Contents

Chapter 1

Introduction

The complexes model is a hierarchical coarse-grained (CG) protein model that has been implemented with a Monte-Carlo engine to generate structures of protein complexes [1]. The model is commonly refereed to as the Kim-Hummer (KH) model in the literature. Since the original publication, the forcefield has been used in a large variety of applications. Those include the study the binding kinetics of the HIV-1 capsid proteins [2, 3], the kinetic behavior of proteins in crowded environments [4–7], folding of knotted protein [8–13], enhancing the structural resolution of experiments [14–18], protein-protein interactions [19–21], protein design [22], docking [23], multi-enzyme complexes [24–27]. The model has also been extended to simulate intrinsically disordered proteins and protein phase separation [7].

The complexes model is implemented in a C++14 program Complexes++ and a Python tool pycomplexes . Complexes++ implements the Monte-Carlo engine and pycomplexes is a helper library and command line interface (CLI) tool to setup simulations and visualize them, see Figure 1.1. The Monte-Carlo integrator accepts input files in the CPLX format and configuration files for simulations. The split enforces that a well defined file format exists that uniquely defines a simulation. We have decided to use the YAML standard [28] for the CPLX files. A library to write YAML files exists for many programming languages allowing easier integration into existing workflow without forcing a specific programming language.

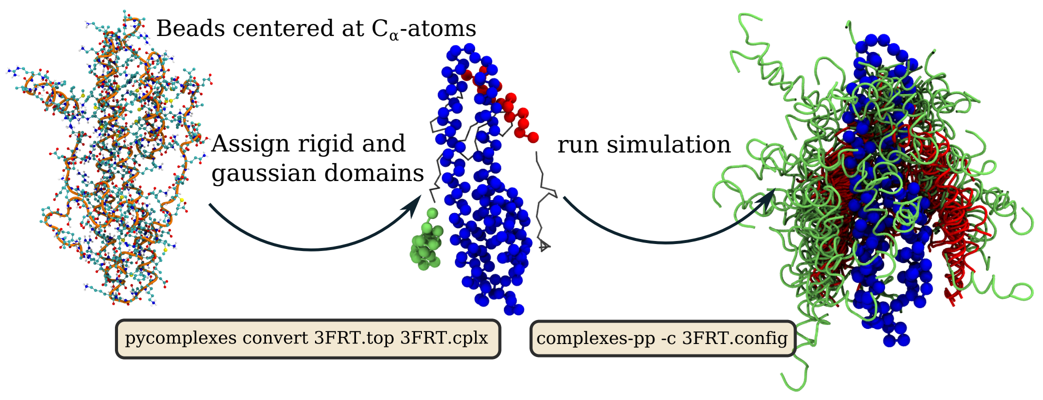
Beads centered at C_α-atoms
Assign rigid and gaussian domains
run simulation
pycomplexes convert 3FRT.top 3FRT.cplx
complexes-pp -c 3FRT.config

Figure 1.1: Example use case how to go from a single known structure to an ensemble of structures using complexes. To prepare the simulation domain types have to be assigned to amino acids and a CPLX file has to be generated.

Chapter 2

Theory

The complexes model describes proteins and large macromolecular structures at three levels of coarse-graining. The first level is a bead. Beads are interaction sites that are used to evaluate potentials and represent a single amino acid, centered on the C_α atoms. The second level is a domain. Domains are collections of beads that define how the bead positions are propagated in a simulation. The complexes model has rigid and flexible domains. The last level is called a topology. It is a collection of connected domains. Topologies are useful to develop efficient sampling algorithms for simulations with multiple complexes. In this thesis, the general expression of the forcefield will be referred to as the complexes model, when specific values for forcefield parameters are given we refer to them as the KH model. In the following, the bead model and pair potential for different beads and different domains will be explained.

2.1 Beads

Beads are the interaction sites at which the force field and additional restraint potentials are evaluated. In the complexes model [1] amino acids are modeled as single beads centered on the C_α atoms. As energy function, a Lennard-Jones like potential is used for effective interactions of native and non-native contacts and a Coulomb term with an exponential screening term for the electrostatics. The potential energies are by convention calculated in units of $k_\mathrm{B}T$, with $T = 300\,\mathrm{K}$ as the reference temperature.

The Lennard-Jones like potential U_{LJ}, between beads i and j, consists of four different branches to model attractive and repulsive interactions

$$U_{LJ}(r, \sigma_{ij}, \epsilon_{ij}) = \begin{cases} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & \text{if } \epsilon_{ij} < 0 \\ 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] + 2\epsilon_{ij} & \text{if } \epsilon_{ij} > 0 \text{ and } r < 2^{1/6}\sigma_{ij} \\ -4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & \text{if } \epsilon_{ij} > 0 \text{ and } r > 2^{1/6}\sigma_{ij} \\ .01\left(\frac{\sigma_{ij}}{r}\right)^{12} & \text{if } \epsilon_{ij} = 0, \end{cases} \tag{2.1}$$

with r the distance between the beads, the bead type pair parameters σ_{ij} and ϵ_{ij} for the contact distance and interaction energy, respectively. For $\epsilon_{ij} < 0$ this is the standard Lennard-Jones potential, see Figure 2.1. For $\epsilon_{ij} > 0$ this potential is purely repulsive, see Figure 2.1. In case of $\epsilon_{ij} = 0$ the potential is a hard wall slightly shorter than the Lennard-Jones minimum of $2^{1/6}\sigma_{ij}$ to avoid overlaps if additional potentials are attractive and have singularities at $r = 0$, for example electrostatic potentials. At contact $r = \sigma_{ij}$ this potential gives equal contributions from attractive and repulsive pairs with opposite sign. The parameters ϵ_{ij} are derived from the knowledge-based statistical contact potentials e_{ij} by Miyazawa and Jernigan (MJ) [29]. The MJ contact potentials have to be scaled to

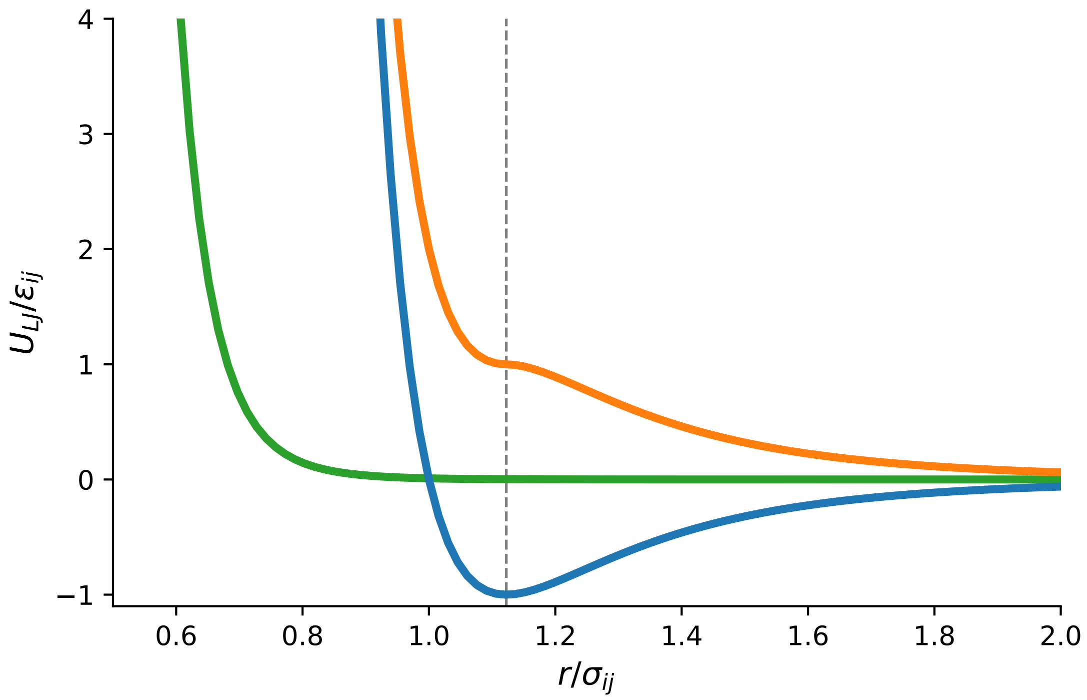

Figure 2.1: Modified Lennard Jones potential, eq 2.1, used in the complexes model in reduced units. The attractive branch $\epsilon_{ij} < 0$ is shown in blue, the repulsive part $\epsilon_{ij} > 0$ is shown in orange and the branch for $\epsilon_{ij} = 0$ in green. The gray dashed line shows the minima at $2^{1/6}$ of the attractive branch.

account for the added electrostatic interactions and the preference of residue-residue to residue-solvent interactions has to be balanced. In the complexes model this is done by scaling with a parameter λ for the electrostatic interaction and shifting the interaction with a parameter e_0 for the residue-residue to residue-solvent interactions with

$$\epsilon_{ij} = \lambda(e_{ij} - e_0). \tag{2.2}$$

For the KH model the values $\lambda = 0.159$ and $e_0 = -2.27\,\mathrm{k_B T}$ have been used, based on parametrizations to reproduce the experimental determined second virial coefficient of hen egg lysozyme and the dissociation constant K_d of the ubiquitin uim1 system [1]. The contact distances σ_{ij} are determined as weighted average $\sigma_{ij} = (\sigma_i + \sigma_j)/2$ from the individual amino acids diameters, Table 2.1. Note that in the original paper these values have been incorrectly labeled as radii.

Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile
5.0	6.6	5.7	5.6	5.5	6.0	5.9	4.5	6.1	6.2

Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
6.2	6.4	6.2	6.4	5.6	5.2	5.6	6.8	6.5	5.9

Table 2.1: Van-der-Waals diameters of amino acids in Å[1].

The electrostatic potential consists of Coulomb interactions with an ionic-screening term

$$U_{el}(r) = \underbrace{\frac{q_i q_j e^2}{4\pi\epsilon_0 D_{el} r}}_{\text{coulomb}} \underbrace{\exp\left(\frac{-r}{\xi}\right)}_{\text{ionic-screening}} \underbrace{\frac{1}{k_{\mathrm{B}}T}}_{\text{scaling factor}}, \tag{2.3}$$

with e the elementary charge, ϵ_0 the vacuum permittivity, D_{el} the dielectric constant and ξ the Debye-length. The scaling factor of $1/k_{\mathrm{B}}T$ is used to convert the electrostatic energy into units of $k_{\mathrm{B}}T$. Bead charges are set according to amino acid type corresponding to a pH of 7. Arginine and lysine are charged with $+e$, histidine with $+\frac{1}{2}e$, due to its isoelectric point and aspartate and glutamine with $-e$. Charges for other amino acids are set to 0. The ionic-screening term is used to set the salt concentration of the environment with the Debye-length

$$\xi = \sqrt{\frac{\epsilon_r \epsilon_0 k_{\mathrm{B}} T}{e^2 N_A 2I}}, \tag{2.4}$$

where ϵ_r the absolute permittivity, and I the ionic strength. For a typical salt concentration of 100 mM NaCl ξ is about 10 Å.

2.2 Domains

Proteins and multiprotein complexes consist of multiple units that are connected together. The domains are the second abstraction level of the complexes model that connect beads. This abstraction is based on the assumption that a protein complex consists of different proteins that behave as single units for a short time span. There can be parts that are rigid during the lifespan of the protein complex and short stretches of unstructured peptide chains that serve to tether together as stiff parts. Domains are used to model this varied behavior. To model the two different behaviors the complexes model has rigid and flexible domains.

2.2.1 Rigid Domain

Rigid domains are the simplest form of a domain and the most versatile at the same time. As the name suggests in a rigid domain the internal coordinates of the beads in the domain do not change over time. Rigid domains are so versatile because they can be used to model very different things. The obvious cases are rigid protein parts like an α-helix or a β-sheet. While not described in the original complexes model it is possible to describe a rigid domain at an even coarser level by grouping together amino acids. Using such a CG description requires finding new forcefield parameters for the interactions but this versatility makes it possible to set up simulations that incorporate experimental data with an appropriate detailed given experimental uncertainties and prior knowledge.

2.2.2 Flexible Domain

A compelling feature of the original complexes implementation [1] was its explicit treatment of flexible chains in protein complexes. These chains served as an anchor to link rigid domains together. In the original paper [1] a peptide chain model using bond, angle and dihedral potentials similar to molecular dynamics (MD) forcefields [30] has been used. The advantage of such a model is that amino acids are modeled explicitly and can interact with the rigid domains. A drawback of using this model with a Monte-Carlo scheme is that only small movements of the whole chain can be made to have small energy differences and therefore good acceptance probabilities. As a side effect of this, the chain diffuses slowly through configuration space and the overall diffusion of attached rigid domains is limited by the linker instead of the translation and rotation step-size chosen for the rigid domains. While such an explicit model can work for smaller complexes [14] it would make simulations of larger complexes [15] difficult.

A linker model that has a limited influence on the diffusion of the rigid domains would be better suited to quickly sample configuration space. It has been shown that for a linker only the length is important and not the exact dynamics [24]. We use this to make the assumption that the linker only has to hold two rigid domains together and has no other functional purpose and replace the explicit peptide chain with a potential of mean force (PMF) that only depends on the distance between the two rigid domains and the number of amino acids in the linker. This PMF potential acts as a restraint potential ensuring that the distance between two rigid domains is physically correct. Because no explicit beads are involved the diffusion of the rigid domains is only influenced by the selected translation and rotation step-size. We can use a Gaussian chain polymer model to calculate a suitable PMF. Gaussian chains are suitable models previously used to explain fluorescence resonance energy transfer (FRET) experiments [31].

In the Gaussian chain model the beads are point particles connected by harmonic springs. The average distance b between two beads determines the spring constant. For a three dimensional chain the distances $R_{ij} = |\vec{r}_j - \vec{r}_i|$ between beads i and j, irrespective of the direction, is distributed according to [32]

$$P(R_{ij}) = 4\pi R_{ij}^2 \left(\frac{3}{2\pi\langle R_{ij}^2\rangle}\right)^{3/2} \exp\left(-\frac{3R_{ij}^2}{2\langle R_{ij}^2\rangle}\right), R_{ij} > 0, \tag{2.5}$$

where $\langle R_{ij}^2\rangle = b^2(j-i)$ is the mean squared distances between beads i and j. The factor $4\pi R_{ij}^2$ is the volume element of a shell with width dR and radius R_{ij}. The end-to-end distance for a Gaussian chain with N beads is

$$\sqrt{\langle R_{1N}^2\rangle} = \sqrt{N}b. \tag{2.6}$$

To construct a PMF for the Gaussian chain model we look at the exponential term in eq 2.5. It is a similar expression as for the Boltzmann distribution for a harmonic oscillator

$$V(\vec{r}_i, \vec{r}_j) = \frac{3}{2b^2}\frac{1}{(j-i)}(\vec{r}_i - \vec{r}_j)^2, \tag{2.7}$$

with spring constant $3/(b^2(j-i))$. From this the potential of mean force between the ends of a Gaussian chain of length N follows as

$$PMF(\vec{r}_1, \vec{r}_N) = \frac{3}{2b^2}\frac{1}{N-1}(\vec{r}_1 - \vec{r}_N)^2. \tag{2.8}$$

To compare the distribution of end-to-end distances obtained from this PMF we run a Monte-Carlo simulation using eq 2.8 for a linker of length $N = 200$ and a bond length $b = 3.81\,\text{Å}$ [33], see Figure 2.2. The distribution of end-to-end distances created using the PMF agrees well with the expected distribution of the Gaussian polymer model. This PMF has been previously used to model proteins and ribonucleic acid (RNA) [34].

The final PMF that we use will be between two beads of the two connected rigid domains. These two beads have to be added to the length N of the linker. Therefore the final PMF for a linker of length N is

$$PMF(\vec{r}_0, \vec{r}_{N+1}) = \frac{3}{2b^2}\frac{1}{N+1}(\vec{r}_0 - \vec{r}_{N+1})^2, \tag{2.9}$$

with $\vec{r}_0$ and $\vec{r}_{N+1}$ being the two beads of the rigid domains that are connected by the linker.

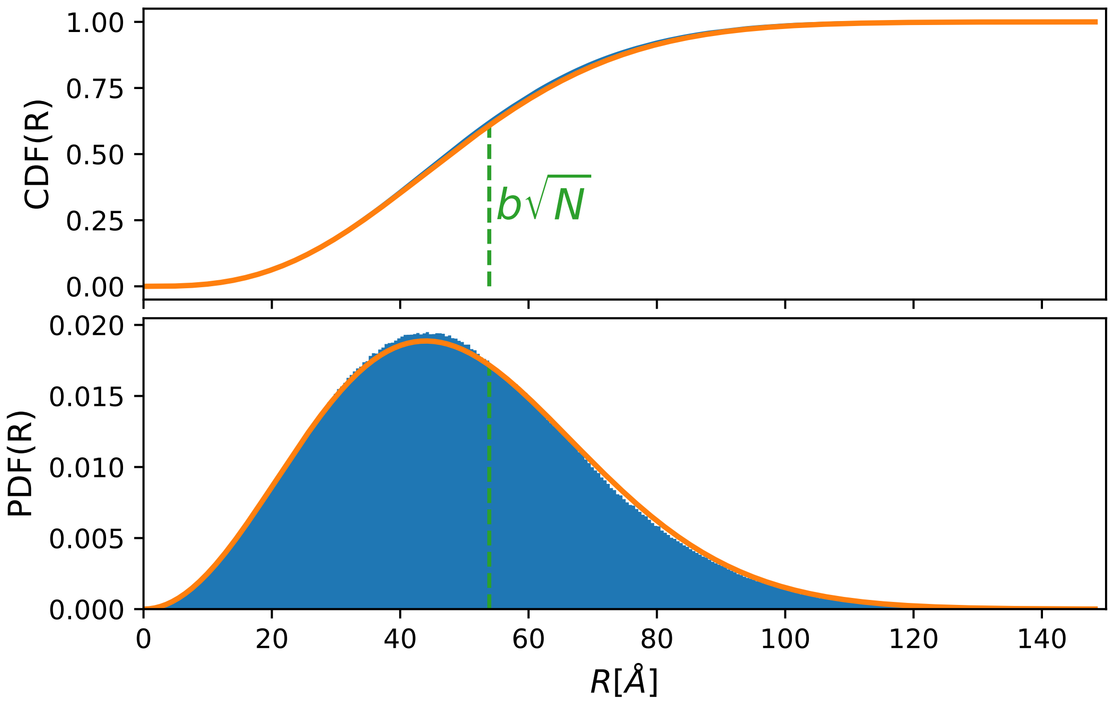

Figure 2.2: Cumulative distribution function (top) and probability density function (bottom) of the end-to-end distance R for a Gaussian chain of length $N = 200$ and bond length $b = 3.81\,\AA$. The distributions for the ideal Gaussian chain are shown in orange. Results from an ensemble of 1 million distances obtained from a Monte-Carlo simulation using the PMF eq 2.8 are shown in blue. The value of the mean end-to-end distance, eq 2.6, is marked in green.

2.2.3 Generating Explicit Beads for Linker Model

While the PMF, eq 2.8, restraint potential, without explicit modeling of the linker beads, is good for fast exploration of phase space some applications, like the comparison of simulations to small angle x-ray scattering (SAXS) measurements [14, 15], require to have explicit beads for the linker domains. We now describe an iterative algorithm to generate positions for the linker beads given fixed positions for the first and last bead of the linker. Given the positions of bead N and 1 a single bead $N-1$ can be generated with the following algorithm:

1. Randomly choose a distance d_{start} between bead N and $N-1$ distributed according to eq 2.5. This distance defines a sphere around bead N on which bead $N-1$ will be placed. (Figure 2.3, red circle)
2. Choose a distance d_{end} between bead 1 and $N-1$ so that the sphere around bead 1 intersects with the sphere calculated in step 1. (Figure 2.3, blue cone)
3. Choose a random point on the intersection of the red and blue sphere to place the bead $N-1$ (Figure 2.3, green bead). In three dimensions the intersection is a circle so a random angle θ has to be drawn.

To grow the next bead, with index $N-2$, simply repeat the algorithm with bead $N-1$ as new starting point to choose the random distance $R_{N-1,N-2}$. Repeat this algorithm until all beads are generated. This algorithm gives the three distance between the beads and

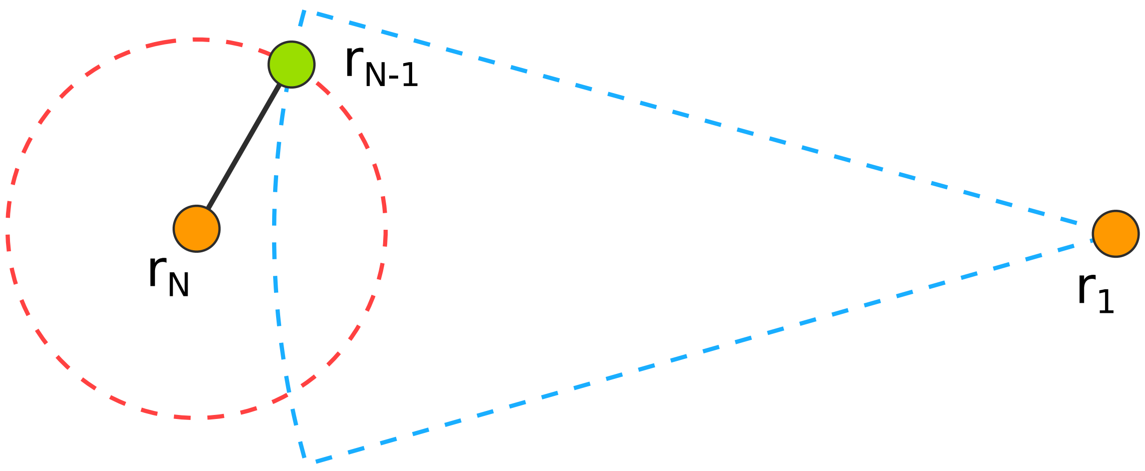

Figure 2.3: Example of distances drawn for a new bead $N-1$ (green) between to fixed endpoints (orange) of a Gaussian linker of length N in two dimensions. The red circle is the distance between the beads N and $N-1$. This distance is randomly chosen from eq 2.5. The bead $N-1$ can be placed anywhere on this circle. The distance between bead 1 and $N-1$ now has to be chosen so that circle (blue) drawn around bead 1 intersects with the first circle (red). In two dimensions this restricts the positions of the new bead to the two intersection points. In three dimensions it would be restricted to a sphere.

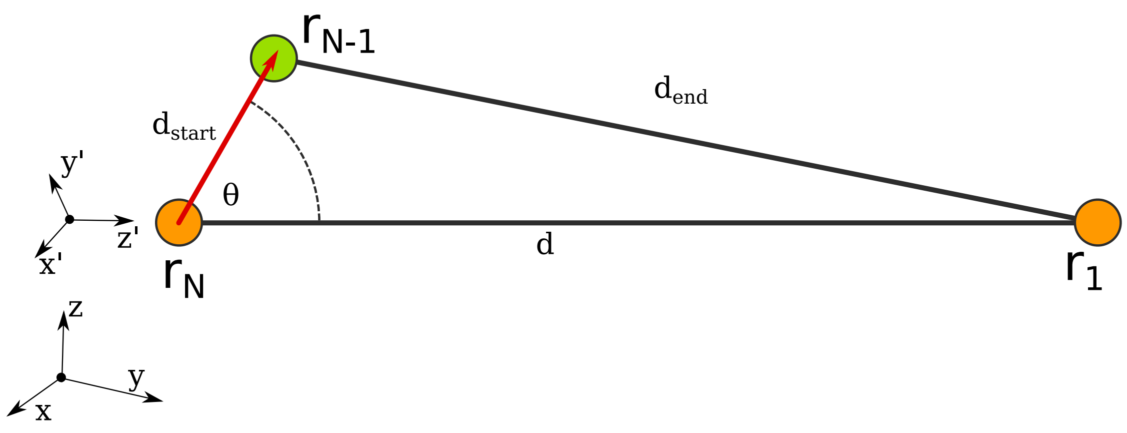

Figure 2.4: Schematic for calculating for calculating the position of bead $\vec{r}_{N-1}$ given the positions of $\vec{r}_N$, $\vec{r}_1$, the distances d, d_{start}, d_{end} and an angle ϕ (not shown). The coordinate system $(\vec{x}, \vec{y}, \vec{z})$ is our reference coordinate system. The coordinate system $(\vec{x'}, \vec{y'}, \vec{z'})$ is used to calculate $\vec{r}_{N-1}$.

orientation that uniquely determines where a new bead should be placed. To calculate the actual coordinates in the coordinate system of the simulation we are using the following algorithm.

1. Determine axis $\vec{z'}$ along the vector $\vec{r_N} - \vec{r_1}$.
2. Determine perpendicular axes $\vec{x'} = \left(\frac{-z'_1 - z'_2}{z'_0}, 1, 1\right)^T$ and normalize. Permutate in elements of $\vec{x'}$ if $z'_0 = 0$.
3. Determine $\vec{y'} = \vec{x'} \times \vec{z'}$, with $\times$ the cross product.

4. Determine angle ϕ using the law of cosines $\cos(\phi) = \frac{d_{\text{start}}^2 + d^2 - d_{\text{end}}^2}{2 d_{\text{start}} d}$, with $d = |\vec{r_n} - \vec{r_1}|$ see Figure 2.4.
5. Calculate $\vec{r_{N-1}}$ in the coordinate system spanned by $(\vec{x'}, \vec{y'}, \vec{z'})$ from the spherical coordinates given by $(d_{\text{start}}, \theta, \phi)$, see Figure 2.4.
6. Convert $\vec{r_{N-1}}$ into reference coordinate system $(\vec{x}, \vec{y}, \vec{z})$.

Linker configurations generated using the above two algorithms will include overlaps between neighboring beads, see Figure 2.5. To avoid overlaps and generate more extended configurations it's sufficient to add overlap checks in step 1 and 3 in the first algorithm algorithm, i.e. when two beads are to close to each other.

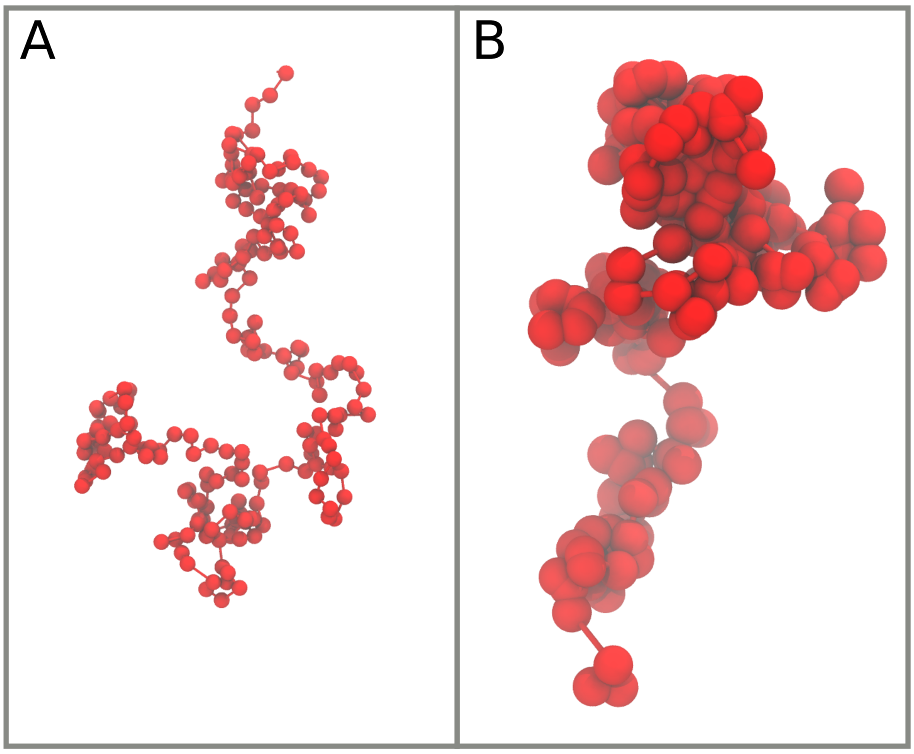

Figure 2.5: Gaussian chain with (left) and without (right) overlap check. Example configuration for a alanine chain of length 200 with a bond-length of b =3.81 Å. All beads are drawn with a diameter of 3.81 Å.

2.2.4 Relaxation of Gaussian Polymerchain

The structures produced by the Gaussian chain growing algorithms are not very physical. The distances between beads vary by a standard deviation of 1 Å with a mean of 5 Å in a single chain when the chain is grown with overlap checks. The fluctuation in typical protein structures is less than a hundredth of an Å with a mean distance of 3.81 Å [33]. For the generation of more physical bead coordinates, it is, therefore, necessary to relax the structures generated by the previous algorithm. For the relaxation, the force field developed by Kim and Hummer [1] can be used.

The forcefield consists of bond potentials for pseudobonds for C_α–C_α, angle potentials for pseudo angles C_α–C_α–C_α and torsion potentials for pseudo torsion C_α–C_α–C_α–C_α. The bond potential is a harmonic potential

$$U_{\text{bond}} = \frac{1}{2} k (r - r_0)^2, \tag{2.10}$$

with r the C_α–C_α distance, $r_0 = 3.81$ Å the reference distance and $k = 378\,\mathrm{kcal\,mol^{-1}\,Å^{-2}}$ the spring constant [35]. The pseudoangle potentials is given a double well potential [33]

$$\exp[-\gamma U_{\mathrm{angle}}(\theta)] = \exp[-\gamma(k_\alpha(\theta-\theta_\alpha)+\epsilon_\alpha)] + \exp[-\gamma k_\beta(\theta-\theta_\beta)^2], \tag{2.11}$$

where θ is the angle between $C_\alpha - C_\alpha - C_\alpha$, the constants are $\gamma = 0.1\,\mathrm{mol\,kcal^{-1}}$, $\epsilon_\alpha = 4.3\,\mathrm{kcal\,mol^{-1}}$, $\theta_\alpha = 1.6\,\mathrm{rad}$, $\theta_\beta = 2.27\,\mathrm{rad}$, $k_\alpha = 106.4\,\mathrm{kcal\,mol^{-1}rad^{-2}}$ and $k_\beta = 23.6\,\mathrm{kcal\,mol^{-1}rad^{-2}}$. This potential accounts for the helical and extended pseudoangles. The torsion potential is given by [35]

$$U_{\mathrm{torsion}}(\phi) = \sum_{n=1}^{4}[1+\cos(n\phi-\delta_n)]V_n, \tag{2.12}$$

where ϕ is the torsion angle of the middle two beads in $C_\alpha - C_\alpha - C_\alpha - C_\alpha$. The constants V_n and δ_n are chosen for alanine as $V_n = [0.936472, 2.307767, 0.131743, 0.613133]$ $\mathrm{k_BT}$ and $\delta_n = [287.354830, 271.691192, 180.488748, 108.041256]$ rad [35].

The relaxation with the above described potential is done using a Monte-Carlo algorithm. For trial moves the position of individual beads is changed. The start and end bead are treated as fixed. For a linker of length N the probability to pick a bead is uniform between all $N-2$ beads that are allowed to move. A sweep consists of $N-2$ trial moves. Because the structure is only supposed to be relaxed it isn't necessary to generate structures from an equilibrium distribution and therefore detailed balance does not need to be preserved. Here the move width is adjusted after each sweep to achieve a target acceptance ratio of 30 %. If after a sweep the acceptance ratio is larger than 30 % the step-width is increased by 10 % and decreased if the acceptance ration is below 30 %.

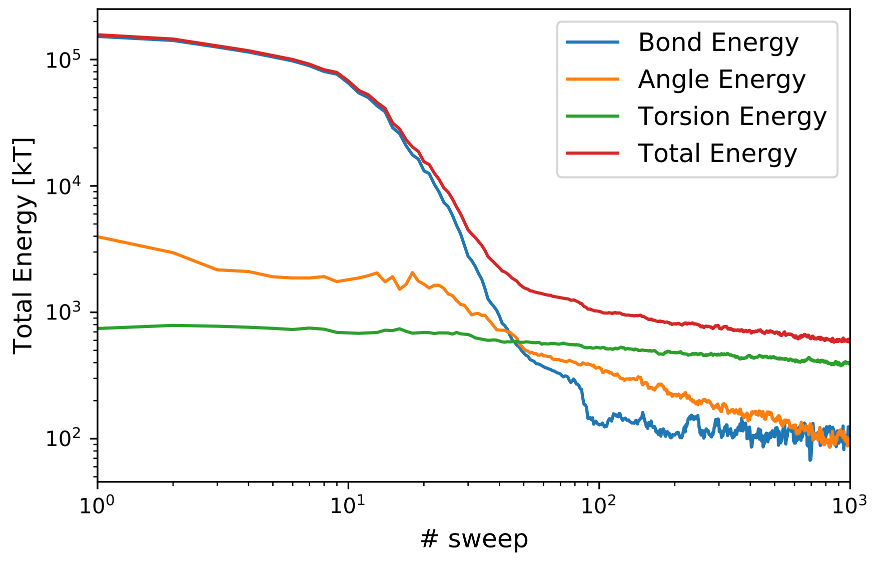

Figure 2.6: Energies of a nonoverlapping Gaussian chain during relaxation. The chain is 200 beads long and the initial structure was generated using the Gaussian chain model with no overlaps. The acceptance rate during the Monte Carlo simulation was set to target 30 %. The bond energy is shown in blue, the angle energy in orange, the torsion energy in green and the total energy in red.

Energy contributions for each potential from a relaxation run for a chain of 200 beads is shown in Figure 2.6. In the beginning, the energy is dominated by the bond potential, this is due to the fact that average bond-length is larger than 3.81 Å. The bond lengths are fully relaxed after around 200 sweeps. The angle potentials start to relax around sweep 50 when the bond energy has already dropped by a factor of two. The angle potential is fully relaxed after around 1000 sweeps. The last potential to relax are the torsion angles. After about 500 sweeps the torsion angles start to see a more pronounced decrease after 3000 sweeps when the other two potentials haven been fully relaxed. The energy difference in the torsion potential from the beginning of the simulation to the final structure is significantly less than for the other two terms in the energy. The reason for this could be that the starting structures generated by the Gaussian chain algorithm are particularly ill chosen or that the simple spatial trial moves of the beads are not good for relaxing this potential function.

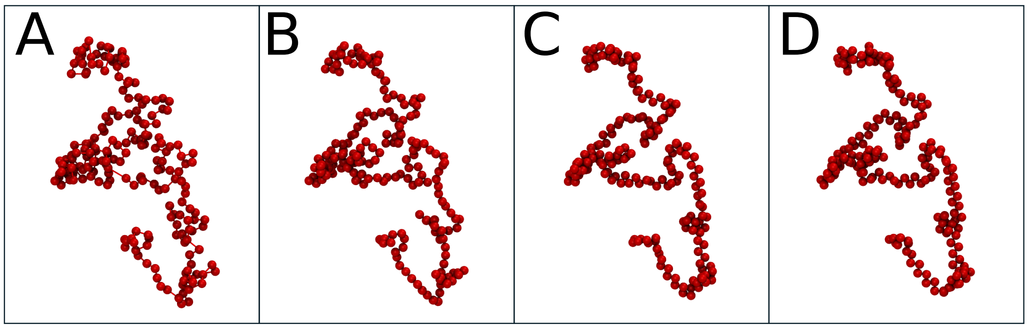

Figure 2.7: Linker configuration before relaxation (A) and after using only the bond potential (B), using the bond and angle potential (C), and using the bond, angle and torsion potential (D). All relaxation runs used the same initial structure (A).

The structures generated by relaxing an initial configuration from the Gaussian chain algorithm can be seen in Figure 2.7. For comparison, single structures have been generated with the full potential, only the bond potential, and the bond and angle potential. The structures have all been generated from the same initial structure and relaxation runs where 1000 sweeps long. The bond potential alone has the biggest visual influence on the structure by achieving a more uniform bond distance. The addition of the angle potential also gives a visual improvement. Differentiating the bond and angle potential structure from the structure with the full potential is difficult as both are very similar.

2.2.5 Comparison of Unfolded Proteins and Linker Model

To understand how well the model describes unfolded protein regions I will compare the radius of gyration R_G, as a measure of compactness, of our model with experimental data. For unfolded proteins the R_G has been determined experimentally in dependence on the protein length with denaturants [36]

$$\langle R_G \rangle = R_0 N^{\nu}, \tag{2.13}$$

with $R_0 = 1.927^{+0.271}_{-0.238}$Å and $\nu = 0.598 \pm 0.028$. The radius of gyration of the Gaussian chain model is

$$\langle R_G \rangle = \sqrt{\frac{1}{6}\frac{N(N+2)}{N+1}}b \approx \sqrt{\frac{1}{6}}N^{1/2}b. \tag{2.14}$$

This scaling behavior is slightly different with $\nu = .5$ and $R_0 = 1.555$ Å. Therefore it is unlikely that the Gaussian polymer without overlap checks reproduces the R_G values of a

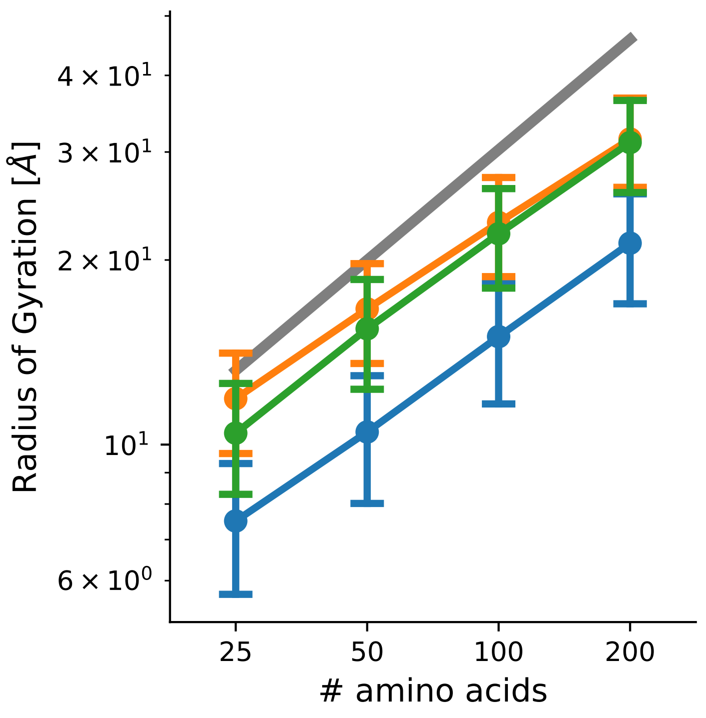

Figure 2.8: Radius of Gyration computed for a linker with of different number of amino acids. For each length 1000 structures have been generated with overlap check (orange), without (blue), and fill relaxation (green). Error bars donate the standard deviation. The experimental R_G scaling law for denatured proteins [36] is shown as gray line.

denatured protein for any number of beads. To compare the R_G scaling behavior of the linker growth algorithm to the scaling of denatured proteins we generate 1000 different structures for a linker of 25 to 200 beads length. To get the values of a truly free chain for each simulation the linker was padded with 25 beads in the front and end so that only the middle N beads have been used to calculate the R_G. The bond-length was set to 3.81 Å [33]. Start and end points have been placed at the optimal end-to-end distance, eq 2.6. Results are shown in Figure 2.8. As anticipated without an overlap check the R_G values are systematically different. But with overlap checks enabled the Gaussian polymer is within one standard of the experimental values if $N < 50$.

It should be noted that for intrinsically disordered proteins it has been shown that the R_G with denaturants is larger than of the protein observed in natural conditions [37–40]. Therefore our Gaussian polymer model is a good enough description for the flexible domains. A similar model has been employed to study intrinsically disordered proteins [7].

2.3 Topologies

Topologies are a collection of connected domains. They can be used to model large protein complexes that consist of multiple domains. The connection is typically modeled as a distance based potential between two beads of the connected domains. The connection potential can be chosen according to the Gaussian chain model eq 2.8 or as a normal harmonic spring.

Chapter 3

Monte-Carlo Engine Complexes++

Complexes++ is a Monte Carlo simulation engine implementing the complexes protein model. The program is written in modern C++14 with a focus on being extensible. The program contains an extensible Monte-Carlo engine, the modified Lennard-Jones potential, eq 2.1 and other pair potentials, the Monte-Carlo movements for different domain types, and a cell-list algorithm [41] to speed up the energy evaluations.

3.1 Flexible and Extensible Core Algorithms

The core algorithms of the Monte-Carlo engine have the ability to add new trial moves for domains and to add new Monte-Carlo algorithms for different statistical ensembles or enhanced sampling techniques. This extensibility is achieved thought a combination of run-time polymorphism and use of modern C++14. In Complexes++ , this technique is used to implement domain trial moves, Monte-Carlo algorithms, evaluation of pair potentials, and a cell-list algorithm. In C++ run-time polymorphism is implemented through abstract classes and inheritance. In Complexes++ abstract classes are marked with the preposition "Abstract". The most interesting abstract classes to add new features to Complexes++ are `AbstractDomain`, `AbstractPairKernel`, `AbstractConnection`, `AbstractInteractionGrid` and `AbstractMcAlgo`. See Figure 3.1 for a schematic how these classes interact with each other during a simulation.

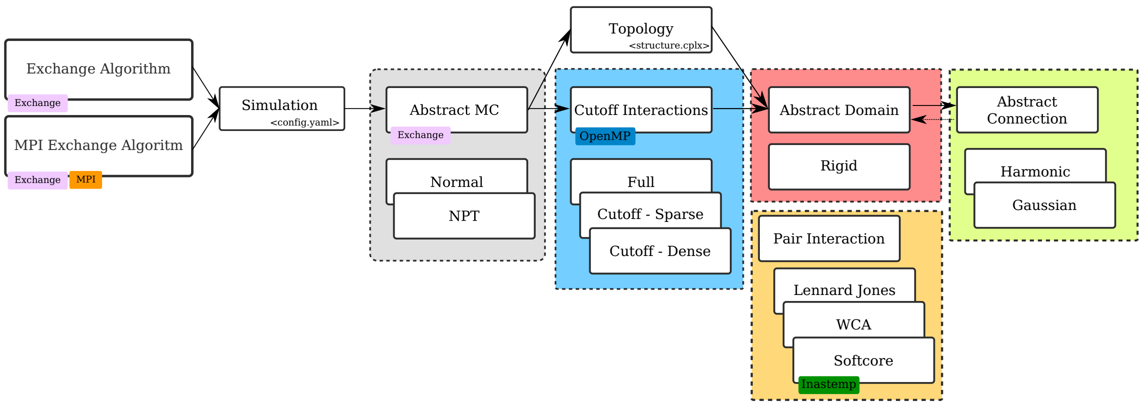

Figure 3.1: Schematic of the classes used in the Complexes++ program and how they interact with each other (arrows). Abstract classes and their explicit implementations are shown as colored boxes. Nested boxes show different implementations of an abstract class. Names are as in the code.

3.2 Monte-Carlo Algorithm

The main logic to run a Monte-Carlo simulation is implement in `AbstractMcAlgo` with a virtual method called `sweep` to implement a sweep. Currently Complexes++ implements sweep functions for the NVT and NΠT ensemble [41]. In addition two different acceptance functions, Metropolis [42] and Glauber [43] have been implemented as well. All Monte-Carlo sweep algorithms work with all acceptance functions.

For a sweep in the NVT ensemble domains are chosen randomly from a uniform distribution. If N is the number of domains that than a sweep will make N trial moves and select a random domain with probability $1/N$ for each trial. In the NΠT ensemble trial moves are generated for the domains and the volume. To account for the additional volume move a sweep consists of $N+1$ trial moves. At each trial, the probability to make a volume move is $1/(N+1)$. If a domain move was chosen the domain to be moved is chosen randomly with a $1/N$ probability. For the volume moves the domains are re-scaled so that the centroid of the domain is scaled by $\sqrt[3]{(V+dV)/V}$. The NΠT algorithm is called NPT in the code and configuration files.

3.3 Boundary Conditions

Complexes++ treats boundary effects using periodic boundary conditions [41]. Under these conditions, the centroid of all domains is placed inside of the unit-cell. Complexes++ only implements rectangular unit-cells. Because Complexes++ keeps the centroid of a domain within the box it can happen that some beads are outside of the box.

To determine the distance between two beads Complexes++ uses the minimum image convention (MIC) [41]. Complexes++ uses an efficient implementation to calculate the minimum image distance, Algorithm 1. This algorithm is an extension of a fast algorithm [44], where the while-loop ensures it works for any distance to ensure the algorithm will also work if trial moves displace a domain by more than one periodic image. To ensure that the number of iterations of the while-loop are as small as possible Complexes++ ensures that the centroid of a domain are inside the unit-cell.

ALGORITHM 1: Algorithm to efficiently convert the distance between two beads to the minimum image distance it domain centers are inside of the simulation box.

```
function mic(distance[3], box[3])
    for i=0; i<3; ++i do
        while distance[i] > .5 * box[i] do
            distance[i] -= box[i]
        end
        while distance[i] < .5 * box[i] do
            distance[i] += box[i]
        end
    end
    return distance;
```

3.4 Replica Exchange Algorithms

For enhanced sampling, Complexes++ implements replica exchange algorithms [45–47]. In replica exchange simulations N independent copies of a simulation, which are further refereed to as replicas, are run simultaneously and exchanged periodically. Implemented are Temperature Replica Exchange [48], Hamiltonian replica exchange [49] and pressure replica exchange [50].

The implemented replica exchange algorithms differ by the acceptance function. For temperature replica exchange the acceptance function for two configurations i and j is [48]

$$W(i \to j) = \min(1, \exp\left((\beta_j - \beta_i)(U(x_j) - U(x_i))\right)), \tag{3.1}$$

with U being the energy function, x_i the configuration of replica i and β_i the temperature of replica i. For Hamiltonian replica exchange the acceptance function is [49]

$$W(i \to j) = \min\left(1, \exp\left(\frac{1}{\beta_i}(U_i(x_i) - U_i(x_j)) + \frac{1}{\beta_j}(U_j(x_j) - U_j(x_i))\right)\right), \tag{3.2}$$

with U_i the energy function of replica i. For pressure replica exchange the acceptance function is [50]

$$W(i \to j) = \min\left(1, \exp\left((\beta_i - \beta_j)(U(x_i) - U(x_j)) + (\beta_i P_i - \beta_j P_j)(V_i - V_j)\right)\right), \tag{3.3}$$

with P_i the pressure of replica i and V_i the volume of replica i.

The theoretical descriptions of the replica exchange algorithms do not specify which replicas are exchanged in an attempt. To increase the probability to accepted an exchange attempt in complexes only neighboring replicas are exchanged [49]. During an exchange, not all neighboring pairs are attempted to exchange, this is because the exchange between replica i and $i+1$ also depends on replica $i-1$. Rather in Complexes++ attempts are only done between even pairs when the attempt number is even and odd pairs otherwise. In an even pair, of replicas i and $i+1$, then i is even and odd for odd pairs. For consistency 0 is counted as an even number and 1 as an odd number.

Replica simulations require that multiple simulations are started. Complexes++ requires that every replica is started in its own folder. Multiple simulations can be started with the `-multidir` flag and a list of the folders containing individual replicas. The `-multidir` option alone only tells complexes to simulate multiple replicas to activate the exchange two flags `-replex` and `-replex-accept` have to be used as well. `-replex` states after how many sweeps an exchange should be attempted. `-replex-accept` states the exchange function to be used.

The replica exchange simulations are also implemented in an extensible manner. Because the different replicas exchange algorithms implemented require different variables to be changed between the replicas, Complexes++ only exchanges the coordinates of the beads and box dimensions during an exchange.

3.5 Random Numbers

Complexes++ uses pseudo random numbers to generate trial moves and accept moves. As a pseudo random number generator (RNG) it is using the mersenne twister [51] implementation in the C++ standard library. Because the RNG cannot be safely used in a multi-threaded program like Complexes++ it uses a different instance of the RNG for each thread. The individual RNG instances are seeded from random numbers from an initial RNG instance during setup. To achieve reproducability Complexes++ requires the user to provide an explicit seed for the initial RNG. This requirement ensures that two runs with the same input are identical on the same computation node.

3.6 Pair Interactions Potentials

In Complexes++ pair potentials are called pair kernels following a common terminology used in computer science. The parameters of all available pair kernels are given in the forcefield class. The parameters σ_{ij} and ϵ_{ij} are set according to bead type and shared between the pair-kernels. During a simulation pair-kernels can be chosen for each individual domain type pair present in the simulation, allowing to fine tune the interactions. The Lennard-Jones like potential eq 2.1 in combination with the electrostatic potential eq 2.3, and several other potentials have been implemented. One additional potential is the Weeks-Chandler-Anderson (WCA) potential [52]

$$U_{\mathrm{WCA}}(r, \sigma_{ij}, \epsilon_{ij}) = \begin{cases} -\epsilon_{ij} & \text{if } r < 2^{1/6}\sigma_{ij} \\ 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & \text{if } \epsilon_{ij} > 0. \end{cases} \tag{3.4}$$

An alternative version of the Lennard-Jones like potential that smoothly decays to zero is implemented with the following smoothing term

$$F_{\mathrm{smooth}}(r, a, b) = \begin{cases} 1 & \text{if } r/\sigma_{ij} < a \\ 0 & \text{if } r/\sigma_{ij} > b \\ \frac{(b^2-(r/\sigma_{ij})^2)^2(b^2+2(r/\sigma_{ij})^2-3a^2)}{(b-a)^3} & \text{otherwise }, \end{cases} \tag{3.5}$$

with $a = 1.4$ Å and $b = 1.8$ Å being the bound in which the potential decays to 0. The value for a and b are hard coded. A purely repulsive potential with

$$U_{\mathrm{repulsive}} = \left(\frac{\sigma_{ij}}{r}\right)^{12} \tag{3.6}$$

is also implemented. Also implemented is a soft-core potential [53] that allows to tune how soft the beads are and therefore they can potentially overlap. The soft-core potential is a modification of the Lennard-Jones like potential without the explicit repulsion branch,

$$U_{\mathrm{SC}}(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}^6}{\alpha\sigma_{ij}^6 + (r_{ij}-s)^6}\right)^2 - \left(\frac{\sigma_{ij}^6}{\alpha\sigma_{ij}^6 + (r_{ij}-s)^6}\right)\right], \tag{3.7}$$

with α the parameter to tune the softness of the beads, and $s = \left(\sqrt[6]{2} - \sqrt[6]{2-\alpha}\right)\sigma_{ij}$ a shift parameter to ensure that the minimum is always at $\sqrt[6]{2}$ independent of α. The other branches of the Lennard-Jones like potential can be obtained by applying the same modifications. α can be changed in the range of zero to one, with $\alpha = 1$ allowing full overlap of the beads as $U_{\mathrm{SC}}(r = 0) = 0$ and $\alpha = 0$ recovering the Lennard-Jones like potential $U_{\mathrm{SC}}(r = 0) = \infty$, eq 2.1. Because this potential explicitly allows overlaps the electrostatics potential also has to be changed to remove the divergence at $r = 0$. For this we use the potential between two Gaussian charge distributions [54]

$$U_{\mathrm{el}}(r_{ij}) = \frac{q_i q_j}{4\pi\epsilon_0 D}\frac{\mathrm{erf}\left(r_{ij}\sqrt{\lambda_{ij}}\right)}{r_{ij}}\exp\left(-\frac{r_{ij}}{\zeta}\right)\frac{1}{k_{\mathrm{B}}T} \tag{3.8}$$

with $\lambda_{ij} = \frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$ and λ_i the charge radius of bead i. The charge radii are specified in a forcefield for every bead type. In the standard Complexes++ forcefield all radii are set to one.

The different pair kernels are implemented with a common abstract base class `AbstractPairKernel`. The Lennard-Jones like and WCA potential have been implemented using Inastemp, a portable single instruction multiple data (SIMD) library, [55].

3.7 Bead and Domain Implementation

No specific bead class exists in Complexes++ , instead, domains contain all information specific to the beads as several lists. This design scheme follows the "struct of arrays" pattern [56] used to optimize memory access. For the beads a domain stores the coordinates in a `m_xyz` field, the charges in `m_charges` and the bead types in `m_beads`. These three fields are the only information needed to evaluate the potential energy with the pair potential functions. Domains are also implemented with run-time polymorphism in an AbstractDomain class that contains information about the beads and coordinates. Derived classes only need to implement trial moves.

In addition to bead information, a domain also contains a unique id and a type id that are defined at runtime. The type id is used to find the corresponding pair kernel when evaluating the energy with a different domain. Choosing a pair-kernel for domain type pairs and using a struct of arrays pattern allows evaluating the pair-kernel for groups of beads. This pattern can be efficiently implemented using inastemp [55]. Because the evaluation of the pair-kernel is the most expensive calculation of Complexes++ this pattern ensures that Complexes++ has good performance while using run-time polymorphism. The type ids are also used to create domains with the same type of move but different parametrizations for it. For example, the rigid domains have two parameters for translation and rotation. Having the final parametrization set at runtime allows creating two distinct rigid domain types for different proteins. This flexibility is useful for simulations with a mixture of small and large domains. The translation and rotation of larger domains can be chosen smaller to increase the acceptance rate for their moves and small domains can be set to large translation and rotation. Allowing to achieve optimal phase space exploration rate for a simulation.

In Complexes++ only the rigid domain types is implemented. For the Monte-Carlo moves the rigid domains can be translated by an arbitrary vector, each component is chosen randomly from the range $[-a, +a]$, with a the maximal displacement. The rotations are generated by choosing a random rotation axes in the unit cube and a random rotation angle [41]. This method generates a known artifact that rotation axes along the edges are over sampled. Because the all edges are equally sampled it does not affect generated ensembles. In each trial move the probability to make a translation or a rotation move is one half.

3.8 Flexible Linkers and Connection Potentials

As described in the theory flexible linker domains are replaced with effective potentials from a Gaussian chain polymer model. For this Complexes++ implements connection potentials in a `Connection` class. A connection contains the ids of the two domains that are connected and the corresponding bead ids in the domains. For each domain, a list of connections is stored in the class. So if domains i and j are connected both contain the same connection. This duplication of data makes it easier finding the corresponding connections for a domain during the energy calculation. This compromise was chosen as it is expected that the number of connections is significantly smaller the number of beads. Currently implemented connection potentials are a flat potential (zero everywhere), a harmonic potential, and a Gaussian chain PMF potential, eq 2.9.

3.9 Cell List Algorithm

The other important part of the performance of complexes is the cell list algorithm [41] used to reduce the number of interactions needed to evaluate. Cell-list algorithms reduce the computation time to calculate the full energy from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ with N being the number of beads in a simulation. The algorithm stores for every cell continues intervals of beads that are in the corresponding cell. Every bead in a domain is given a unique id defined by the order in which they appear in the input files. The interval will therefore only store the id of the first bead entering the cell and the length of beads in the cell. For domains that are larger than a single cell it can happen that two different intervals are in a cell, see the red domain in Figure 3.2 that has two intervals in the cell (1, 4). In that case, the cell will store two intervals for the same domain. An intervals is stored in the `CoInterval` class, Listing 1. Each cell stores a list of CoInterval instances. As an example of the data stored for a list take the cell (1, 4) containing the red domain in Figure 3.2.

```
class CoCell {
  // The list of intervals inside the current
  cell std::vector<CoInterval> m_intervals;
  // ...
};

class CoInterval {
  // The domain related to the current interval
  int m_domainId;
  // The position of the first element of the element-list
  int m_beginingOfInterval;
  // The number of elements in the current interval
  int m_nbElementsInInterval;
  // ...
};
```

Listing 1: Definition of the CoCell and CoInterval class used in the cell list algorithm implemented in Complexes++ . The comment "..." indicates other implementation details.

To calculate the completely potential energy between all domains we iterate over all cells in the cell-list, see Algorithm 2. For each cell we then iterate over the list of CoIntervals. During a Monte Carlo move only a single domain will be moved. We therefore do not need to recalculate the complete potential energy. Instead we need iterate over the cells occupied by the moved domain and update the energy difference appropriately, replacing the outer most loop in Algorithm 2 with a loop over only the cell currently occupied by the domain. The last step further reduces the computational effort to calculate energies during a simulation.

The interface to the cell list algorithm to calculate pair interactions has also been implemented without references to the underlying algorithm and can be swapped out with different algorithms. In Complexes++ , there are two data-structures available for the cutoff grid. The *dense* data structure is allocated cell objects for all cells in a simulation box and offers efficient computation of neighboring cells. The memory consumption of this data structure scales with $(L_{\mathrm{Box}}/L_{\mathrm{Cell}})^3$, where L_{Box} is the edge-length of the simulation box and L_{Cell} is the edge-length of a cell, see Figure 3.3. This data structure is optimal for dense simulations. The *sparse* data structure uses a hash-map to only allocate cells that

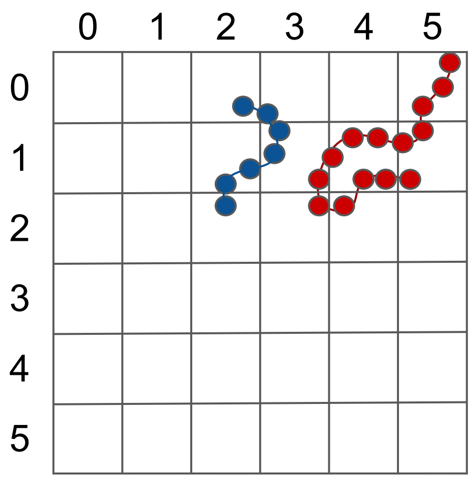

Figure 3.2: Example configuration for two domains (blue and red) in a 2D grid. Numbers on the top and left are used to index cells.

ALGORITHM 2: Cell list algorithm to calculate all pair interactions.

```
function computeEnergy(Cells[K], Domains[M])
    energy = 0
    // Compute energy (particle to particle interactions)
    for cell in Cells do
        for target in cell do
            for source in cell do
                if source.domainId ≠ target.domainId then
                    target_dom = Domains[target.domainId]
                    source_dom = Domains[source.domainId]
                    energy += kernel(target_dom, source_dom, target, source)
                end
            end
            for nb_cell in cell.neighbors() do
                for source in nb_cell do
                    if source.domainId ≠ target.domainId then
                        target_dom = Domains[target.domainId]
                        source_dom = Domains[source.domainId]
                        energy += kernel(target_dom, source_dom, target,
                          source)
                    end
                end
            end
        end
    end
    return energy
```

are occupied by beads. The memory consumption of this data structure scales with the number of beads in a simulation and is independent of the box size. It is suited for sparse simulations. The high-level interface is also agnostic to the underlying cell-list algorithm allowing to chose a completely different algorithm if desired.

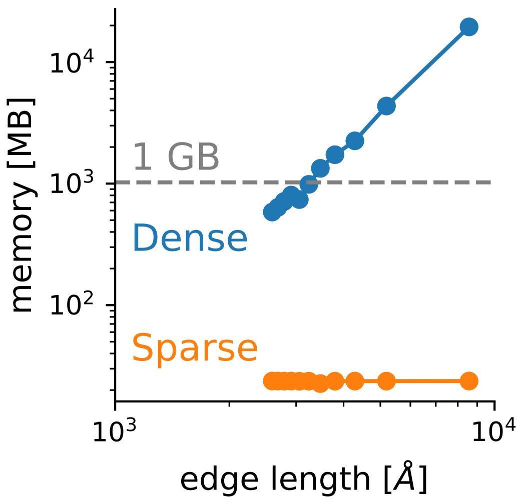

Figure 3.3: Memory consumption of the dense (blue) and sparse (orange) cell-list data-structure with a constant number of beads. The cutoff is set to 12 Å. The gray line marks one gigabyte.

3.10 Task based parallelism

Monte Carlo algorithms have two points which can be parallelized, the energy calculation and the trial move generation. The trial move generation in complexes is not computationally expensive and a minuscule amount of time is spend on it. Leaving the energy calculation for the single replica simulations. In replica exchange Monte-Carlo all replicas can be executed in parallel with few synchronization points to exchange coordinates. Ideally, the single replica and replica exchange simulation can use the same parallelisms scheme. It has to be taken into account that replicas can be unbalanced or change during a simulation if a phase transition occurs (from liquid to solid). Another requirement was that it should be possible to always use the maximal number of available central processing unit (CPU) cores independent of the number of replicas. Meaning a single replica can use multiple threads and in the case that more replicas than available threads are simulated the different replicas have a scheduler queue.

In Complexes++ this problem is solved using the task application programming interface (API) in OpenMP [57]. Tasks are generated based on either domains or cell lists (if activate). Meaning tasks are small and plentiful. At the beginning of the energy calculation, the number of tasks is distributed equally to the available threads. If a thread finishes early it can then steal tasks from other threads. Allowing threads to be always busy. It balances when different replicas have different work loads. Nothing special is done to minimize the numa effect. Therefore, it is maybe better to one process per memory node (explicit sync) and threads inside.

The replica simulations can run in parallel. To make better use of available resources that allow multi-node jobs also a message passing interface (MPI) implementation has been developed. There is no work stealing between replicas in this configuration.

3.11 CPLX File Format

The input for Complexes++ simulation is stored in a CPLX file similar to the tpr files in GROMACS. The CPLX file contains all information about the domains and pair kernels to simulate, with the exception of the parameters for the Monte-Carlo algorithm and the used forcefield parameters. The CPLX file format is based on YAML [28] and divided into four sections *box, definitions*, *topologies*, and *forcefield*.

The *box* section contains the dimensions of the rectangular simulation box as a YAML list. The length in each dimension is given in Å.

The *definitions* section defines the type of domains that are used in the simulation and the pair kernels used to calculate the energy between domains. The *definitions* section is separated into two sub sections to define the final domain-types and their interactions between them, called *domains* and *pair-interactions* respectively. The *domains* section contains a dictionary with the entry names being the domain names and the definition for the domain. A definition contains the type of move the domain can make, currently only rigid is implemented, and the parameters for the move. Allowing to define several rigid bodies in a simulation that have different rotation and translation parameters. The definitions can be used for systems with a mixture of large and small rigid bodies to model the corresponding mobility. The *pair-interaction* section defines which pair interaction potential to choose for each combination of domain types defined in *domains*. In the definition, one can define more domain types than what will be used in a simulation. Allowing to create standard definitions (like the KH forcefield) for different applications. See Listing 2 for an example definition using two domain types **A** and **EM**. Here the **EM** type is set to not move at all and interacts with **A** domains using the WCA potential. This definition could be used to fit any domain of type **A** into a domain defined by **EM**.

```
definitions:
  domains:
    A:
      defaults: {rotation: 2, translation: 1}
      move: rigid
   EM:
      defaults: {rotation: 0, translation: 0}
   move: rigid
pair-interaction:
  - domain-type-pair: [A, EM] function: WCA
  - domain-type-pair: [A, A] function: LJH
  - domain-type-pair: [EM, EM] function: None
```

Listing 2: Complexes++ domain definitions for a simulation with two domain types **A** and **EM**. Both domain types move as rigid bodies but they have different pair interaction potentials with each other.

The *topologies* section defines the actual domains that are used in the simulation. It consists of a YAML list of topology entries. Each topology entry contains domain definitions and connections between domains of the same topology. A domain contains the following field: beads, chain-ids, charges, coordinates, nbeads, type, mc-moves, meta-data, name. Here the type field specifies the domain type, which has to be defined in the definitions section. Domains have to be numbered consecutively and uniquely across all topologies.

The *forcefield* section contains definitions for the available bead types as well as the energy and diameter pair parameters.

Chapter 4

pycomplexes toolbox

```
box: [100, 100, 100]
topology:
  protein-name:
    coordinate-file: structure.pdb
    move: true
    domains:
      A:
        type: rigid
        selection: 'namd CA and segid A'
      link:
        type: gaussian
        selection: 'name CA and segid L'
        start_connection: [A, 'segid A and resid 10']
        end_connection: [B, 'segid B and resid 10']
      B:
        type: rigid
        selection: 'name CA and segid B'
```

Listing 3: TOP file for a simulation for two rigid domains connected by a Gaussian domain. All selections are written in the atom selection language used by MDAnalysis.

pycomplexes is a Python library and CLI program that includes several tools to help setup simulations for Complexes++ and analyze them later. The CPLX format accepted by Complexes++ is very complex and allows the user a lot of freedom in setting up the simulation. A lot of simulation setups do not need to leverage the full flexibility of complexes and therefore pycomplexes includes a tool called `convert` that takes as input a simplified format, called a TOP file, for describing simulations and generating CPLX files from it. As part of the conversion, the script will automatically choose charges and interaction energies based on amino acid type. As interaction energies, the user can choose either the KH model or the unaltered MJ model. The charges are set according to the KH model in both cases. The interaction potential is chosen based on domain type, so far allowed are rigid and gaussian. The rigid domain type uses the modified lennard jones potential, eq 2.1. For the gaussian domains positions of the C_α beads will be stored in the CPLX as rigid domains that do not move and a connection potential, eq 2.8, will be used between the two rigid domains connected by the gaussian domain. To define a domain in the top file the type and a selection of beads have to be specified. For the gaussian domain, in addition, the beads of the rigid domains connected by the gaussian domain have to be specified as well. The convert script could ignore known beads for a gaussian domain in the structure but keeping the information in the CPLX file allows to generate explicit linker positions in the post processing, i.e. with the `addlinker` tool.

The structures and selections are read and parsed using MDAnalysis[58, 59]. In the molecular dynamics community there exists a large variety of structure file format and they are often not well defined or popular programs write and accept ill formatted files, MDAnalysis helps to read a large variety of different formats with an easy to understand atom selection language.

In addition to the `convert` command pycomplexes also include a variety of other commands to help the user setup and visualize simulations. As of the time of writing the other implemented commands are

- `equilibration` Update coordinates stored in a CPLX from a trajectory.
- `forcefield` Convert a forcefield using scaling parameters λ and e_0, see eq 2.2.
- `visualize` Create VMD scripts from simulations.
- `demux` generate input files for the GROMACS tool `trjcat` to generate time-continuous trajectories from replica exchange simulations.
- `addlinker` generate explicit linker conformations for a simulation with Gaussian linkers.

Bibliography

[1] Young C. Kim and Gerhard Hummer. Coarse-grained Models for Simulations of Multiprotein Complexes: Application to Ubiquitin Binding. *J. Mol. Biol.*, 375(5): 1416–1433, 2008. ISSN 00222836. doi: 10.1016/j.jmb.2007.11.063.

[2] Fangqiang Zhu and Bo Chen. Monte Carlo Simulations of HIV Capsid Protein Homodimer. *J. Chem. Inf. Model.*, 55(7):1361–1368, 2015. ISSN 15205142. doi: 10.1021/acs.jcim.5b00126.

[3] Hao Sha and Fangqiang Zhu. Parameter Optimization for Interaction between C-Terminal Domains of HIV-1 Capsid Protein. *J. Chem. Inf. Model.*, 57(5):1134–1141, 2017. ISSN 15205142. doi: 10.1021/acs.jcim.7b00011.

[4] Young C. Kim and Jeetain Mittal. Crowding induced entropy-enthalpy compensation in protein association equilibria. *Phys. Rev. Lett.*, 110(20):1–5, 2013. ISSN 00319007. doi: 10.1103/PhysRevLett.110.208102.

[5] Young C. Kim, Robert B. Best, and Jeetain Mittal. Macromolecular crowding effects on protein-protein binding affinity and specificity. *J. Chem. Phys.*, 133(20), 2010. ISSN 00219606. doi: 10.1063/1.3516589.

[6] J Rosen and Young C. Kim. Modest Protein - Crowder Attractive Interactions Can Counteract Enhancement of Protein Association by Intermolecular Excluded Volume Interactions. *J. Phys. Chem. B*, I:2683–2689, 2011.

[7] Gregory L. Dignon, Wenwei Zheng, Young C. Kim, Robert B. Best, and Jeetain Mittal. Sequence determinants of protein phase behavior from a coarse-grained model. *PLoS Comput. Biol.*, 14(1):1–23, 2018. ISSN 15537358. doi: 10.1371/journal.pcbi.1005941.

[8] Tatjana Skrbić, Pietro Faccioli, and Cristian Micheletti. The role of non-native interactions in the folding of knotted proteins: insights from molecular dynamics simulations. *PLoS Comput. Biol.*, 8(6):1–12, 2012. ISSN 2218273X. doi: 10.3390/biom4010001.

[9] Silvio a Beccara, Tatjana Škrbić, Roberto Covino, Cristian Micheletti, and Pietro Faccioli. Folding Pathways of a Knotted Protein with a Realistic Atomistic Force Field. *PLoS Comput. Biol.*, 9(3), 2013. ISSN 1553734X. doi: 10.1371/journal.pcbi.1003002.

[10] Anshul Sirur and Robert B. Best. Effects of interactions with the groel cavity on protein folding rates. *Biophys. J.*, 104(5):1098–1106, 2013. ISSN 00063495. doi: 10.1016/j.bpj.2013.01.034.

[11] Roberto Covino, Tatjana Skrbić, Silvio A. Beccara, Pietro Faccioli, and Cristian Micheletti. The role of non-native interactions in the folding of knotted proteins: insights from molecular dynamics simulations. *Biomolecules*, 4(1):1–19, 2014. ISSN 2218273X. doi: 10.3390/biom4010001.

[12] Anshul Sirur, David De Sancho, and Robert B. Best. Markov state models of protein misfolding. *J. Chem. Phys.*, 144(7), 2016. ISSN 00219606. doi: 10.1063/1.4941579.

[13] Roberto Covino. *Investigating Protein Folding Pathways at Atomistic Resolution : from a Small Domain to a Knotted Protein.* Phd, UNIVERSITÀ DEGLI STUDI DI TRENTO, 2013.

[14] Bartosz Różycki, Young C. Kim, and Gerhard Hummer. SAXS Ensemble Refinement of ESCRT-III CHMP3 Conformational Transitions. *Structure*, 19(1):109–116, 2011. ISSN 09692126. doi: 10.1016/j.str.2010.10.006.

[15] Jürgen Köfinger, Michael J. Ragusa, Il-Hyung Lee, Gerhard Hummer, and James H. Hurley. Solution Structure of the Atg1 Complex: Implications for the Architecture of the Phagophore Assembly Site. *Structure*, 23(5):809–818, 2015. ISSN 09692126. doi: 10.1016/j.str.2015.02.012.

[16] A. Baumlova, D. Chalupska, B. Rozycki, M. Jovic, E. Wisniewski, M. Klima, A. Dubankova, D. P. Kloer, R. Nencka, T. Balla, and E. Boura. The crystal structure of the phosphatidylinositol 4-kinase II. *EMBO Rep.*, 15(10):1085–1092, 2014. ISSN 1469-221X. doi: 10.15252/embr.201438841.

[17] Bartosz Rózycki, Marek Cieplak, and Mirjam Czjzek. Large conformational fluctuations of the multi-domain xylanase Z of Clostridium thermocellum. *J. Struct. Biol.*, 191(1):68–75, 2015. ISSN 10958657. doi: 10.1016/j.jsb.2015.05.004.

[18] Dominika Chalupska, Andrea Eisenreichova, Bartosz Różycki, Lenka Rezabkova, Jana Humpolickova, Martin Klima, and Evzen Boura. Structural analysis of phosphatidylinositol 4-kinase IIIβ (PI4KB) – 14-3-3 protein complex reveals internal flexibility and explains 14-3-3 mediated protection from degradation in vitro. *J. Struct. Biol.*, 200(1):36–44, 2017. ISSN 10958657. doi: 10.1016/j.jsb.2017.08.006.

[19] Kei Ichi Okazaki, Takato Sato, and Mitsunori Takano. Temperature-enhanced association of proteins due to electrostatic interaction: A coarse-grained simulation of actin-myosin binding. *J. Am. Chem. Soc.*, 134(21):8918–8925, 2012. ISSN 00027863. doi: 10.1021/ja301447j.

[20] Duccio Malinverni, Alfredo Jost Lopez, Paolo De Los Rios, Gerhard Hummer, and Alessandro Barducci. Modeling Hsp70/Hsp40 interaction by multi-scale molecular simulations and coevolutionary sequence analysis. *Elife*, 6:1–20, 2017. ISSN 2050084X. doi: 10.7554/eLife.23471.

[21] Krishnakumar M. Ravikumar, Wei Huang, and Sichun Yang. Coarse-grained simulations of protein-protein association: An energy landscape perspective. *Biophys. J.*, 103(4):837–845, 2012. ISSN 00063495. doi: 10.1016/j.bpj.2012.07.013.

[22] Shilpa Yadahalli, V. V. Hemanth Giri Rao, and Shachi Gosavi. Modeling Non-Native Interactions in Designed Proteins. *Isr. J. Chem.*, 576104, 2014. ISSN 00212148. doi: 10.1002/ijch.201400035.

[23] Nicole Fortoul, Pankaj Singh, Chung Yuen Hui, Maria Bykhovskaia, and Anand Jagota. Coarse-grained model of SNARE-mediated docking. *Biophys. J.*, 108(9): 2258–2269, 2015. ISSN 15420086. doi: 10.1016/j.bpj.2015.03.053.

[24] Bartosz Różycki, Pierre-André Cazade, Shane O'Mahony, Damien Thompson, and Marek Cieplak. The length but not the sequence of peptide linker modules exerts the primary influence on the conformations of protein domains in cellulosome multi-enzyme complexes. *Phys. Chem. Chem. Phys.*, 19(32):21414–21425, 2017. ISSN 1463-9076. doi: 10.1039/C7CP04114D.

[25] M Fortoul, Maria Bykhovskaia, and Anand Jagota. Coarse-Grained Model of the SNARE Complex Shows that Quick Zippering Requires Partial Assembly. *bioRxiv*, 2018. doi: 10.1101/294181.

[26] Brandon G Horan, Dimitrios Vavylonis, and Young C Kim. Computational modeling highlights disordered Formin Homology 1 domain's role in profilin-actin transfer. *bioRxiv*, 2018. doi: 10.1101/263566.

[27] Bartosz Rozycki and Marek Cieplak. Stiffness of the C-terminal disordered linker affects the geometry of the active site in endoglucanase Cel8A. *Mol. BioSyst. Mol. BioSyst*, 12(I):1–2, 2016. ISSN 1463-9076. doi: 10.1039/C6MB00606J.

[28] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. http://yaml.org/spec/1.2/spec.html, 2018.

[29] Sanzo Miyazawa and Robert L. Jernigan. Residue – Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading. *J. Mol. Biol.*, 256(3):623–644, 1996. ISSN 00222836. doi: 10.1006/jmbi.1996.0114.

[30] Viktor Hornak, Robert Abel, Asim Okur, Bentley Strockbine, Adrian Roitberg, and Carlos Simmerling. Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters. *Proteins*, 65:712–725, 2006.

[31] Edward P. O'Brien, Greg Morrison, Bernard R. Brooks, and D. Thirumalai. How accurate are polymer models in the analysis of Förster resonance energy transfer experiments on proteins? *J. Chem. Phys.*, 130(12), 2009. ISSN 00219606. doi: 10.1063/1.3082151.

[32] Hiromi Yamakawa. *Modern Theory of Polymer Solutions*. Harper & Row, NewYork, 1971.

[33] Robert B. Best, Yng Gwei Chen, and Gerhard Hummer. Slow protein conformational dynamics from multiple experimental structures: The helix/sheet transition of Arc repressor. *Structure*, 13:1755–1763, 2005. ISSN 09692126. doi: 10.1016/j.str.2005.08.009.

[34] C. Hyeon, G. Morrison, and D. Thirumalai. Force-dependent hopping rates of RNA hairpins can be estimated from accurate measurement of the folding landscapes. *Proc. Natl. Acad. Sci.*, 105(28):9604–9609, 2008. ISSN 0027-8424. doi: 10.1073/pnas.0802484105.

[35] John Karanicolas and Charles L Brooks. The origins of asymmetry in the folding transition states of protein L and protein G. *Protein Sci.*, 11(10):2351–2361, 2002. ISSN 0961-8368. doi: 10.1110/ps.0205402.

[36] Jonathan E. Kohn, Ian S. Millett, Jaby Jacob, Bojan Zagrovic, Thomas M. Dillon, Nikolina Cingel, Robin S. Dothager, Soenke Seifert, P. Thiyagarajan, Tobin R. Sosnick, M. Zahid Hasan, Vijay S. Pande, Ingo Ruczinski, Sebastian Doniach, and Kevin W. Plaxco. Random-coil behavior and the dimensions of chemically unfolded proteins. *Proc. Natl. Acad. Sci. U. S. A.*, 101(34):12491–6, 2004. ISSN 0027-8424. doi: 10.1073/pnas.0403643101.

[37] Gustavo Fuertes, Niccolò Banterle, Kiersten M. Ruff, Aritra Chowdhury, Davide Mercadante, Christine Koehler, Michael Kachala, Gemma Estrada Girona, Sigrid Milles, Ankur Mishra, Patrick R. Onck, Frauke Gräter, Santiago Esteban-Martín, Rohit V. Pappu, Dmitri I. Svergun, and Edward A. Lemke. Decoupling of size and shape fluctuations in heteropolymeric sequences reconciles discrepancies in SAXS vs. FRET measurements. *Proc. Natl. Acad. Sci.*, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1704692114.

[38] Jianhui Song, Gregory Neal Gomes, Tongfei Shi, Claudiu C. Gradinaru, and Hue Sun Chan. Conformational Heterogeneity and FRET Data Interpretation for Dimensions of Unfolded Proteins. *Biophys. J.*, 113(5):1012–1024, 2017. ISSN 15420086. doi: 10.1016/j.bpj.2017.07.023.

[39] Wenwei Zheng, Alessandro Borgia, Karin Buholzer, Alexander Grishaev, Benjamin Schuler, and Robert B. Best. Probing the Action of Chemical Denaturant on an Intrinsically Disordered Protein by Simulation and Experiment. *J. Am. Chem. Soc.*, 138(36):11702–11713, 2016. ISSN 15205126. doi: 10.1021/jacs.6b05443.

[40] Alessandro Borgia, Wenwei Zheng, Karin Buholzer, Madeleine B. Borgia, Anja Schüler, Hagen Hofmann, Andrea Soranno, Daniel Nettels, Klaus Gast, Alexander Grishaev, Robert B. Best, and Benjamin Schuler. Consistent View of Polypeptide Chain Expansion in Chemical Denaturants from Multiple Experimental Methods. *J. Am. Chem. Soc.*, 138(36):11714–11726, 2016. ISSN 15205126. doi: 10.1021/jacs.6b05917.

[41] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Academic press, 2001.

[42] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953. ISSN 00219606. doi: doi: 10.1063/1.1699114.

[43] Roy J. Glauber. Time-Dependent Statistics of the Ising Model. *J. Math. Phys.*, 4(2): 294, 1963. ISSN 00222488. doi: 10.1063/1.1703954.

[44] Ulrich K. Deiters. Efficient coding of the minimum image convention. *Zeitschrift fur Phys. Chemie*, 227(2-3):345–352, 2013. ISSN 09429352. doi: 10.1524/zpch.2013.0311.

[45] Rh Swendsen and Js Wang. Replica Monte Carlo simulation of spin glasses. *Phys. Rev. Lett.*, 57(21):2607–2609, 1986. ISSN 1079-7114. doi: 10.1103/PhysRevLett.57.2607.

[46] Charles H. Bennett. Efficient estimation of free energy differences from Monte Carlo data. *J. Comput. Phys.*, 22(2):245–268, 1976. ISSN 10902716. doi: 10.1016/0021-9991(76)90078-4.

[47] Mark Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation.* Oxford University Press, 2010.

[48] Ulrich H. E. Hansmann. Parallel tempering algorithm for conformational studies of biological molecules. *Chem. Phys. Lett.*, 281(1-3):140–150, 1997. ISSN 00092614. doi: 10.1016/S0009-2614(97)01198-6.

[49] Giovanni Bussi. Hamiltonian replica exchange in GROMACS: a flexible implementation. *Mol. Phys.*, 112(3-4):379–384, 2014. ISSN 0026-8976. doi: 10.1080/00268976.2013.824126.

[50] Tsuneyasu Okabe, Masaaki Kawata, Yuko Okamoto, and Masuhiro Mikami. Replica-exchange Monte Carlo method for the isobaric-isothermal ensemble. *Chem. Phys. Lett.*, 335(5-6):435–439, 2001. ISSN 00092614. doi: 10.1016/S0009-2614(01)00055-0.

[51] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30, 1998.

[52] John D. Weeks, David Chandler, and Hans C. Andersen. Role of repulsive forces in determining the equilibrium structure of simple liquids. *J. Chem. Phys.*, 54(12): 5237–5247, 1971. ISSN 00219606. doi: 10.1063/1.1674820.

[53] Iris Antes. DynaDock: A now molecular dynamics-based algorithm for protein-peptide docking including receptor flexibility. *Proteins Struct. Funct. Bioinforma.*, 78(5):1084–1104, 2009. ISSN 08873585. doi: 10.1002/prot.22629.

[54] David R Yarkony. *Modern electronic structure theory Part II.* World Scientific Publishing, Singapore, 1995.

[55] Berenger Bramas. Inastemp : A Novel Intrinsics-as-Template Library for Portable SIMD-Vectorization. 2017:1–22, 2017. ISSN 10589244. doi: 10.1155/2017/5482468.

[56] https://en.wikipedia.org/wiki/AOS_and_SOA.

[57] OpenMP Architecture Review Board. OpenMP Application Program Interface Version 3.0, 2013. URL https://www.openmp.org/wp-content/uploads/OpenMP4.0.0.pdf.

[58] Richard J Gowers, Max Linke, Jonathan Barnoud, Tyler J E Reddy, Manuel N Melo, Sean L Seyler, Jan Domański, David L Dotson, Sébastien Buchoux, Ian M Kenney, and Oliver Beckstein. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. In *Proc. 15th Python Sci. Conf.*, pages 98–105, 2016.

[59] Naveen Michaud-Agrawal, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein. Software News and Updates MDAnalysis : A Toolkit for the Analysis of Molecular Dynamics Simulations. *J. Comput. Chem.*, 32(10):2319–2327, 2011. doi: 10.1002/jcc.

List of Abbreviations

API	application programming interface
CG	coarse-grained
CLI	command line interface
CPU	central processing unit
FRET	fluorescence resonance energy transfer
KH	Kim-Hummer
MD	molecular dynamics
MIC	minimum image convention
MJ	Miyazawa and Jernigan
MPI	message passing interface
PMF	potential of mean force
RNA	ribonucleic acid
RNG	random number generator
SAXS	small angle x-ray scattering
SIMD	single instruction multiple data
WCA	Weeks-Chandler-Anderson