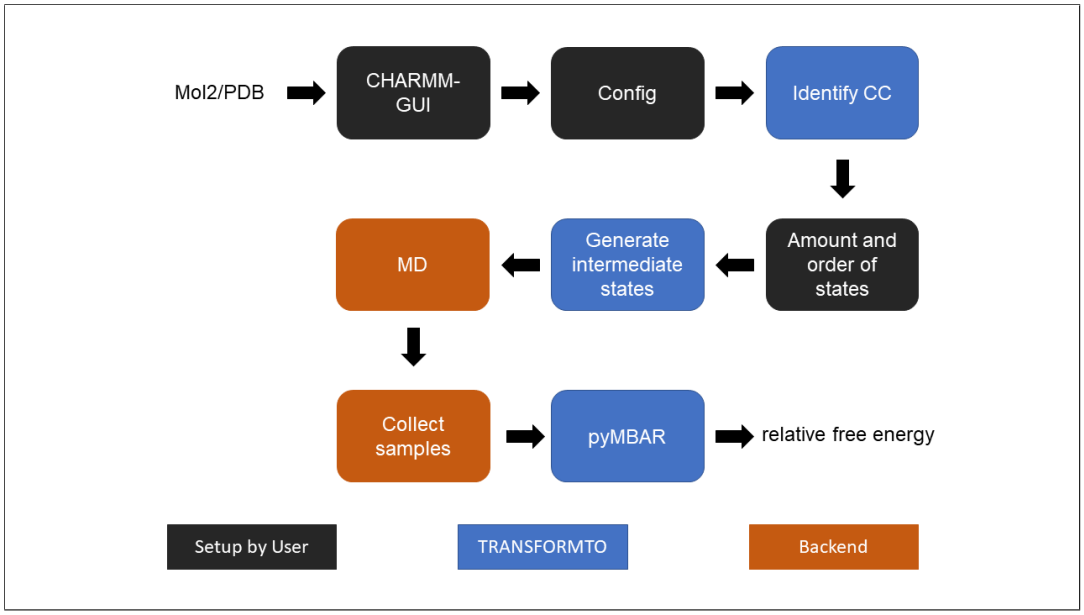

FIGURE 3.1: General workflow with TRANSFORMATO. Preparations done by a user with external tools are shown in grey, processes inside TRANSFORMATO are highlighted in blue and backend related steps in orange.

Afterward, the user defines the input and output directory as well as the simulation parameters and the backend of choice in a configuration file. The input files and the configuration file TRANSFORMATO suggests a common core between the molecules. The user can decide to stick with the suggestion or define the common core manually. Then, the user must manually decide on the number of states needed to turn off the charges and define the order in which the Lennard-Jones potentials of the heavy atoms are turned off. After starting a TRANSFORMATO run, the input files for all the intermediate states are produced and an equilibrium simulation for each intermediate state is started. From the saved trajectories the thermodynamic samples are extracted and analyzed with MBAR. This process has to be done for both systems to obtain the relative free energy difference. The general workflow is illustrated in Figure 3.1.

3.1.1 Setup

MD simulations

The initial configurations for each state were created with TRANSFORMATO as just described. All simulations were run with the CHARMM simulation package.

The calculation for each state in vacuum used the following steps. The system was minimized using the steepest descent minimization method (SD) for 200 steps. Afterward, a simulation with a Langevin thermostat[67] was executed at 300 K for a simulation time of 2 ns and a time step of 1 fs with a total of 2,000,000 steps. Coordinates were saved every 100th step. A leap-frog integrator was used for all simulations. No cut-off was used. An example of an input file for a calculation in vacuum can be found in appendix B.3.

TABLE 3.1: Overview of the simulation settings for each state.

	vacuum	solvent
DOMDEC	✗	✔
minimization	SD	SD & ABNR
thermostat	Langevin	Langevin
integrator	leap-frog	leap-frog
SHAKE	✗	✔
PBC	✗	✔
cubic box	✗	30 Å
cut-on	✗	10 Å
cut-off	✗	12 Å
κ	✗	0.34 $Å^{-1}$
PME	✗	✔
switch	✗	VSWI & VFSW
Simulation Parameters		
simulation time	2 ns	2 ns
simulation steps	2,000,000	2,000,000
frames saved	20,000	20,000
time steps	1 fs	1 fs
temperature	300 K	300 K

The calculation for each state in a water box was calculated with the domain decomposition[68] (domdec) method using the following steps. The system was minimized using the steepest descent minimization method (SD) for 500 steps and an Adopted Basis Newton-Raphson method (ABNR) for 500 steps. Afterward, a NPT simulation with Langevin thermostat[34] was executed at 300 K for a simulation time of 2 ns and a time step of 1 fs with a total of 2,000,000 steps. Coordinates were saved every 100th step. The pressure was controlled by a Langevin piston barostat[67]. A leap-frog integrator was used for all simulations. Periodic boundary conditions were used with a box length of 30 Å as initial simulation box size. The Lennard-Jones interactions were switched off between 10 Å and 12 Å. `VSWItch`[17] and `VFSWitch`[7] were used as switching functions for the truncation of the Lennard-Jones interactions. Electrostatic interactions were calculated by Ewald summation[69] with $\kappa = 0.34 Å^{-1}$, employing the Particle Mesh Ewald (PME)[25, 26] on a $32 \times 32 \times 32$ grid and cubic spline interpolation of order 6. The TIP3[70, 71] water was kept rigid by SHAKE[72]. An example for an input file for a calculation in solvent can be found in appendix B.4.

An overview of the steps used as well as various settings for the calculations in vacuum and the water box are shown in Table 3.1.

Post-processing

The first 20 % of each trajectory was discarded to guarantee that all systems were in equilibrium. Afterward, the remainders of the trajectories were merged into a single trajectory and passed back to the `CHARMM` backend to evaluate the energetic information from each state. This information was passed to the Python implementation of the MBAR method, described in Section 2.5.1. `TRANSFORMATO` returns all the data

of interest in a compressed file format (pickle) for the free energy difference of the systems in a vacuum and in a solvent.

3.2 Absolute solvation free energies, PERT

The absolute solvation free energies for each of the compounds were computed with the PERT module of CHARMM.[15] The central step is turning off the non-bonded interactions between the solute and the surrounding waters. Since this removes all intramolecular non-bonded interactions in the solute as well, a gas phase correction was carried out, in which the non-bonded interactions of the solute were restored. System size, treatment of non-bonded interactions, thermostat, and barostat settings, etc. were analogous to the calculations based on `TRANSFORMATO`. Twenty-one λ states were used. At each step, 200,000 MD steps of equilibration, followed by 2,000,000 steps of production were simulated. Coordinates were saved every 100th step. Free energy differences between neighboring λ states were computed using BAR, the total free energy difference was the sum of these. Calculations in aqueous solution and the gas phase were repeated five times, starting from different random initial velocities. Error estimates were obtained by averaging each of the 20 pairwise free energy differences, the corresponding standard deviations were accumulated using Gaussian error propagation.

3.3 Common Core/SAI by hand

All free energy differences between the compounds and the respective common core were also computed using PSF and parameter files generated by an early version of `TRANSFORMATO`, which were then used to set up MD simulations "by hand".[15] All protocols and computational details are analogous to the automated `TRANSFORMATO` calculations described earlier, except for the following differences: For each state, 200,000 MD steps were followed by 2,000,000 production steps, during which 20,000 coordinate sets were saved to disk. Free energy differences were computed by "pairwise" BAR, as for the absolute solvation free energy differences. Similarly, statistical error estimates of the five repetitions of each series of simulations were computed as outlined there.

3.4 Systems studied

Figure 3.2 shows the nine systems evaluated. The relative solvation free energy was calculated for the transformation from methanol, ethane, 2-methylfuran, 2-methylindole, neopentane, and toluene to methane with methyl-X (CH_3X) as the common core. Also, the transformation from 2-cyclopentaylindole (2-CPI) to 7-cyclopentaylindole (7-CPI) with pentane-X ($C_5H_{11}X$) as common core was evaluated. These systems were proposed by Loeffler et al. as benchmark systems to compare free energy methods in different simulation packages.[10] An overview of all simulations and the common core used is given in Table 3.2.

In Table 3.3 the number of intermediate states used is shown. Here N_{atoms} refers to the number of heavy atoms outside of the common core. $n_{initial}$ is the initial state of the transformation. $n_{charges}$ refers to the number of intermediate states used to scale the charges of the atoms not part of the common core to zero. n_H refers to

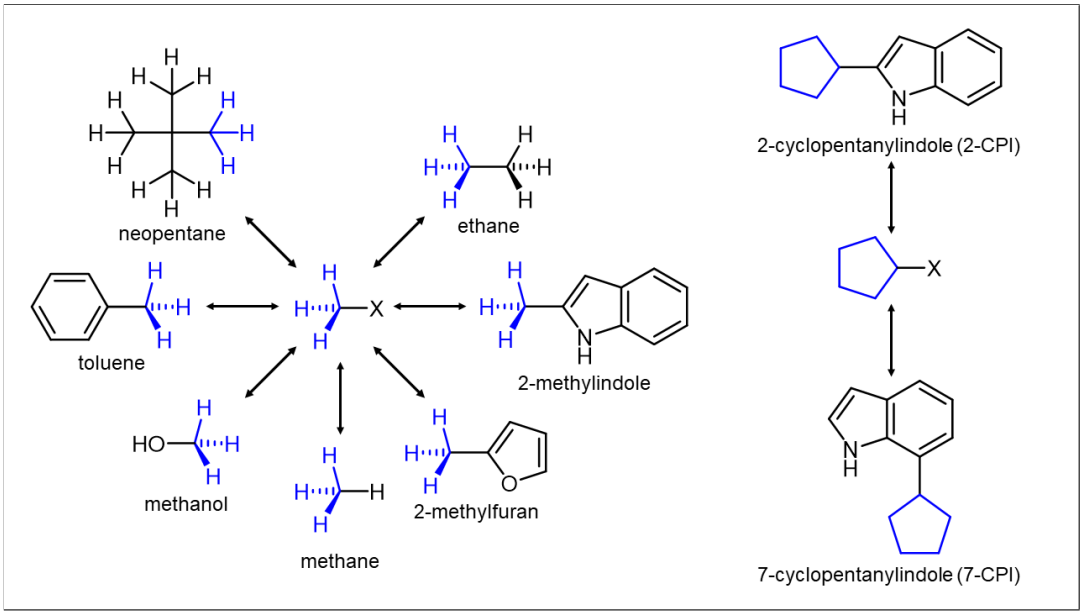

FIGURE 3.2: Systems evaluated with different free energy methods. The common core is highlighted in blue.

the number of intermediate states used to turn off the Lennard-Jones interaction of all hydrogens not part of the common core, and n_{LJ} is the number of intermediate states used to turn off the Lennard-Jones interactions of the heavy atoms not part of of the common core. n_X is the number of states needed to turn the atom connecting the common core to the chemical moiety, which has been transformed into dummy atoms, into a Lennard-Jones particle without charge. n_{CC} is the number of intermediate states used to scale the common core CC_A to the common core CC_B. Exceptions are methane and 7-CPI, because they have already the common core CC_B. n_{total} sums up the total number of intermediate states used for each system.

For the example of methanol (CH_3OH), the only heavy atom not part of the common core (CH_3X) is the oxygen ($N_{atoms} = 1$). Starting from the initial state ($n_{initial} = 1$) five intermediate states were needed to turn off the charges of all atoms outside of the common core ($n_{charges} = 5$). This was followed by a single intermediate state to turn off the Lennard-Jones interaction of the hydrogen ($n_H = 1$). The single hydrogen not part of the common core is now a dummy atom (CH_3OD). Since the oxygen is the atom connecting the common core to the moiety turned into dummy atoms, no intermediate state is needed to turn off the Lennard-Jones interaction ($n_{LJ} = 0$) and the oxygen is transformed into the special Lennard-Jones particle in one intermediate state ($n_X = 1$). Afterward, $CH_3X_{methanol}$ is scaled to $CH_3X_{methane}$ in four additional intermediate states ($n_{CC} = 4$). Therefore, the transformation of methanol CH_3OH) to the the common core ($CH_3X_{methane}$) was done with a total of twelve intermediate states ($n_{total} = 12$).

3.5 Handling of the results

3.5.1 Transformato

For each system of interest, five production runs were carried out. The results shown in Table 4.1 and 4.4 are mean values obtained from these five series of calculations.

TABLE 3.2: Overview of the simulations carried out and the common cores used.

Alchemical transformation			Abbrev.	common core
ethane	$\longrightarrow$	methane	ETH2MET	CH_3X
methanol	$\longrightarrow$	methane	MEOH2MET	CH_3X
neopentane	$\longrightarrow$	methane	NEO2MET	CH_3X
toluene	$\longrightarrow$	methane	TOL2MET	CH_3X
2-methylfuran	$\longrightarrow$	methane	MFU2MET	CH_3X
2-methylindole	$\longrightarrow$	methane	MIN2MET	CH_3X
2-CPI	$\longrightarrow$	7-CPI	CPI2CPI	$C_5H_{11}X$

TABLE 3.3: Overview of the number of intermediate states used to transform each system to the respective common core.

CC_A	${N_{atoms}}^a$	${n_{initial}}^b$	${n_{charges}}^c$	${n_H}^d$	${n_{LJ}}^e$	${n_X}^f$	${n_{CC}}^g$	${n_{total}}^h$
ethane	1	1	1	1	0	1	4	8
methanol	1	1	5	1	0	1	4	12
neopentane	4	1	5	1	3	1	5	16
toluene	6	1	1	1	5	1	4	13
2-methylfuran	5	1	1	1	4	1	4	12
2-methylindole	9	1	2	1	8	1	4	17
2-CPI	9	1	3	1	5	1	4	15
CC_B								
methane	0	1	1	0	0	1	0	3
7-CPI	9	1	3	1	5	1	0	11

[a]number of heavy atoms not part of the CC; [b]initial state used for the transformation; [c]number of intermediate states used to scale the charges of the atoms not part of the common core; [d]number of intermediate states used to turn off the Lennard-Jones interaction of all hydrogens not part of the common core; [e]number of intermediate states used to turn off the Lennard-Jones interactions of the heavy atoms not part of of the common core; [f]number of states needed to turn the atom connecting the common core to the chemical moiety, which has been transformed into dummy atoms, into a Lennard-Jones particle without charge; [g]number of intermediate states used to scale the common core CC_A to the common core CC_B; [h] total number of states used.

The mean was calculated as shown in Equation 3.1. Wherein n is the number of production runs, i the specific run, and x the evaluated value.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + ... + x_n}{n} \tag{3.1}$$

As an error estimate, the standard deviation σ was used . The standard deviation σ is calculated as the square root of the variance (Equation 3.3). The variance is calculated as the average squared deviation of all x_i from the mean $\overline{x}$ as shown in Equation 3.2.

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{3.2}$$

$$\sigma = \sqrt{\sigma^2} \tag{3.3}$$

3.5.2 Comparison of approaches

To validate the common core approach coupled with SAI, as well as the implementation into `TRANSFORMATO` requires the comparison to reference values. Because of the inadequacies of the force field, which will lead to systematic deviations between computed and experimental values, the direct comparison with experimental solvation free energies is of no use. Instead, as already outlined in the Introduction, a comparison to calculations of absolute solvation free energies made with the `PERT` module of `CHARMM`[7] using the PSSP soft-core model[7] was made. The solvation free energy differences derived from differences of absolute solvation free energies (`PERT`) and those calculated along the alchemical pathways should agree within statistical error bars when closing the thermodynamic cycle (cf. Section 2.5.2 and Figure 2.13 with the absolute path in black and the alchemical path in red). The variance along these two pathways can differ. Therefore, a Welch's t-test[73] was used to test the statistical significance of the cycle closure, since it is robust to unequal variances in contrast to the Student's t-test. The Welch t-statistic was calculated according to Equation 3.4, where $\overline{x}_i$ and $\overline{x}_j$ are the mean values of data set i and j. n_i and n_j are the number of points in the data sets. σ_i and σ_j are the standard deviations of the data sets. The data set j is the reference data set used, embodying the null hypothesis.[74] All values will be reported as two-tailed p-values and were calculated with SciPy[75].

$$t = \frac{\overline{x}_i - \overline{x}_j}{\sqrt{\frac{\sigma_i^2}{n_i} + \frac{\sigma_j^2}{n_j}}} \tag{3.4}$$

Chapter 4

Results and Discussion

4.1 Calculations with VSWItch

The validity of relative free energy calculations depends on the equivalence of the absolute and the alchemical path. To test the general approach (common core with SAI), the calculations were also set up "by hand". In Table 4.1 the relative solvation free energy differences for `CHARMM` calculations using CC/SAI set up manually ($\Delta\Delta G_{solv}^{hand}$)[1], the relative solvation free energy differences for `TRANSFORMATO` with the `CHARMM` backend ($\Delta\Delta G_{solv}^{TF}$) and the respective absolute solvation free energy differences ($\Delta\Delta G_{solv}^{abs}$)[2] are listed.[3] All results shown were calculated with the `VSWItch` switching function for the Lennard-Jones interactions. In Table 4.2 the difference $\Delta\Delta G_{solv}^{hand} - \Delta\Delta G_{solv}^{abs}$ and $\Delta\Delta G_{solv}^{TF} - \Delta\Delta G_{solv}^{abs}$ are shown with the respective statistical uncertainty. These differences should ideally be zero around the thermodynamic cycle to indicate a cycle closure as described in Section 2.5.2. The statistical significance of the cycle closure is given through the calculated p-values. A p-value ≥ 0.05 indicates that the null hypothesis holds true with 95 % confidence. Here it indicates that the cycle closure error is statistically indistinguishable from zero. For the calculations set up manually, the p-value is shown in Table 4.2. The alchemical pathway calculated set up manually and the absolute path calculated with `PERT` are in perfect agreement with the single exception of the transformation CPI2CPI. For CPI2CPI the cycle closure was not obtained within the statistical error (p-value = 0.02). The relative free energy deviates statistically significantly from the absolute path by 0.34 kcal/mol. CPI2CPI is the largest system investigated and the deviation in the results may indicate that more than five production runs are needed for this system.

The automated calculations with the `TRANSFORMATO` package using the `CHARMM` backend are also in excellent agreement with relative free energy differences along the alchemical path calculated in `TRANSFORMATO` and via absolute paths calculated with `PERT` (Table 4.3). All p-values are $\geq$ 0.05, indicating that the cycle closure error is statistically indistinguishable from zero. Here the transformation CPI2CPI has the lowest p-value (p-value = 0.07). The calculations along the alchemical path are also in excellent agreement with the calculations set up manually. All p-values are $\geq$ 0.05, indicating that the two calculations along the alchemical path are indistinguishable from each other. The lowest p-value (0.16) was obtained for the transformation ETH2MET.

[1] The values are part of the unpublished data calculated by Stefan Boresch[15]

[2] The values are part of the unpublished data calculated by Stefan Boresch[15]

[3] Raw results can be found at https://docs.google.com/spreadsheets/d/1EYYUOLXwSve1k8G56LWbJSbKqIxjJmCKvZVqmZga5o4/edit?usp=sharing

TABLE 4.1: Free energy difference of various systems in kcal/mol obtained from calculations with CHARMM. The VSWItch switching function was used in all calculations.

Transformation[a]	$\Delta\Delta G_{solv}^{hand\,b}$	$\Delta\Delta G_{solv}^{TF\ c}$	$\Delta\Delta G_{solv}^{abs\ d}$
ETH2MET	0.15 ± 0.05	0.07 ± 0.10	0.10 ± 0.02
MEOH2MET	7.06 ± 0.05	7.05 ± 0.05	7.05 ± 0.03
NEO2MET	-0.23 ± 0.12	-0.16 ± 0.13	-0.24 ± 0.02
TOL2MET	2.47 ± 0.08	2.41 ± 0.05	2.38 ± 0.03
MFU2MET	3.37 ± 0.13	3.31 ± 0.04	3.29 ± 0.07
MIN2MET	9.34 ± 0.12	9.36 ± 0.10	9.45 ± 0.05
CPI2CPI	-1.29 ± 0.22	-1.42 ± 0.19	-1.63 ± 0.07

[a]Abbrev. as in Table 3.2; [b]Relative solvation free energies using CC/SAI set up manually (alchemical path); [c]relative solvation free energies using CC/SAI set up with TRANSFORMATO (alchemical path); [d]absolute solvation free energies using PERT (absolute path).

TABLE 4.2: Comparison of relative solvation free energy differences from absolute and alchemical transformations of calculations using CC/SAI set up manually and calculations performed with PERT. The VSWItch switching function was used in all calculations. Energies are shown in kcal/mol.

Transformation[a]	$\Delta\Delta G_{solv}^{hand} - \Delta\Delta G_{solv}^{abs\ b}$	$\mathbf{p}^c$
ETH2MET	0.05 ± 0.05	0.09
MEOH2MET	0.01 ± 0.06	0.71
NEO2MET	0.01 ± 0.12	0.86
TOL2MET	0.09 ± 0.09	0.06
MFU2MET	0.08 ± 0.15	0.27
MIN2MET	-0.11 ± 0.13	0.11
CPI2CPI	0.34 ± 0.23	0.02

[a]Abbrev. as in Table 3.2; [b]see Table 4.1; [c]p-value obtained using Welch's t-test (cf. Section 3.5.2).

The results of the calculations set up manually fulfill the requirement of cycle closure, validating the Common Core approach in general.[15] The results of the newly implemented CHARMM backend also fulfill the requirement of cycle closure. This indicates that the implementation of a CHARMM backend into the TRANSFORMATO package was successful for the calculations of solvation free energy calculations.

In Figure 4.1 the results for the calculations with VSWItch calculated with PERT (orange), with CHARMM using SAI/CC set up manually (green) and TRANSFORMATO with CHARMM backend (blue) are illustrated. All results were scaled with respect to the results obtained with TRANSFORMATO. The red border indicates the maximal spread between the results reported by Loeffler et al. for the same systems.[10] The average of the calculations with VSWItch calculated with PERT, CHARMM (manually) and TRANSFORMATO with CHARMM are all within the range of the maximal spread observed by Loeffler et al.[10]

TABLE 4.3: Comparison of relative solvation free energy differences from absolute and alchemical transformations using `TRANSFORMATO` with `CHARMM` backend and calculations performed with `PERT` (left). Comparison of relative solvation free energy differences from two alchemical transformations using `TRANSFORMATO` with `CHARMM` backend and calculations set up manually using CC/SAI (right). The `VSWItch` switching function was used in all calculations. Energies are shown in kcal/mol.

Transformation[a]	$\Delta\Delta G^{TF}_{solv} - \Delta\Delta G^{abs\ b}_{solv}$	**p**[c]	$\Delta\Delta G^{TF}_{solv} - \Delta\Delta G^{hand\ b}_{solv}$	**p**[c]
ETH2MET	-0.03 ± 0.10	0.54	-0.08 ± 0.11	0.16
MEOH2MET	0.00 ± 0.06	1.00	-0.01 ± 0.07	0.76
NEO2MET	0.08 ± 0.13	0.24	0.07 ± 0.17	0.40
TOL2MET	0.03 ± 0.06	0.29	-0.06 ± 0.09	0.20
MFU2MET	0.02 ± 0.08	0.60	-0.06 ± 0.14	0.37
MIN2MET	-0.09 ± 0.11	0.12	0.02 ± 0.16	0.78
CPI2CPI	0.21 ± 0.20	0.07	-0.13 ± 0.29	0.35

[a]Abbrev. as in Table 3.2; [b]see Table 4.1; [c]p-value obtained using Welch's t-test (cf. Section 3.5.2).

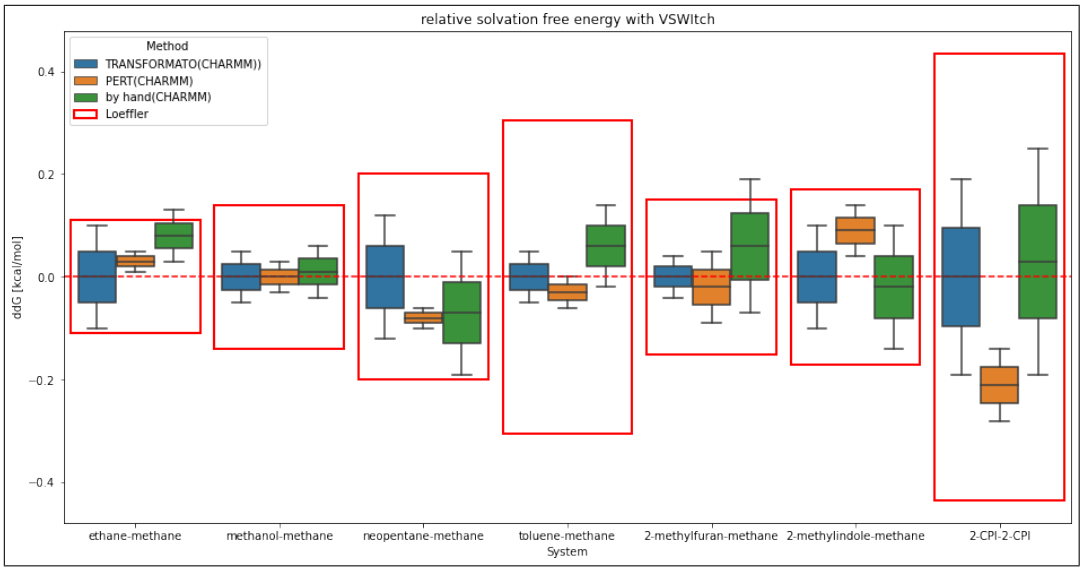

FIGURE 4.1: Results for calculations with `PERT` (orange), `CHARMM` set up manually (green) and `TRANSFORMATO` with `CHARMM` backend (blue). All calculations were performed with the `VSWItch` switching function for the Lennard-Jones interactions. The red box indicates the maximum spread reported by Loeffler et al. for the corresponding transformations.[10]

4.2 Calculations with VFSWitch

Since the CHARMM backend of TRANSFORMATO was successfully validated (Section 4.1), it was then used to validate the results obtained with openMM. In Table 4.4 the relative solvation free energy differences for CHARMM calculations with VFSWitch done with TRANSFORMATO ($\Delta\Delta G_{solv}^{CMM}$) and the relative solvation free energy differences for Tranformato with openMM backend and VFSWitch ($\Delta\Delta G_{solv}^{OMM}$)[4] are listed, together with the difference $\Delta\Delta G_{solv}^{OMM} - \Delta\Delta G_{solv}^{CMM}$ and the respective p-values. Here the difference between the two alchemical paths was evaluated. The results indicate that the relative free energies along the alchemical path are statistically indistinguishable from each other, with the single exception of MEOH2MET (p-value = 0.03).

In Figure 4.2 the results for the calculations with TRANSFORMATO with openMM backend (orange) and TRANSFORMATO with CHARMM backend (blue) are illustrated. All results were scaled to the difference between the averages of the simulations with openMM backend and CHARMM backend. The red border indicates the maximal spread between results reported by Loeffler et al. for the same systems.[10] The average of the results from the calculations with the VFSWitch switching function in TRANSFORMATO with openMM backend and TRANSFORMATO with CHARMM backend are all within the range of the maximal spread observed by Loeffler et al.[10]

Due to the statistical indistinguishability of the free energy differences calculated with the CHARMM backend and the openMM backend, the openMM backend was successful validated as well. Even though for one system the results between CHARMM and openMM do not agree within statistical error bars, the results nevertheless validate the equivalence of using CHARMM and openMM as backends.

In general, the comparison of results obtained with different simulation packages is quite difficult and the results have statistically significant deviations.[10] In this special case validation is possible since CHARMM and openMM are compatible and CHARMM-GUI provides setup-files for both programs. TRANSFORMATO enables us to use the same "high-level" input[5] to generate the files needed for the free energy calculations with CHARMM and openMM backend.

[4] The values are part of the unpublished data calculated by Marcus Wieder[15]

[5] High-level input refers to the files obtained by CHARMM-GUI and the user created configuration file.

TABLE 4.4: Comparison of relative solvation free energy differences from alchemical transformations using TRANSFORMATO with CHARMM and openMM backend. The VFSWitch switching function was used in all calculations. Energies are shown in kcal/mol.

Transformation[a]	$\Delta\Delta G_{solv}^{omm}$ [b]	$\Delta\Delta G_{solv}^{cmm}$ [c]	$\Delta\Delta G_{solv}^{omm} - \Delta\Delta G_{solv}^{cmm}$	**p**[d]
ETH2MET	0.09 ± 0.09	-0.01 ± 0.09	0.10 ± 0.13	0.12
MEOH2MET	7.12 ± 0.11	6.92 ± 0.04	0.20 ± 0.12	0.03
NEO2MET	-0.38 ± 0.11	-0.45 ± 0.12	0.07 ± 0.16	0.36
TOL2MET	2.21 ± 0.08	2.15 ± 0.06	0.06 ± 0.10	0.22
MFU2MET	3.12 ± 0.13	3.10 ± 0.14	0.02 ± 0.19	0.82
MIN2MET	8.97 ± 0.08	8.93 ± 0.11	0.04 ± 0.14	0.53
CPI2CPI	-1.13 ± 0.25	-1.66 ± 0.64	0.53 ± 0.69	0.14

[a]Abbrev. as in Table 3.2; [b]Relative solvation free energies using CC/SAI set up with TRANSFORMATO with openMM backend (alchemical path); [c]relative solvation free energies using CC/SAI set up with TRANSFORMATO with CHARMM backend (alchemical path); [d]p-value obtained using Welch's t-test (cf. Section 3.5.2).

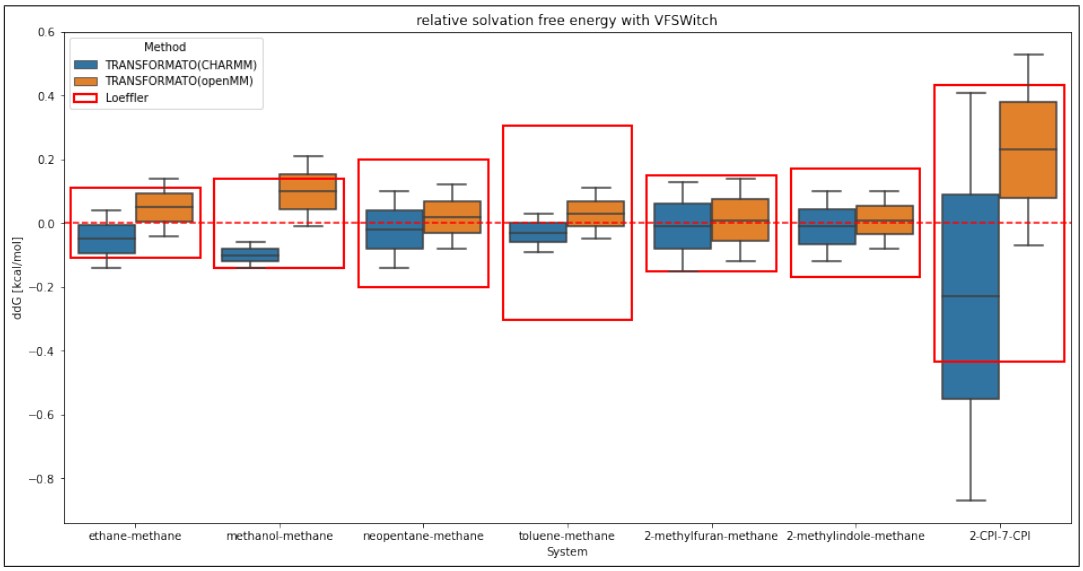

FIGURE 4.2: Results for calculations with TRANSFORMATO with openMM backend (orange) and TRANSFORMATO with CHARMM backend (blue). All calculations were performed with the VFSWitch switching function for the Lennard-Jones interactions. The red box indicates the maximum spread reported by Loeffler et al. for the corresponding transformations.[10]

Chapter 5

Conclusion

Solvation free energy calculations were performed for nine systems, transforming each to a suitable common core. The results were used to calculate seven relative solvation free energy differences which were also reported by Loeffler et al. All simulations were carried out with TRANSFORMATO using the newly implemented CHARMM backend. These calculations employed two different switching functions for the truncation of the Lennard-Jones interactions. The results validate the successful implementation of the CHARMM backend by showing cycle closure with respect to calculations made along the absolute path using PERT. The cycle closure error is statistically indistinguishable from zero and, therefore, the implementation can be considered correct.

The CHARMM backend was then further used to validate the results obtained with the openMM backend by showing that the relative solvation free energies are statistically indistinguishable from each other along the alchemical path. The results indicate that the common core approach coupled with the SAI approach is successfully implemented in TRANSFORMATO for the CHARMM and openMM backends. Further, it is possible to perform alchemical transformations to calculate solvation free energies without statistically significant deviations between the two backends starting from the same input files.

Appendix A

Translation of Abstract and Conclusion

A.1 Abstract (German)

In den letzten Jahren haben sich Berechnungen der freien Energie zu einer beliebten Methode im akademischen Bereich sowie in der pharmazeutischen Industrie entwickelt, da es die Möglichkeit bietet die thermodynamischen Eigenschaften eines Systems zu berechnen. Bei Berechnungen der freien Energie ist es möglich, die Differenz in der freien Energie entlang eines nicht-physikalischen Pfades zu berechnen, indem kleine Teile des Systems transformiert werden, z.B. eine funktionelle Gruppe in eine andere umgewandelt wird. Diese Transformation wird über eine Reihe von Intermediaten durchgeführt. Ein neuer Ansatz berechnet einen Common-Core zwischen dem Anfangs- und dem Endzustand ein und berechnet die Differenz der freien Energie zwischen dem Anfangszustand und dem Common-Core, sowie zwischen dem Endzustand und dem Common-Core, um die Differenz der freien Energie zwischen Anfangs- und Endzustand zu erhalten. Das Programmpaket TRANSFORMATO ist eine Implementierung dieses Ansatzes, welche automatisch die verschiedenen benötigten Zwischenzustände aufsetzt. In dieser Arbeit wurde ein CHARMM-Backend erfolgreich in das TRANSFORMATO-Paket implementiert. Diese Implementierung wurde verwendet, um zu zeigen, dass TRANSFORMATO den Common-Core Ansatz sowohl für CHARMM als auch für das openMM-Backend erfolgreich umsetzt, indem Berechnungen, die mit TRANSFORMATO durchgeführt wurden, gegen Referenzberechnungen validiert wurden, die mit dem PERT-Modul von CHARMM durchgeführt wurden.

A.2 Conclusion (German)

Die Berechnungen der freien Solvatationsenergie wurden für neun Systeme durchgeführt, wobei jedes System zu einen geeigneten Common-Core transformiert wurde. Die Ergebnisse wurden zur Berechnung von sieben relativen Differenzen der freien Solvatationsenergie verwendet. Für selbige Systeme wurden auch von Loeffler et al. relativen Differenzen der freien Solvatationsenergie veröffentlicht. Alle Simulationen wurden mit TRANSFORMATO unter Verwendung des neu implementierten CHARMM-Backends durchgeführt. Bei diesen Berechnungen wurden zwei verschiedene Switching-Funktionen für die Trunkierung der Lennard-Jones-Wechselwirkungen verwendet. Die Ergebnisse bestätigen die erfolgreiche Implementierung des CHARMM-Backends, indem sie die Schließung des thermodynamischen Zykluses in Bezug auf die Berechnungen entlang des absoluten Pfades unter Verwendung von PERT zeigen. Der Fehler der Zyklusschließung ist statistisch nicht von Null zu unterscheiden, so

dass die Implementierung als korrekt angesehen werden kann.

Das CHARMM-Backend wurde dann weiter verwendet, um die mit dem openMM-Backend erhaltenen Ergebnisse zu validieren, indem gezeigt wurde, dass die relativen freien Solvatationsenergien entlang des alchemistischen Pfades statistisch nicht voneinander zu unterscheiden sind. Die Ergebnisse zeigen, dass der Common-Core Ansatz gekoppelt mit dem SAI Ansatz erfolgreich in TRANSFORMATO für die Backends CHARMM und openMM implementiert sind. Außerdem ist es möglich, alchemistische Transformationen zur Berechnung der freien Solvatationsenergien ohne statistisch signifikante Abweichungen zwischen den beiden Backends ausgehend von denselben Eingabedateien durchzuführen.

A.3 Abstract (English)

In recent years free energy calculations have become a popular method in academia and the pharmaceutical industry, due to the possibility to calculate the thermodynamic properties of a system. In free energy calculations, it is possible to calculate free energy differences along non-physical pathways by transforming small parts of the system, i.e., changing a functional group into another. This transformation is done over a series of intermediate states. A new approach inserts a common core state between the initial and the final state and calculates the free energy difference between the initial state and the common core, as well as between the final state and the common core to obtain the free energy difference between initial and final state. The TRANSFORMATO package is an implementation of this approach, automatically setting up the various intermediate states needed. In this thesis, a CHARMM backend was successfully implemented into the TRANSFORMATO package. This implementation was used to show that TRANSFORMATO successfully implemented the common core approach for both CHARMM and openMM backend by validating calculations done with TRANSFORMATO against reference calculations done with the established PERT module of CHARMM.

A.4 Conclusion (English)

Solvation free energy calculations were performed for nine systems, transforming each to a suitable common core. The results were used to calculate seven relative solvation free energy differences which were also reported by Loeffler et al. All simulations were carried out with TRANSFORMATO using the newly implemented CHARMM backend. These calculations employed two different switching functions for the truncation of the Lennard-Jones interactions. The results validate the successful implementation of the CHARMM backend by showing cycle closure with respect to calculations made along the absolute path using PERT. The cycle closure error is statistically indistinguishable from zero and, therefore, the implementation can be considered correct.

The CHARMM backend was then further used to validate the results obtained with the openMM backend by showing that the relative solvation free energies are statistically indistinguishable from each other along the alchemical path. The results indicate that the common core approach coupled with the SAI approach is successfully implemented in TRANSFORMATO for the CHARMM and openMM backends. Further, it is possible to perform alchemical transformations to calculate solvation free energies

without statistically significant deviations between the two backends starting from the same input files.

Appendix B

Input files

After setting up the systems of interest in CHARMM-GUI[7, 65, 66], it is necessary to define a config file. In this file the structure name, the solvent system, the protein structure file (.psf), the coordinate file (.crd), a restart file form an equilibration run by CHARMM-GUI[7, 65, 66], a simulation parameter file, and the name of the files created for the intermediate states are defined for each of the molecules. In a second section, the parameters for the MD simulation are set. An example for such a **config.yaml** is provided in the appendix B.1. Since CHARMM-GUI[7, 65, 66] names most of the mentioned files the same for every molecule, it is only necessary to change the solvent system TLC, the name of the structures, and the simulation parameters (Table 3.1).

Afterward the intermediate states are created by modifying the code B.2 with the following steps:

- Parsing the paths to the config.yaml, the molecule directory and the output directory into the load_config_yaml() function (appendix B.2 line 1)
- Defining the amount of intermediate states needed to scale charges to zero (appendix B.2 line 26)
- Defining the order in which the Lennard Jones interactions are turned of for the heavy atoms (appendix B.2 line 50-56)
- Defining the amount of intermediate states needed to transform cc1 to cc2 (appendix B.2 line 77)

The intermediate states are then created into the output directory with all the needed files to start an MD simulation for each of the intermediate states.

B.1 Exemplary Config file for Transformato

```
1  ---
2  ################
3  system:
4  ################
5      structure1:
6          name: "toluene"
7          tlc: "UNL"
8          vacuum:
9              dirname: "vacuum"
10             psf_file_name: "step3_charmm2omm"
```

```
            crd_file_name: "step3_charmm2omm"
            rst_file_name: "step4_equilibration"
            simulation_parameter: "step5_production.inp"
            intermediate-filename: "lig_in_vacuum"
        waterbox:
            dirname: "waterbox"
            psf_file_name: "step3_charmm2omm"
            crd_file_name: "step3_charmm2omm"
            rst_file_name: "step4_equilibration"
            simulation_parameter: "step5_production.inp"
            intermediate-filename: "lig_in_waterbox"

    structure2:
        name: "methane"
        tlc: "LIG"
        vacuum:
            dirname: "vacuum"
            psf_file_name: "step3_charmm2omm"
            crd_file_name: "step3_charmm2omm"
            rst_file_name: "step4_equilibration"
            simulation_parameter: "step5_production.inp"
            intermediate-filename: "lig_in_vacuum"
        waterbox:
            dirname: "waterbox"
            psf_file_name: "step3_charmm2omm"
            crd_file_name: "step3_charmm2omm"
            rst_file_name: "step4_equilibration"
            simulation_parameter: "step5_production.inp"
            intermediate-filename: "lig_in_waterbox"

################
simulation:
################
    parameters:
        nstep: 2000000
        nstdcd: 100
        nstout: 5000
        dt: 0.001
        switch: "vswitch"
    GPU: True
    free-energy-type: "solvation-free-energy"
################
```

B.2 Exemplary script to create the intermediate files

```
configuration = load_config_yaml(
    config="conf.yaml", input_dir="input_dir", output_dir="output_dir"
)

```

```
s1 = SystemStructure(configuration, "structure1")
s2 = SystemStructure(configuration, "structure2")

s1_to_s2 = ProposeMutationRoute(s1, s2)
s1_to_s2.calculate_common_core() #Displays the proposed commen core

mutation_list = s1_to_s2.generate_mutations_to_common_core_for_mol2()

i = IntermediateStateFactory(
    system=s1,
    configuration=configuration,
)
# write out endpoint
output_files = []
intst = 1
output_file_base = i.write_state(mutation_conf=[], intst_nr=intst)
output_files.append(output_file_base)

# start with charges
# turn off charges
charges = mutation_list["charge"]
for lambda_value in np.linspace(1, 0, 4)[1:]:
    print(lambda_value)
    intst += 1
    output_file_base = i.write_state(
        mutation_conf=charges,
        lambda_value_electrostatic=lambda_value,
        intst_nr=intst,
    )
    output_files.append(output_file_base)

# Turn off hydrogens
intst += 1
hydrogen_lj_mutations = mutation_list["hydrogen-lj"]
output_file_base = i.write_state(
    mutation_conf=hydrogen_lj_mutations,
    lambda_value_vdw=0.0,
    intst_nr=intst,
)
output_files.append(output_file_base)

# turn off heavy atoms
d=transformato.utils.map_lj_mutations_to_atom_idx(mutation_list["lj"])
print(d)
# turn off heavy atoms
for m in [
    [d[(13,)]],             # numbers have to be switched
    [d[(11,)], d[(8,)]],    # with the heavy atoms outside the CMS
    [d[(0,)]],
    [d[(2,)], d[(10,)]],
    [d[(4,)], d[(7,)]],
```

```
]:
    intst += 1

    output_file_base = i.write_state(
        mutation_conf=m,
        lambda_value_vdw=0.0,
        intst_nr=intst,
    )
    output_files.append(output_file_base)

# generate terminal lj
intst += 1

output_file_base = i.write_state(
    mutation_conf=mutation_list["terminal-lj"],
    lambda_value_vdw=0.0,
    intst_nr=intst,
)
output_files.append(output_file_base)

m = mutation_list["transform"]
for lambda_value in np.linspace(0.75, 0, 4):
    intst += 1
    print(lambda_value)
    # turn off charges
    output_file_base = i.write_state(
        mutation_conf=m,
        common_core_transformation=lambda_value,
        intst_nr=intst,
    )
    output_files.append(output_file_base)
```

B.3 CHARMM vacuum input script

```
*Version September 2020
*Run script for CHARMM jobs from transformato
*

! Read topology and parameter files
stream charmm_toppar.str

! Read PSF
open read unit 10 card name lig_in_vacuum.psf
read psf  unit 10 card

! Read Coordinate
open read unit 10 card name lig_in_vacuum.crd
read coor unit 10 card

```

```
coor orie sele all end ! put the molecule at the origin

MMFP
GEO rcm sphere -
    Xref 0.0 Yref 0.0 Zref 0.0 XDIR 1.0 YDIR 1.0 ZDIR 1.0 -
    harmonic FORCE 1.0 select .not. ( hydrogen .or. resname TIP3 ) -
    end
END

set ctofnb 990.
set ctonnb 980.
set cutnb  1000.

nbonds ctonnb @ctonnb ctofnb @ctofnb cutnb @cutnb -
  atom swit vatom vswitch -
  inbfrq 1

energy   inbfrq 1
energy   inbfrq 0

mini sd nstep 200

set nstep = 2000000
set temp = 300.0

scalar fbeta set 5. sele all end
open write unit 21 file name lig_in_vacuum.dcd

DYNA lang leap start time 0.001 nstep @nstep -
    nprint 5000 iprfrq 5000 -
    iunread -1 iunwri -1 iuncrd 21 iunvel -1 kunit -1 -
    nsavc 100 nsavv 0 -
    rbuf 0. tbath @temp ilbfrq 0  firstt @temp -

stop
```

B.4 CHARMM waterbox input script

```
*Version September 2020
*Run script for CHARMM jobs from transformato
*

! Read topology and parameter files
stream charmm_toppar.str

! Read PSF
open read unit 10 card name lig_in_waterbox.psf
read psf  unit 10 card

```

```
! Read Coordinate
open read unit 10 card name lig_in_waterbox.crd
read coor unit 10 card

!
! Setup PBC (Periodic Boundary Condition)
!

stream charmm_step3_pbcsetup.str

!
! Image Setup
!

open read unit 10 card name charmm_crystal_image.str
CRYSTAL DEFINE @XTLtype @A @B @C @alpha @beta @gamma
CRYSTAL READ UNIT 10 CARD

!Image centering by residue
IMAGE BYRESID XCEN @xcen YCEN @ycen ZCEN @zcen sele resname TIP3 end

!
! Nonbonded Options
!

nbonds atom vatom VSWItch bycb -
       ctonnb 10.0 ctofnb 12.0 cutnb 16.0 cutim 16.0 -
       inbfrq -1 imgfrq -1 wmin 1.0 cdie eps 1.0 -
       ewald pmew fftx @fftx ffty @ffty fftz @fftz -
       kappa .34 spline order 6

energy

!
!use a restraint to place center of mass
!of the molecules near the origin

MMFP
GEO rcm sphere -
    Xref @xcen Yref @ycen Zref @zcen XDIR 1.0 YDIR 1.0 ZDIR 1.0 -
    harmonic FORCE 1.0 select .not. ( hydrogen .or. resname TIP3 ) -
    end
END

shak bonh para fast sele segi SOLV end

mini SD nstep 500
mini ABNR nstep 500

!
! NPT dynamics:
```

```
! you can change
! nstep  : number of MD steps
! nprint : print-out frequency
! nsavc  : the trajectory saving frequency
!

! estimate Pmass from SYSmass (total system mass)
! there could be problems with exreme values,
! such as  Pmass << SYSmass or Pmass >> SYSmass
scalar mass stat
calc Pmass = int ( ?stot  /  50.0 )

energy
domdec gpu only
energy

set nstep = 2000000
set temp = 300.0

scalar fbeta set 5. sele all end
open write unit 13 file name lig_in_waterbox.dcd

DYNA CPT leap start time 0.001 nstep @nstep -
     nprint 5000 iprfrq 5000 -
     iunread -1 iunwri -1 iuncrd 13 iunvel -1 kunit -1 -
     nsavc 100 nsavv 0 -
     PCONSTANT pref   1.0  pmass @Pmass  pgamma   20.0 -
     lang rbuf 0. tbath @temp ilbfrq 0  firstt @temp -
     ECHECK 0

stop
```

B.5 Energy evaluation in vacuum input script

```
*Version September 2020
*Run script for CHARMM jobs from transformato
*

! Read topology and parameter files
stream charmm_toppar.str

! Read PSF
open read unit 10 card name @{top}.psf
read psf  unit 10 card

! Read Coordinate
open read unit 10 card name @{top}.crd
read coor unit 10 card

```

```
coor orie sele all end ! put the molecule at the origin

set ctofnb 990.
set ctonnb 980.
set cutnb  1000.

nbonds ctonnb @ctonnb ctofnb @ctofnb cutnb @cutnb -
  atom swit vatom vswitch -
  inbfrq 1

energy

energy inbfrq 0

scalar fbeta set 5. sele all end

open read file unit 41 name ../traj.dcd
traj query unit 41

set start ?start
set nframes ?nfile
set skip ?skip

set nframes @nframes !?nfile
traj firstu 41 nunit 1 begi @start skip @skip stop @nframes

open form writ unit 51 name ener_vac.log
echu 51
set idx 0
label loop
traj read
energy
echo ?ener
incr idx by 1
if @idx .lt. @nframes goto loop


stop
```

B.6 Energy evaluation in waterbox input script

```
*Version September 2020
*Run script for CHARMM jobs from transformato
*

! Read topology and parameter files
stream charmm_toppar.str

! Read PSF
```

```
open read unit 10 card name @{top}.psf
read psf  unit 10 card

! Read Coordinate
open read unit 10 card name @{top}.crd
read coor unit 10 card

stream charmm_step3_pbcsetup.str

!
! Image Setup
!

open read unit 10 card name charmm_crystal_image.str
CRYSTAL DEFINE @XTLtype @A @B @C @alpha @beta @gamma
CRYSTAL READ UNIT 10 CARD

!Image centering by residue
IMAGE BYRESID XCEN @xcen YCEN @ycen ZCEN @zcen sele resname TIP3 end

!
! Nonbonded Options
!
cons fix sele segi solv end

nbonds atom vatom VSWItch bycb -
       ctonnb 10.0 ctofnb 12.0 cutnb 12.0 cutim 12.0 -
       inbfrq 1 imgfrq 1 wmin 1.0 cdie eps 1.0 -
       ewald pmew fftx @fftx ffty @ffty fftz @fftz -
       kappa .34 spline order 6

energy

open read file unit 41 name ../traj.dcd
traj query unit 41

set start ?start
set nframes ?nfile
set skip ?skip

set nframes @nframes !?nfile
traj firstu 41 nunit 1 begi @start skip @skip stop @nframes

open form writ unit 51 name ener_solv.log
echu 51
set idx 0
label loop
traj read
energy
echo ?ener
incr idx by 1
```

```
60 if @idx .lt. @nframes goto loop
61
62 stop
```

Bibliography

[1] https://scifinder.cas.org/, Keyword: molecular simulations, 03.01.2021.

[2] S. Hug, *Classical Molecular Dynamics in a Nutshell*, (Eds.: L. Monticelli, E. Salonen), Springer-Verlag GmbH, Berlin, **2012**, pp. 127–152, 702 pp.

[3] Z. Cournia, B. Allen, W. Sherman, *Journal of Chemical Information and Modeling* **2017**, *57*, 2911–2937.

[4] R. Abel, L. Wang, D. L. Mobley, R. A. Friesner, *Current Topics in Medicinal Chemistry* **2017**, *17*, 2577–2585.

[5] S. Wan, G. Tresadern, L. Pérez-Benito, H. Vlijmen, P. V. Coveney, *Advanced Theory and Simulations* **2019**, *3*, 1900195.

[6] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, R. J. Woods, *Journal of Computational Chemistry* **2005**, *26*, 1668–1688.

[7] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, M. Karplus, *Journal of Computational Chemistry* **2009**, *30*, 1545–1614.

[8] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, E. Lindahl, *SoftwareX* **2015**, *1-2*, 19–25.

[9] P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M. R. Shirts, V. S. Pande, *Journal of Chemical Theory and Computation* **2012**, *9*, 461–469.

[10] H. H. Loeffler, S. Bosisio, G. D. R. Matos, D. Suh, B. Roux, D. L. Mobley, J. Michel, *Journal of Chemical Theory and Computation* **2018**, *14*, 5567–5582.

[11] M. R. Shirts, D. L. Mobley, *An Introduction to Best Practices in Free Energy Calculations*, (Eds.: L. Monticelli, E. Salonen), Springer-Verlag GmbH, Berlin, **2012**, pp. 271–311, 702 pp.

[12] H. H. Loeffler, J. Michel, C. Woods, *Journal of Chemical Information and Modeling* **2015**, *55*, 2485–2490.

[13] L. Wang, Y. Wu, Y. Deng, B. Kim, L. Pierce, G. Krilov, D. Lupyan, S. Robinson, M. K. Dahlgren, J. Greenwood, D. L. Romero, C. Masse, J. L. Knight, T. Steinbrecher, T. Beuming, W. Damm, E. Harder, W. Sherman, M. Brewer, R. Wester, M. Murcko, L. Frye, R. Farid, T. Lin, D. L. Mobley, W. L. Jorgensen, B. J. Berne, R. A. Friesner, R. Abel, *Journal of the American Chemical Society* **2015**, *137*, 2695–2703.

[14] M. Wieder, https://github.com/wiederm/transformato.

[15] M. Wieder, B. Braunsfeld, S. Boresch, *unpublished*, manuscript in preparation.

[16] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, V. S. Pande, *PLOS Computational Biology* **2017**, *13*, 1005659.

[17] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, M. Karplus, *Journal of Computational Chemistry* **1983**, *4*, 187–217.

[18] B. S. Daan Frenkel, *Understanding Molecular Simulation*, Elsevier LTD, Oxford, **2001**, 664 pp.

[19] A. Leach, *Molecular Modelling*, Pearson Education (US), **2001**, 784 pp.

[20] L. Verlet, *Physical Review* **1967**, *159*, 98–103.

[21] W. C. Swope, H. C. Andersen, P. H. Berens, K. R. Wilson, *The Journal of Chemical Physics* **1982**, *76*, 637–649.

[22] R. W. Hockney, The Potential Calculation and Some Applications, Methods In Comp vol. 9, **1970**.

[23] W. Pauli, *Zeitschrift für Physik* **1925**, *31*, 765–783.

[24] D. M. York, T. A. Darden, L. G. Pedersen, *The Journal of Chemical Physics* **1993**, *99*, 8345–8348.

[25] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *The Journal of Chemical Physics* **1995**, *103*, 8577–8593.

[26] T. Darden, D. York, L. Pedersen, *The Journal of Chemical Physics* **1993**, *98*, 10089–10092.

[27] R. Hentschke, *Statistische Mechanik : eine Einführung für Physiker, Chemiker und Materialwissenschaftler ; [mit zahlreichen Beispielaufgaben in MATHEMATICA*, Wiley-VCH, Weinheim, **2004**, 348 pp.

[28] C. P. Lowe, *Europhysics Letters (EPL)* **1999**, *47*, 145–151.

[29] S. Nosé, *Molecular Physics* **1984**, *52*, 255–268.

[30] W. G. Hoover, *Physical Review A* **1985**, *31*, 1695–1697.

[31] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, J. R. Haak, *The Journal of Chemical Physics* **1984**, *81*, 3684–3690.

[32] T. Schneider, E. Stoll, *Physical Review B* **1978**, *17*, 1302–1322.

[33] H. C. Andersen, *The Journal of Chemical Physics* **1980**, *72*, 2384–2393.

[34] D. S. Lemons, A. Gythiel, *American Journal of Physics* **1997**, *65*, 1079–1081.

[35] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, *The Journal of Chemical Physics* **1953**, *21*, 1087–1092.

[36] R. W. Zwanzig, *The Journal of Chemical Physics* **1954**, *22*, 1420–1426.

[37] D. Wu, D. A. Kofke, *The Journal of Chemical Physics* **2005**, *123*, 054103.

[38] A. Pohorille, C. Jarzynski, C. Chipot, *The Journal of Physical Chemistry B* **2010**, *114*, 10235–10253.

[39] M. R. Shirts, V. S. Pande, *The Journal of Chemical Physics* **2005**, *122*, 144107.

[40] N. Lu, J. K. Singh, D. A. Kofke, *The Journal of Chemical Physics* **2003**, *118*, 2977–2984.

[41] J. G. Kirkwood, *The Journal of Chemical Physics* **1935**, *3*, 300–313.

[42] S. Bruckner, S. Boresch, *Journal of Computational Chemistry* **2010**, *32*, 1303–1319.

[43] H. Resat, M. Mezei, *The Journal of Chemical Physics* **1993**, *99*, 6052–6061.

[44] M. Jorge, N. M. Garrido, A. J. Queimada, I. G. Economou, E. A. Macedo, *Journal of Chemical Theory and Computation* **2010**, *6*, 1018–1027.

[45] C. Shyu, F. M. Ytreberg, *Journal of Computational Chemistry* **2009**, 2297–2304.

[46] C. H. Bennett, *Journal of Computational Physics* **1976**, *22*, 245–268.

[47] G. E. Crooks, *Physical Review E* **2000**, *61*, 2361–2366.

[48] M. R. Shirts, E. Bair, G. Hooker, V. S. Pande, *Physical Review Letters* **2003**, *91*, 140601.

[49] S. W. Rick, *Journal of Chemical Theory and Computation* **2006**, *2*, 939–946.

[50] F. M. Ytreberg, R. H. Swendsen, D. M. Zuckerman, *The Journal of Chemical Physics* **2006**, *125*, 184114.

[51] M. R. Shirts, J. D. Chodera, *The Journal of Chemical Physics* **2008**, *129*, 124105.

[52] E. C. Dybeck, G. König, B. R. Brooks, M. R. Shirts, *Journal of Chemical Theory and Computation* **2016**, *12*, 1466–1480.

[53] A. M. Ferrenberg, R. H. Swendsen, *Physical Review Letters* **1989**, *63*, 1195–1198.

[54] S. Kumar, J. M. Rosenberg, D. Bouzida, R. H. Swendsen, P. A. Kollman, *Journal of Computational Chemistry* **1992**, *13*, 1011–1021.

[55] M. Souaille, B. Roux, *Computer Physics Communications* **2001**, *135*, 40–57.

[56] J. P. M. Postma, H. J. C. Berendsen, J. R. Haak, *Faraday Symposia of the Chemical Society* **1982**, *17*, 55.

[57] T. Simonson, *Molecular Physics* **1993**, *80*, 441–447.

[58] M. Zacharias, T. P. Straatsma, J. A. McCammon, *The Journal of Chemical Physics* **1994**, *100*, 9025–9031.

[59] T. C. Beutler, A. E. Mark, R. C. van Schaik, P. R. Gerber, W. F. van Gunsteren, *Chemical Physics Letters* **1994**, *222*, 529–539.

[60] S. Boresch, S. Bruckner, *Journal of Computational Chemistry* **2011**, *32*, 2449–2458.

[61] Y. Cao, T. Jiang, T. Girke, *Bioinformatics* **2008**, *24*, 366–374.

[62] S. O'Hagan, D. B. Kell, *Journal of Cheminformatics* **2017**, *9*.

[63] S. Liu, Y. Wu, T. Lin, R. Abel, J. P. Redmann, C. M. Summa, V. R. Jaber, N. M. Lim, D. L. Mobley, *Journal of Computer-Aided Molecular Design* **2013**, *27*, 755–770.

[64] M. Fleck, M. Wieder, S. Boresch, *unpublished*, manuscript in review.

[65] S. Jo, T. Kim, V. G. Iyer, W. Im, *Journal of Computational Chemistry* **2008**, *29*, 1859–1865.

[66] J. Lee, X. Cheng, J. M. Swails, M. S. Yeom, P. K. Eastman, J. A. Lemkul, S. Wei, J. Buckner, J. C. Jeong, Y. Qi, S. Jo, V. S. Pande, D. A. Case, C. L. Brooks, A. D. MacKerell, J. B. Klauda, W. Im, *Journal of Chemical Theory and Computation* **2015**, *12*, 405–413.

[67] S. E. Feller, Y. Zhang, R. W. Pastor, B. R. Brooks, *The Journal of Chemical Physics* **1995**, *103*, 4613–4621.

[68] A.-P. Hynninen, M. F. Crowley, *Journal of Computational Chemistry* **2013**, *35*, 406–413.

[69] P. P. Ewald, *Annalen der Physik* **1921**, *369*, 253–287.

[70] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, M. L. Klein, *The Journal of Chemical Physics* **1983**, *79*, 926–935.

[71] E. Neria, S. Fischer, M. Karplus, *The Journal of Chemical Physics* **1996**, *105*, 1902–1921.

[72] J.-P. Ryckaert, G. Ciccotti, H. J. Berendsen, *Journal of Computational Physics* **1977**, *23*, 327–341.

[73] B. L. Welch, *Biometrika* **1947**, *34*, 28.

[74] M. Krzywinski, N. Altman, *Nature Methods* **2013**, *10*, 1041–1042.

[75] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, l. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, S. 1. 0. Contributors, *Nature Methods* **2020**.