Thesis for the title Bachelor of Science in Chemistry

Implementation of Auxiliary Restraints for Relative Binding Free Energy Calculations in the 'common-core serial-atom-insertion' Framework `Transformato`

Supervised by: Univ-Prof. Dr. Stefan Boresch
Submitted by: Alexander Grasser

September 2022

Department of Computational Biological Chemistry
University of Vienna

Abstract

The knowledge of relative (and absolute) binding free energy differences is essential to quantify the binding affinities of a series of ligands to a protein binding site of interest. Traditionally, calculating these energies is done using direct "alchemical scaling" from one molecule of interest to another. The 'common-core serial-atom-insertion' framework as implemented in `Transformato` instead allows to efficiently divide the problem into sequential fragments along mutation paths of actual atoms. However, complications may arise when ligands leave the binding site during simulation. This work presents a possible solution by enabling both automatic and manual addition of restraints to the system.

Notes

The data compiled for this thesis may be accessed at GitHub under `https://github.com/agrass15268/AuxiliaryRestraintsTransformato/tree/data` or with the QR - Code to the right.

Source code and packaged versions of the `Transformato` package may be retrieved from its GitHub repository at `https://github.com/wiederm/transformato` or with the QR - Code to the right.

Table of Contents

1 Introduction

Almost since the inception of computer-assisted molecular modeling, a significant focus has been on the interactions in biological protein-ligand complexes. Designing and testing lead compounds to catalyze or inhibit biochemical processes is a major focus of research and development efforts, and a comparatively easy-to-access parameter with nevertheless good predictive capabilities has long been the relative binding free energy difference (RBFE) $\Delta\Delta G_{binding}$, especially with recent advancements in computational power [1].

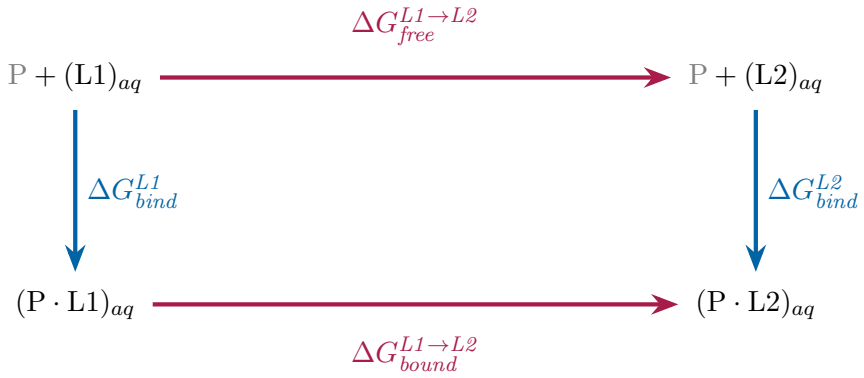


Figure 1.1: The traditional thermodynamic cycle[2] used to calculate the relative binding free energy difference $\Delta\Delta G_{binding}$ between two ligands L_1 and L_2 using Eq. 2.5. The physical path is marked in blue; the alchemical one in wine red.

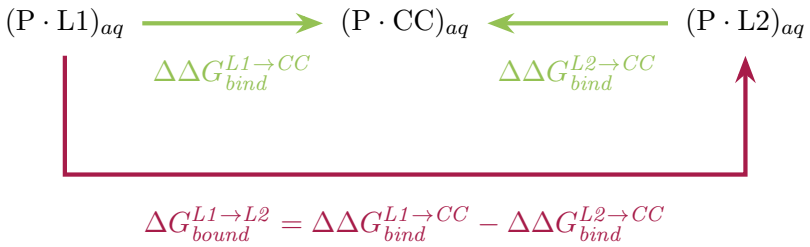


Figure 1.2: Alternative calculation pathway as used by `Transformato`. The computed free energy difference is no longer between bound and unbound complexes, but instead from the bound complexes to their common core.

Recently, the package `Transformato` was introduced by Karwounopoulos, Wieder, Braunsfeld, and Boresch [3–5], implementing a novel approach to compute $\Delta\Delta G_{binding}$. It bypasses issues with classical, λ-scaling methods utilizing the circle in Fig. 1.1, in

particular the need for dummy atoms at the endstates. Being a Python package, `Transformato` is easy to install, use, and modify to adapt to system-specific challenges or support different MD engines. Instead of full alchemical transformations of one compound into the other, it relies instead on transforming the two physical endpoints into a *common core* (CC). The free energy differences between the CC and the endstates then allow the calculation of the free energy difference between the endstates themselves (see Fig. 1.2).

With this method, atoms outside the common core are transformed into so-called *dummy atoms*, a term for particles that, while still present in the simulated system, no longer have nonbonded interactions and are only kept in place by weak bonded interactions. If handled properly, these do not contribute to the calculated free energy differences[6].

These dummy atoms are introduced sequentially through several intermediate states, alongside scaling of their nonbonded interactions.

In a post-processing step, the energy differences between these states are computed and processed with the *multistate Bennett Acceptance Ratio*[7] method (mBAR), an expanded version of Bennett's acceptance ratio method [8], to obtain the free energy difference between the respective endpoints. For (m)BAR to work, it is necessary that the distributions of energy differences between neighboring intermediate states, at least those of the respective nearest neighbors, overlap. Therefore, as a prerequisite, mBAR requires topological overlap between the various intermediate states - a problematic proposition if the ligand is likely to leave its binding site during the simulation. This is usually the case if repulsive interactions between the binding site and ligand exist that are not counteracted by an attractive force, and gets significantly more likely with increased simulation time. To prevent this, an additional module for `Transformato` called `restraints.py` was developed and integrated into `Transformato`. It allows the user to either request automatic restraints or define their own. As implemented, these restraints act on the center of mass of the selected ligand atoms, restraining them to their original positions relative to the protein, with variable force as defined by the user.

To suit a variety of sampling needs, both harmonic and flat-bottomed energy potentials may be used as the functional form for the restraints. With flat-bottoms, a ligand may be specifically allowed to sample the entire binding site without being constrained, yet never leave it. This allows for better sampling and longer runtimes even at increased temperatures, without fear of losing the ligand.

While a stringent validation of the methods implemented and coded by me was not possible within the time frame of this thesis, a number of qualitative simulations with a variety of testing parameters were conducted and evaluated, finding no significant impact of reasonable restraints on the calculated free energy differences.

2 Theory

2.1 Theoretical Background

Simulating the dynamics of (bio-)chemical processes (referred to as *Molecular Dynamics* (MD)) in their entirety can be divided into two classes: calculations utilizing *Quantum Mechanics* (QM), where interactions are calculated according to Schrödinger's Equation, and *Molecular Mechanics* (MM), where calculations instead use a simplified mass-charge-force model, the so-called force fields, to calculate potential energies. Combination methods (QM/MM) also exist, being routinely used to improve geometries and improve calculation precision, with QM/MM methods allowing for enhanced accuracy while limiting computational costs [9]. Importantly, QM also forms the basis for the force fields used in MD [10].

While there is no doubt that methods utilizing QM can deliver more accurate results (especially if using "pure" QM and not using semi-empirical methods), computational costs prohibit their use in large biological systems ("large" in this context referring to even just a single protein and its ligand) [1]. Thus, for free energy calculations of the sort `Transformato` is designed to accomplish, QM methods are unsuitable - which leaves MM.

2.1.1 Basics of Molecular Dynamics Modeling

Since the discovery of cells, the inner workings of biological systems have always been a mystery science has raced to solve. While, of course, physical methods have allowed some degree of monitoring biochemical processes, these have significant limits - most glaringly being limited to physically-present structures. With Molecular Dynamics (MD) on the other hand, it is entirely possible to observe molecular interactions as they would happen in reality - at the leisure and convenience of the observer, with an unmatched resolution and clarity, and without the need for any kind of wet chemistry - the so-called *in silico* approach [11]. In an MD system, the total potential energy is simply the sum of bonded and nonbonded potential energy terms, both of which are dependent on the relative positions ($\vec{r}$) of the atoms. Transformato uses the Charmm General Force Field[10]. In general, MD aims to simulate the movement of atoms in a molecular system over time, governed by the physical interactions present in the molecule [12]. Unlike QM, which tries to calculate interactions using various approximations or even solutions of the Schrödinger equation, MM simulations are by their very nature much more of an approximation. Forces are approximated by what are essentially the mechanical principles of Newton: Atoms are perfect spheres, with mass and charges, and all their interactions are defined by a set of force terms governing the interactions of specific types of atoms. As such, the kinetic energy E_{kin} of an MD system is simply the kinetic energy of its atom masses m_i at speed v_i, using the familiar equation from Newtonian mechanics (Eq. 2.1), while their potential energy is defined similarly by classically mechanical

potentials governing their interactions.

$$E_{kin} = \sum_i \frac{m_i v_i^2}{2} \tag{2.1}$$

Force Fields For almost all force fields, forces in MM simulations may be divided into two categories - *bonded* and *nonbonded* interactions. These are often used synonymously with *intramolecular* and *intermolecular* forces, but are far from the same. It is quite possible, and — especially in biological systems the rule more than the exception — that nonbonded but intramolecular interactions contribute a significant part to the protein structure, with electrostatic forces and the hydrophobic effect being especially important for protein folding and, thus, functionality [13].

Both bonded and nonbonded forces are generally given an explicit set of parameters for each atom type. For a set of atoms i, j, k, l and r_{ij} denoting the distance between two atoms i and j and a atom- and bond-type dependent force constant k, the CHARMM General Force Field[10] uses the following bonded terms:

- Bonds (dependent on the distance r between two atoms)

  $E_{bond}^{i,j} = k(r_{ij} - r_{eq})^2$ - a simple harmonic potential with the bond-specific force constant k and equilibrium distance r_{eq}.

- Angles (dependent on the angle ϕ across three atoms)

  $E_{angle}^{ijk} = k(\phi_{ijk} - \phi_{eq})^2$ - a simple harmonic potential with angle-specific force constant k, current angle ϕ_{ijk} and equilibrium angle ϕ_{eq}.

- Dihedrals (dependent on the torsion angle ψ across four atoms)

  $E_{dihedral}^{ijkl} = k(1 + cos(n\psi_{ijkl} - \delta))$ - a periodic function dependent on the current torsion angle ψ_{ijkl} with force constant k, periodicity n and offset δ.

- Impropers (also called *improper dihedrals*, the out-of-plane angle of a fourth atom with respect to the plane formed by the first three atoms)

  $E_{improper}^{ijkl} = k(\Psi_{ijkl} - \Psi_{eq})^2$ - a simple harmonic potential with improper dihedral force constant k, dependent on the improper angle Ψ_{ijkl} and its equilibrium value Ψ_{eq}.

- Urey-Bradley, an optional modification of the angle term

  $E_{Urey-Bradley}^{ik} = k(r_{ik} - r_{eq}^{ik})^2$ - a simple harmonic potential in the distance between the two outer atoms i and k of the angle formed by atoms i, j, and k. It depends on their distance r_{ik}, the Urey-Bradley force constant and the equilibrium distance r_{eq}^{ik}.

```
BONDS
CG1N1  CG2R51  375.00     1.4220 ! DCG, yxu, RNA
CG1N1  CG2R61  345.00     1.4350 ! 3CYP, 3-pyridine (PYRIDINE pyr-CN)

ANGLES
CG2R51 CG1N1  NG1T1    40.00    180.00 ! DCG, yxu, RNA
CG2R61 CG1N1  NG1T1    40.00    180.00 ! 3CYP, 3-pyridine (PYRIDINE pyr-CN)

DIHEDRALS
NG1T1  CG1N1  CG2R61 CG2R61  0.0100 2    0.00 ! CNP2, by ac_aa
NG1T1  CG1N1  SG311  CG321   0.0060 1    0.00 ! XCN, by ac_aa

IMPROPERS
CG2D1  CG331  NG2D1  HGA4      25.00  0     0.00 ! SCH1, xxwy
CG2D1  CG331  NG2P1  HGR52     18.00  0     0.00 ! SCH2, xxwy
```

Figure 2.1: Illustrative excerpt from a parameter file created by CGenFF with a few definitions for bonds, angles, dihedrals and impropers each.

The so-called nonbonded terms consist of van-der-Waals (vdW) and electrostatic interactions. The van-der-Waals force represents the non-polar attractive and repulsive forces between nonbonded atoms and is usually represented by a Lennard-Jones (LJ) potential as given in Eq.2.2, with ϵ_i, ϵ_j and σ_i, σ_j being atom-specific constants often referred to as "softness" and "interaction range" respectively. Atom types are combined by using the Lorentz-Berthelot combination rules (Eq. 2.3 [14, 15]) .

$$U_{LJ}(r_{ij}) = 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{2.2}$$

$$\sigma_{ij} = \frac{\sigma_i \cdot \sigma_j}{2}; \epsilon_{ij} = \sqrt{\epsilon_i \cdot \epsilon_j} \tag{2.3}$$

The electrostatic interactions on the other hand are represented by the Coloumb potential as given in Eq. 2.4, with q_i,q_j being the involved atomic partial charges and ϵ_0 being the dielectric constant.

$$U_{Coloumb}(r) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} \tag{2.4}$$

The values of the force constants, equilibrium bond lengths, angles, etc. are stored in *parameter files*, a short excerpt is shown in Fig. 2.1. Not all atom types participate in all types of bonded terms; both impropers and Urey-Bradley terms are optional.

As can be seen in Fig. 2.1, force-field parameter files are structured very simply: The first few columns define the "atom types" for which a given set of parameters is valid. The others define the numeric value of the various parameters for the potential; for simple bonds, for example, the first value denotes the force constant, and the second the equilibrium distance.

However, the equations describing bonded and nonbonded interactions do not offer the complete information we require. What we want is the movement of atoms as a function of time (thus, Molecular *Dynamics*). However, these potentials are a significant part of the way there. To simulate a MM system, all atoms are assigned random initial velocities. For each step, the forces resulting from bonded and nonbonded interactions are calculated and applied to these atoms. The atoms are then moved according to the laws of classical mechanics as dictated by their momentum, the forces acting upon them, and the timestep chosen. After updating the positions, the cycle is repeated. QM/MM differs from classical MM in the way the forces are calculated: the interactions are derived from quantum mechanical calculations rather than force fields; the integration of the equations of motion, however, is still carried out classically.

This procedure results in positions for each atoms for every timestep you calculate. By saving these into a single file, one gets a so-called *trajectory*, containing the positions of all atoms of the system as a function of time along the entire simulation. However, trajectories typically do not contain every single calculated position, but rather only every n-th step, as trajectory files would otherwise be prohibitively large.

2.1.2 Relative and absolute binding free energies

Calculating free energy differences from these trajectories requires yet additional steps. For our purposes, we need to differentiate two types of free energy differences:

- *absolute* free energy differences (ΔG_{bind}); Here, the free energy difference is between the ligand bound in complex, and the ligand in aqueous solution free to explore all other possible configurations.
- *relative* free energy differences ($\Delta\Delta G_{bind}^{A\to B}$): Here, the difference between two absolute binding affinities for two ligands L_1 and L_2 is computed.

Calculating relative binding free energies (RBFE) typically exploits thermodynamic cycles. In Fig. 2.2 you can see `Transformato`'s version of the cycle introduced in Fig. 1.1 to calculate relative *binding* free energy differences: two ligands L_1 and L_2 are in aqueous solution with the same Protein P. The quantity of interest is the relative binding free energy $\Delta\Delta G_{bind}$, which can easily be calculated by using the nonphysical - *alchemical* transformations (Eq. 2.5) marked with dotted lines in Fig. 2.2, substituting the physical paths with the alchemical transformations from $L_1 \to L_2$. As the process forms a cycle (Fig. 1.1), it is thus possible to calculate $\Delta\Delta G_{bind}$ without simulating the physical pathway. While it would be possible to calculate ΔG_{bind}^{L1} (the *absolute binding free energy*) directly, doing so increases complexity and is not recommended for many applications[16].

$$\Delta\Delta G_{bind} = G_{bind}^{L2} - G_{bind}^{L1} = G_{bound}^{L1\to L2} - G_{free}^{L1\to L2} \tag{2.5}$$

For the alchemical transformations, in principle, all that needs to be done is to replace one ligand bound to the protein with the other. Doing so directly, however, does not work (well). Calculating free energies requires *phase space overlap*. *Phase space* in this

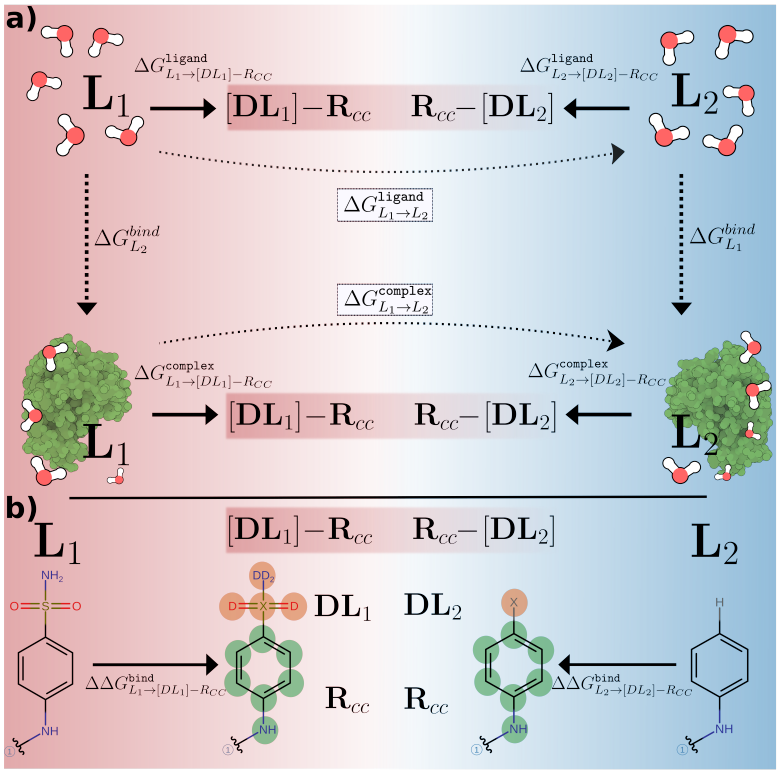


Figure 2.2: Overview of `Transformato`'s workflow
a) Comparison between alchemical transformations undertaken by traditional FES (dotted arrows) and the approach taken by `Transformato` (bold arrows). b) Example mutation path taken to transform two simple molecules into their common core. Green represents the common core, red those atoms that undergo mutation effects or are transferred into dummy atoms. Graphic from [3], used with permission.

context refers to the total allowed set of positions and momenta of the system. Two completely different ligands will overlap barely, if at all, resulting in error bars bigger than the free energy difference one wants to estimate. Thus, the amount of change in a single calculation needs to be reduced by introducing *intermediate states*. Free energy differences are then not calculated between the endstates, but rather as the sum of free energy differences between the endstates (Eq. 2.6). This allows for much more overlap between the individual states, leading to much smaller margins of error in total.

$$\Delta A_{1,K} = \sum_{i=1}^{K-1} \Delta A_{i,i+1} \tag{2.6}$$

The same considerations apply to the calculation of relative *solvation* free energy differences. Here, the "physical" path leads from an unsolvated compound to a solvated

one, whereas the alchemical pathway again requires changing the first into the second compound.

2.1.3 Generation of intermediate states and dummy atoms

The simplest method of generating intermediate states as used in Eq. 2.6 is to linearly "mix" the potentials of the endstates 0 and 1 U_λ from $U_0 \rightarrow U_1$, with λ being known as the *coupling parameter*. Notably, λ only affects a subset of potential energy terms (usually nonbonded interactions, i.e., charges and Lennard-Jones terms in the region which is different between the two endstates), while most of the potential energy terms remain unaffected (these are referred to as "environment"). The total potential for a given intermediate state i as required by Eq. 2.6 is thus given by Eq. 2.7.

$$U_\lambda^{total} = (1-\lambda)U_0 + \lambda U_1 + U_{environment} \tag{2.7}$$

Such linear scaling, however, creates a number of problems; most pressingly a distinct lack of phase space overlap required for analysis because of the so-called van-der-Waals endpoint problem (see section 2.1.5). `Transformato` does *not* use these kinds of "λ-scaling" methods, instead opting for an approach called "serial atom insertion" (see section 2.2.1).

An even more egregious problem arises when the number of atoms changes between endstates: The statistical mechanics underlying MD does not allow for changing the number of particles in the system. To sidestep this problem, *dummy atoms* are introduced by turning off all interactions except their bonded terms - either just for surplus atoms (known as the *single topology approach*) or by having both topologies fully present in all states, with the currently unused one treated as dummy atoms (known as the *dual topology approach*). It has been shown that, used correctly and provided they remain well-defined in position and orientation, their influence on calculated energies is zero as their terms in $\Delta\Delta A$ cancel each other out[6]. This allows the system to retain a constant number of particles.

2.1.4 Intermediate state analysis, post-processing and calculation of free energy

But one question has so far remained unanswered: how to actually obtain the free energy differences $\Delta A_{i,i+1}$ appearing in Eq. 2.6? FES conducted by the methods above typically rely on either Thermodynamic Integration (TI) [17] or Bennett's Acceptance Ratio [7, 8] to analyse the results from the various intermediate states. Both start from Eq. 2.8

$$\Delta A_{i,j} = -k_B T \cdot \ln \frac{Q_i}{Q_j} \tag{2.8}$$

which expresses the free energy difference between two states i, j as the logarithm of the ratio of their *partition functions* Q_i/Q_j.

The partition function in the so-called canonical ensemble depends on the set of possible states the system can have while exchanging energy with the outside world in a

state of thermal equilibrium. For a single molecules, this consists of exponential terms for all possible *quantum states* (Eq. 2.9, with q the single - molecule particle function and ϵ_i the state energy), for a system its the sum of all possible *energy states*[15]. For this thesis, all simulations were conducted using an **NPT** ensemble, meaning the **number** of particles, and the average **pressure** and **temperature** of the system were held constant.

$$q = \sum_{i=states}^{\infty} e^{-\frac{\epsilon_i}{k_B T}} \tag{2.9}$$

Eq. 2.10 is the partition function for the *canonical* or NVT ensemble (the derivation for the NPT ensemble follows the same pattern, but gives Gibbs instead of Helmholtz free energies). In classical statistical mechanics the (configurational) partition function of the canonical ensemble is defined as

$$Q = \int_{\Gamma} e^{-\frac{U(\vec{q})}{k_B T}} d\vec{q} \tag{2.10}$$

As can be seen, the function integrates over the entire phase space volume Γ (in mathematical terms, this is equivalent to the integral over all possible coordinates and momenta). In the following $\langle X \rangle$ represents the statistical expectation value in the canonical ensemble, as given by Eq. 2.11 (N_i representing the samples taken of the ensemble). For the ensembles used by MD, the *ergodic hypothesis* is assumed to be true, meaning that over time, an ensemble will include all possible configurations and furthermore, that a time-average of a representative part of the ensemble will be equivalent to the actual, ideal *ensemble average*[15].

$$\langle X \rangle = \lim_{t \to \infty} \frac{1}{t} \int_0^1 X(t) dt = \lim_{N_i \to \infty} \frac{1}{N_i} \sum_{i=1}^{N_i} X_i \tag{2.11}$$

Thermodynamic Integration Of the two methods, TI is the older one. It relies on numerical quadrature of the integral of the expectation value of the partial derivation of the potential energy with respect to the coupling factor as shown in Eq. 2.12 [18]. However, as we cannot calculate the integrand in a continuous manner for $\lambda = [0 \to 1]$, a numeric approximation as shown in Eq. 2.13 is required, where the difference of free energy is given as the sum of the integrand evaluated at discrete states K, each weighted with the weight factor w_k. The w_k may be computed a number of different ways; the commonly recommended all-purpose method is the trapezoidal method, despite significant drawbacks[16, 18]. The weighting method should be choosen carefully, as it has significant impact on result quality.

$$\Delta A = \int_0^1 \left\langle \frac{dU(\lambda, \vec{q})}{d\lambda} \right\rangle_{\lambda} d\lambda \tag{2.12}$$

$$\Delta A \approx \sum_{k=1}^{K} w_k \left\langle \frac{dU(\lambda, \vec{q})}{d\lambda} \right\rangle_{k} \tag{2.13}$$

While the calculation is rather simple, there are a some drawbacks to TI. First, it requires the computation of $\frac{dU}{d\lambda}$ for all configurations, which requires a MD engine with facilities capable of the appropriate derivations. Secondly, TI does badly with sudden steps in the integral, requiring either a highly efficient numeric integrator, or the use of techniques increasing phase space overlap (such as soft-core potentials); otherwise, large uncertainties/errors will arise. It should be noted that this, alongside a lack of a continuous, well-defined λ, makes it unsuitable for the Serial-Atom-Insertion approach used in `Transformato` (see section 2.2.1).

Bennett's Acceptance Ratio BAR and its multistate extension mBAR on the other hand do not face such limitations, at the cost of significantly increased complexity. First a set of $K \times K$ weighting functions $\alpha_{i,j}(\vec{q})$ are generated for the phase space overlap. The actual derivation then starts from Eq. 2.8 (just as that of TI), but by introducing the weighting factor, an equivalency between Q_i and Q_j can be expressed (Eq. 2.14).

$$Q_i \langle \alpha_{i,j} e^{-\beta U_j} \rangle_i = Q_j \langle \alpha_{i,j} e^{-\beta U_i} \rangle_j \tag{2.14}$$

Introducing the empirical estimator $\langle g \rangle_i = N_i^{-1} \sum_{n=1}^{N_i} g(\vec{q_i}, n)$ allows to rewrite the expectation values in Eq. 2.14 as sums (Eq. 2.15). Through the use of a *numerical bridge integrator* [19] it is possible to arrive at the free energy equation 2.16. Notably, this equation formally gives a *single* free energy instead of a free energy difference; however the presence of an undefined additive constant means that all calculations need to be put in reference to the free energy of one of the states (typically the starting point), thus again leading to a free energy difference.

$$\sum_{j=1}^{K} \frac{\hat{Q}_i}{N_i} \sum_{n=1}^{N_i} \alpha_{i,j} e^{(-\beta U_j(\vec{q_i}, n))} = \sum_{j=1}^{K} \frac{\hat{Q}_j}{N_j} \sum_{n=1}^{N_j} \alpha_{i,j} e^{(-\beta U_i(\vec{q_j}, n))} \tag{2.15}$$

$$\hat{A}_i = -\beta^{-1} \ln \sum_{j=1}^{K} \sum_{n=1}^{N_j} \frac{e^{(-\beta U_i)}}{\sum_{k=1}^{K} N_k e^{\beta \hat{A}_k - \beta U_k}} \tag{2.16}$$

Unlike TI, BAR and mBAR are able to give accurate results even with significant changes between states K and correspondingly little phase space overlap. Notably, it is able to deal with the significant changes caused by the Serial-Atom-Insertion method used by `Transformato` and was, thus, used in this work to calculate the free energy differences of interest.

2.1.5 Soft-core potentials and the van-der-Waals endpoint problem

In a standard MD simulation utilizing λ-scaling, the Lennard-Jones potential U_{LJ} is given by formula 2.17, introducing the coupling factor into the standard vdW-potential given by Eq. 2.2. This usually works rather well - the shape of the LJ potential ensures that no two particles inhabit the same spot. However, free energy calculations, for the reasons discussed in section 2.1.3, usually require the presence of *dummy atoms* - which

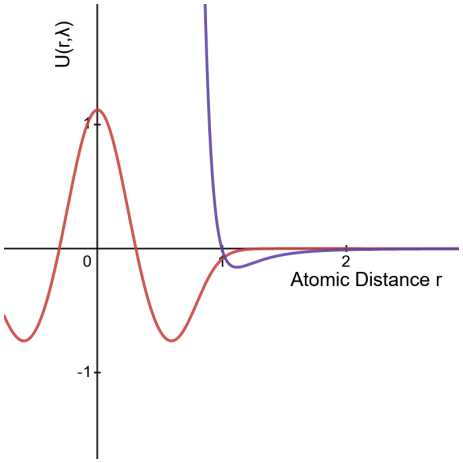


Figure 2.3: Comparison between behavior of standard Lennard-Jones (violet) and soft-core potential (red) for the same value of λ as they approach $r = 0$. Visibly: Compared to the soft-core potential, which remains defined throughout, the standard LJ potential has limited well size and diverges with extremely steep walls as $r \to 0$.

obviously do not have Lennard-Jones interactions. It's here where problems may now occur - as there are no LJ interactions, atoms may move freely into each other. This may not seem like such a significant problem - but forces still need to be calculated. Should particles manage to inhabit the exact same position, this would result in a division by zero - problematic, but most modern MD software can handle this.

Both more problematic and more common, however, is the case that two particles get very close to each other. This is especially common in simulations with linear coupling between λ and the LJ potential (Eq. 2.17), as for low values of λ the interaction range may decrease to such an extent that a particle may skip the entire "well" of the potential, and directly encounter its "wall", leading to extremely high forces being applied [20, 21]; see Fig. 2.3.

$$U_{LJ}(r_{ij}) = \lambda\, 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] \tag{2.17}$$

In this case, (i) the derivative $\langle \frac{\delta U}{\delta \lambda} \rangle$ may diverge, and (ii) because of the sudden change in energy the entire simulation may become unstable. This is known as the *van-der-Waals endpoint problem* (or *-catastrophe*, for the more dramatically inclined).

To avoid this problem, soft-core potentials dependent on the coupling factor λ were introduced as shown in Equation 2.18[21], with δ a positive constant (typically, $\delta = 5$ Å^2):

$$U_{LJ}^{SC}(r, \lambda) = \lambda\, 4\epsilon_i j \left[\frac{\sigma_{ij}^{12}}{(r_{ij}^2 + (1-\lambda)\delta)^6} - \frac{\sigma_{ij}^{6}}{(r_{ij}^2 + (1-\lambda)\delta)^3} \right] \tag{2.18}$$

These mostly solve the problem, as there can now be no division by zero and the fractions remain finite in all circumstances. They can, however, be computationally expensive -

especially for MD engines lacking native support for them[22]. It is important to note that `Transformato` for this reason does **not** use soft-core potentials, but rather the "serial-atom-insertion" approach (see section 2.2.1).

2.2 The common-core serial-atom-insertion framework `Transformato`

To avoid these problems and to reduce the general computational cost of FES, the common-core serial-atom-insertion framework `Transformato` was conceptualized and implemented[3–5]. Currently, while, in principle usable with any MD engine, `Transformato` only generates input scripts for openMM[23] and CHARMM[24]. Within these limitations, however, it allows straightforward calculation of relative binding and solvation free energy differences in the form of a single, easy-to install Python package.

2.2.1 Theoretical Principles

`Transformato` is built on two distinct theoretical pillars: The *Serial Atom Insertion* approach to generating intermediate states, and the *Common-Core* approach to handling thermodynamic cycles, together referred to as the "serial-atom-insertion common-core" method (SAI/CC).

Serial Atom Insertion (SAI)[21] refers to an alternative method of avoiding the van-der-Waals-Endpoint problem without relying on the soft-core potentials discussed in section 2.1.5, developed by Boresch and Bruckner in 2011. This allows for a significant decrease of computational cost in MD engines without native, GPU-assisted soft-core modeling, at the drawback of having significant, stepwise changes to the states present in phase space, making it unsuitable for analysis via Thermodynamic Integration. Central to this method is to turn off the changed atoms in a successive fashion, rather than scaling all interactions as one λ-dependent potential, to generate the intermediate states. Thus, rather than depending on a continuous coupling parameter λ to scale interactions (Eq. 2.19), the potential energy of the system is calculated by each atom turned on or off being its own intermediate state, with no intermediate scaling of the nonbonded parameters (Eq. 2.20) The two approaches are contrasted in Eqs. 2.19 and 2.20 for the case of a molecule consisting of m atoms surrounded by M Lennard-Jones particles; $u_{i,j}^{LJ}$ denotes the Lennard-Jones interaction between two particles.

$$U(\lambda) = \sum_{1 \leq i < j \leq M} u_{i,j}^{LJ} + \lambda \sum_{1 \leq i \leq M} \sum_{1 \leq l \leq m} u_{i,j}^{LJ} \qquad \lambda = [0, 1] \tag{2.19}$$

$$U(k) = \sum_{1 \leq i < j \leq M} u_{i,j}^{LJ} + \sum_{1 \leq i \leq M} \sum_{1 \leq l \leq k} u_{i,l}^{LJ} \qquad k = [0, m] \tag{2.20}$$

The term *Common Core* [4, 5] refers to the alternative alchemical path taken by `Transformato`. Whereas traditional FES transform the endstates directly into each other, `Transformato` instead transforms each of them into their maximum common topology - the *Common Core* (Fig. 2.2).

Routing via the common core is useful, as the complete free energy difference between two ligands on the alchemical pathway (illustrated in Fig. 1.1) can be expressed as the sum of the individual free energy differences (Eq. 2.21).

$$A_{bound}^{L_1 \to L_2} = \Delta\Delta G_{bound}^{L_1 \to L_2} = \Delta\Delta G_{bound}^{L_1 \to [D_{L_1}] - R_{CC}} - \Delta\Delta G_{bound}^{L_2 \to [D_{L_2}] - R_{CC}} \tag{2.21}$$

To reach the common core, the *mutation path* is generated by `Transformato` starting from each ligand L_1, L_2 to the CC R_{CC}. Atoms not belonging to the CC are decoupled and transformed into dummy atoms. First, all their electrostatic interactions are turned off, followed by the Lennard-Jones interactions of first all hydrogens simultaneously, then, separately, of all non-hydrogen atoms successively. The last atom connecting the *dummy regions* DL_1, DL_2 with the CC is called the *terminal junction* atom X. Interaction terms involving X are scaled linearly to have the bond specifications of L_1 match those of L_2. As the SAI/CC requires redistribution of charges[1], this atom retains some Lennard-Jones interaction and forms the anchor point for the dummy region. Each stage of this mutation forms an intermediate state similar to those used with λ-scaling methods, but does not depend on a continuous coupling parameter λ. After each intermediate state is simulated, the energy differences $U_{i,j}$ with respect to all other states are calculated during a post-processing step, allowing the calculation of the free energy differences via mBAR. This is done both for the ligand in complex with the protein, as well as for the ligand in solution, allowing the calculation of $A_{bound}^{L_1 \to L_2}$ by exploiting the thermodynamic cycle shown in Fig. 1.1.

It should be noted that the two methods are independent from each other; it is quite possible to compute a direct transformation using SAI or to use the CC approach alongside soft-core potentials, with increasing adoption of native soft-core methods in MD engines making the latter especially a useful proposition. However, this is as of yet not implemented in `Transformato`.

2.2.2 Theoretical considerations for Restraints in `Transformato`

While `Transformato` yields excellent results for almost all test systems, problems may arise in unstable systems with a ligand prone to leaving the binding site of the receptor within the timeframe of the simulation. This may even be a problem in comparatively stable systems, once longer-than-usual simulations per intermediate state are carried out. To avoid this, restraints can be applied within the system, keeping the ligand within the binding site while still allowing sampling within it. However, irrespective of other considerations, the addition of restraints must not influence the calculated free energy differences, as this would render the entire exercise pointless. Restraints as implemented act as bonded force, and are a simple addition to the sum forming U_{bonded}. Given that the free energy is directly dependent on U via the partition functions (Eq. 2.10) and $\Delta\Delta G$ results from the subtraction of the L_2 pathway from the L_1 pathway (Eq. 2.21), this means that $U_{L_1}^{restraints}$ must be equal to $U_{L_2}^{restraints}$ for the restraints to cancel each other out.

[1] The sum of the partial charges of the terminal junction at the end needs to be the same as that of the original ligand to cancel itself out across the thermodynamic cycle, see [3, 5] for details

However, this is not possible for restraints involving the entire system. In case of a simple harmonic restraint as given by Eq. 2.22), with r the distance between the restrained particles - one anchored on the ligand, one anchored to the protein -, r_0 the equilibrium distance, and k the force constant, the amount of energy added to the system is proportional to the movement of the restrained systems vis-a-vis each other.

$$U^{rest} = 0.5 \cdot k(r - r_0)^2 \tag{2.22}$$

To achieve $\langle U_{L_1}^{rest} \rangle == \langle U_{L_1}^{rest} \rangle$, the first condition is that the number of restraints in both systems is the same. However, as $\langle U \rangle$ is dependent on r, the expectation value for r also must not change. As r is dependent on the interactions between atoms and thus their force field parameters and interactions, that means that the restraint needs to be connected to the identical atom types (or groups of atoms/atom types) at both endstates. The current implementation in `Transformato` achieves this by restricting its restraints to the common core, as that represents a useful list of atoms that will both remain constant during the mutations and have the same atom type in both ligands, with the exception of the terminal junction atom X.

However, the common core does not exist in a vacuum, and r is dependent not only on movement by the ligand, but that of the protein as well. To satisfactorily keep ligands restrained in the binding site while still allowing for low enough force constants to facilitate sampling, current automatic restraints use protein C_α carbons in the binding site as anchor points for the restraint(s). Their positions are affected by the ligand, with different ligand structures producing different effects. These differing interactions will also have differing effects on the movement of the common core structures, further diverging $\langle (r - r_0) \rangle$ between L_1 and L_2. While the anchoring of the restraints exclusively to C_α carbons is an attempt to minimize this and contributions are likely to be small, it is nevertheless an inaccuracy introduced into the calculations.

To avoid this source of error, it is recommended to turn off the restraints for the endstates entirely and just gradually introduce them. With mBAR, as the $\hat{A}_i$ are calculated from the sum differences between two adjacent states (Eq. 2.16), as long as the restraints are introduced away from the endstates, their energy contribution is considered in the calculations between the first intermediate states $\Delta\hat{A}_1 \rightarrow \hat{A}_2$ at both starting points instead of being part of the initial system. As the contribution at the common core is necessarily the same, the terms thus cancel each other out. `Transformato` provides the keyword `scaling` to achieve this behavior. If set, U^{restr} is turned on linearly during the first four intermediate states, with the initial state (the physical ligand) receiving no contribution at all.

Harmonic restraints as in Eq. 2.22 always give a contribution to the total potential energy because the distance $r - r_0$ rarely evaluates to 0. Similarly, the harmonic shape of the restraint means that a restraint capable of stopping escape from a binding site will have significant contributions even within that binding site, discouraging sampling and possibly introducing errors into the energy calculations. To alleviate this, it is recommended to instead use one of the flat-bottom potentials provided, e.g., as shown in Eq. 2.23.

$$U^{rest} = k \cdot step(|r - r_0| - wellsize) \cdot (r - r_0)^2 \tag{2.23}$$

Here r once again represents distance, r_0 the equilibrium distance and k the force constant. New additions are the parameter *wellsize* and a *step* function. The wellsize represents the radius of a sphere around the equilibrium distance. Within this region, the restraint evaluates to zero. As long as the expression $|r - r_0| - wellsize$ evaluates negative, $step(x) = 0$. Should it however evaluate positive, meaning that the restraint has left the well, the step function evaluates to 1 and the standard harmonic restraints are applied. This allows definition of fairly stringent restraints without impact on the sampling within the binding site.

2.2.3 Installation and Usage

Installing `Transformato` is fairly straightforward: it requires a working installation of `conda` and `python`. Using `git clone`, download the package from the repository[2] and run `python setup.py install`. Install a `conda` environment called `fep`, consisting of `Transformato` and all its dependencies, located in `transformato/test_environments/`. Activate the environment and you're done.

To calculate free energies, retrieve a PDB containing your ligand-protein complex for one endpoint, then modify the ligand to suit your purposes, using, e.g., CHARMM-GUI's[3] [25–27] ligand builder. For each endpoint, solvate once the complex including the ligand and once just the ligand using, e.g., CHARMM-GUI's Solvation Builder. Take the output folders, equilibrate them and name them 'complex' and 'waterbox', respectively. You will then need to create a config.yaml file containing the names of your structures and parameters you want your simulation to have - any restraints you wish to apply must also be defined here. The simplest case for specifying restraints is simply done by adding the keyword `restraints: auto` to the *simulation* part of the configuration file. This restrains the entire ligand backbone via its center of mass to the center of mass of the protein backbone. Inside the code, this is referred to as `simple` mode. Another possibility is the generation of restraints on the extremities of the ligand using the `extremities=[int]` keyword. This algorithmically selects those carbon atoms furthest away from the center of mass (and each other), and restrains these and their surroundings to the protein, thus restricting rotational freedom. The downside of this is that it requires some prior knowledge of the ligand's shape - not usually a problem, but potentially in large-scale applications. Of course, this also reduces sampling volume even further than the simple restraint. Lastly, it is also possible to manually define restraints in addition to - or instead of - the automatic ones, using the keyword `restraints: manual` and defining manual restraints below. This allows full use of the MDAnalysis[28, 29] selection syntax, though only common core atoms of the ligand may be selected (cf. above).

To actually create the simulation runs, you will need a submit script, templates of which are readily available at the repository. These files sequentially load the .yaml config and the structure topologies, then propose a common core. At this point, you may add (or remove) atoms from the common core. Afterward, `Transformato` will propose

[2] `https://github.com/wiederm/transformato`
[3] `https://www.charmm-gui.org/`

a mutation route. If it looks satisfactory, it will then create several folders containing intermediate states, each a self-contained simulation. Simulate these (as this is the most computationally intensive step, it is highly recommended to use a cluster for this step) and run the analytics script (a cluster is also recommended here). In the end, you should receive the energy difference along with an uncertainty interval (e.g. `Free energy to common core: 12.02382 [kT] with uncertanty: 0.5843413703 [kT]`).

Significantly more extensive documentation of `Transformato`'s abilities and features is available through the package documentation,[4] along with sample input and data files.

[4] at `https://wiederm.github.io/transformato/`

3 Methods

3.1 The preexisting codebase of `Transformato`

`Transformato` does not contain a main program or executable. Instead, as alluded to in section 2.2.3, it is subdivided into a number of modules that mostly operate independently of each other and are called by the user script on an as-needed basis. The most important of these modules are:

utils.py The first module to be called for relative free energy (RFE) calculations, `utils.py` handles miscellaneous functions. Importantly, it handles user I/O, providing the facilities needed to read and (in part) interpret the `config.yaml` provided by the user. It also handles postprocessing of trajectories.

mutate.py As the name indicates, `mutate.py` is mainly responsible for calculating the necessary steps to transform the endstates into the common core. Its main tool for doing so is the `ProposeMutationRoute` class, which also provides user-interface commands to visualize and manipulate the common core (as the CC generated by `Transformato` may not always be the ideal one, or one suitable to the task at hand). On a technical level, this is accomplished by generating for each state a new *Protein Structure File* (.psf, representing the physical locations of and atom types of the molecule's atoms) and an additional *parameter file* (derived from the original; an excerpt of such a file is shown in Fig. 2.1, with atom types added and parameters modified to represent the mutation at hand) for the various mutations, a set of each representing a new intermediate state.

state.py While `mutate.py` does most of the heavy lifting in calculating and performing the mutations, it is `state.py` and its `IntermediateStateFactory` class that actually writes the generated mutation steps to the intermediary folders, taking the .psf and .prm data provided by `mutate.py` along with the other necessary data, parameter and script files copied from either `Transformato` or the original provided structure, creating for each state a separate folder containing all files to run a simulation at this state, independent of `Transformato`.

analysis.py Lastly, `analysis.py` only is called after the MD simulations. Its purpose is to extract the relative free energy difference from the trajectories post-processed by `utils.py`. It does this using the pyMBAR module developed by Shirts and Chodera [7, 30]. As secondary function, it also generates overlap and potential energy plots used for troubleshooting and quality assurance.

A number of additional, small, miscellaneous modules (`systems.py`, `charmm_factory.py`...) exist, providing helper functions to the above-mentioned modules or for testing.

3.2 Implementation of restraints into `Transformato`

The process of applying restraints was divided into three parts:

1. Processing user input: the general demand for restraints, and parameters for these restraints
2. Finding suitable binding sites and generating the atom selections for the restraints
3. Generating an openMM Force object and applying this force to the actual simulation

These processes were complicated by the fact that the script that generates the intermediate states for processing (and thus has access to user input) is separate from those that do the actual simulations (which need to be independent to be able to run on distributed computing networks).

As much code as possible was exiled into a separate module file called `restraints.py`, with only minimal changes to `state.py` and `mutate.py` being necessary.

Currently, restraints are only available as proof-of-concept for openMM. OpenMMs *CustomCentroidBondForce* was chosen as the basis for all restraints, meaning the restraint forces act upon the atom group's center of mass. To facilitate analysis of ligand and protein structures, MDAnalysis[28, 29] was used.

3.2.1 Processing user input

To leverage the existing codebase, as well as keeping with the established design principles of `Transformato`, additional user input was restricted to the configuration `.yaml` already required; to ensure both backward compatibility and prevent accidental use of restraints, all commands to use restraints are purely optional. If a user is unaware of the possibility of restraints, they will not run into the danger of accidentally using them. Further, there is no additional workflow in the user's submit script - all of it is done during `utils.py::load_config_yaml()` automatically.

Due to the structure of the existing code, no changes to `utils.py` were necessary - all relevant information was available for further processing immediately. During the run of the initial submit script, if `Transformato` finds defined restraints in the configuration file, a separate `restraints.yaml` is created in each of the intermediate states, containing the restraint information passed through from the configuration file along with the atoms in the common core and possible scaling effects.

These `restraints.yaml` are then read in by the `openmm_run.py` during simulation startup. Their mere presence informs the simulation to run the code applying restraints. It thus starts evaluating the restraints specified, cross-checks them against the atoms in the common core, and creates/applies the force.

3.2.2 Generation of restraints

While ideally every system would be run after careful, manual inspection, using hand-crafted restraints, perfectly weighed and modified to the specific system, the reality is that for most purposes, ease of use and speed of "good-enough" restraints is paramount. For this reason, a number of facilities for generating automatic restraints were implemented. These generally come in two flavors: the first is the "simple" restraint, which anchors the combined carbon backbone of the ligand to the surrounding C_α carbons (by default: all C_α carbons carbons between 5 and 15 Å away from the ligand). This effectively restrains the ligand in the binding site, while allowing for movement of said binding site relative to the protein as a whole without affecting the restraint.

The second flavor are the 'extremities' - restraints. Instead of acting upon the entire ligand, these act only upon N areas of ligand carbons furthest away from the ligand's center of mass (COM), restraining them to the protein C_α carbons that surround them. This is significantly more restrictive than the simple restraint as rotational movement is now limited as well.

For all of this, the automatic facility assumes that the ligand is already near its binding site at the protein. Should this not be the case, a molecular docking step must be prepended to the RBFE calculation to find a suitable starting position for the ligand. If for one reason or another, such a starting position cannot be found, no sensible RBFE can be calculated, as it is intrinsically linked to the site-ligand interaction.

Technical details On a technical level, when running the simulation, the `restraints.yaml` created by the Intermediate State Factory is read in. Then, molecular types and networks are analyzed using MDAnalysis. In the first step, the ligand carbons are cross-referenced against the CC provided by the .yaml, and any carbon not found there is discarded. These as a whole then constitute the initial ligand group `group1`. For a simple restraint, the program then takes any protein alpha-carbon from within 0.5 to 1.5 nm of the carbon and uses these as the anchor group (internally referenced as `group2`).

For an "extremities" - type restraint, the process is slightly more complicated. The algorithm first selects the carbon within `group1` that is furthest from the group's center of mass (C1). It then selects the carbon furthest from that carbon (C2). It then retrieves additional carbons up to the amount specified by the `n_extremities` value set by the user by selecting the carbon where the sum of distances from the new carbon to all previously selected carbons is highest, repeating this process for every carbon it requires.

Once all extremity carbons have been selected, the individual restraints are created. Every extremity gets its own restraint, with `group1` consisting of the selected extremity carbon and ligand carbons in close proximity, and `group2` being comprised of protein C_α carbons in a spherical layer between 0.3 and 1 nm around `group1`. Note that these groups exist individually for each restraint, and each extremity is endowed with its own restraint. As such, a value of `n_extremities = 4` would produce 4 restraints with 8 groups total, each applying forces independently of the others. Also, note that no cross-check between these restraints is done; a carbon may very well be a member of multiple

restraints at the same time.

Manual restraints Manual restraints offer expanded functionality compared to automatic restraints. The atoms belonging to `group2` are transcribed directly from the user-provided selection string. The selection for `group1` is similar, but undergoes the same CC adjustment as the automatic restraints. Importantly, unlike automatic restraints, manual restraints are *not* restricted to carbon atoms but may be anything you can express using the MDAnalysis syntax - if you'd like to restrain a ligand entirely by its sulfur-bonded hydrogens to a zinc atom on the opposite side of the protein, that is entirely possible.

3.2.3 Applying the Force

Once the atom selection groups have been found, their constituent atoms need to be referenced and a restraint created within the underlying molecular dynamics engine. Whereas all previous steps are engine-agnostic, at this stage engine-dependent facilities need to be used. For this thesis, only facilities supporting openMM were created. Specifically, all restraints are mapped as openMMs' CustomCentroidBondForce, which applies a given energy expression between the centers of mass of two atom groups. These atom groups are simply arrays of atom indices, similar to those used by openMM. User-defined parameters relevant to the energy calculations alongside the intermediate state-dependent scaling factor are passed directly to the energy expression. A list of implemented energy expressions is provided in table 3.1.

Keyword	Expression	Description
harmonic	0.5*k*(distance(g1,g2)-r0)$\hat{}$2	A standard harmonic potential
flatbottom-oneside-sharp	step(distance(g1,g2)-r0) * (k/2)*(distance(g1,g2))$\hat{}$2	Flatbottom with no potential towards the origin, but a step outside the well
flatbottom-oneside	step(distance(g1,g2)-r0) * (k/2)*(distance(g1,g2)-r0)$\hat{}$2	Flatbottom with no potential towards the origin and no step out of the well
flatbottom-twoside	step(abs(distance(g1,g2)-r0)-w)*k*(distance(g1,g2)-r0)$\hat{}$2	A two-sided flatbottom potential, with steps outside the well

Table 3.1: Overview of the available energy expressions and their corresponding potential shapes. k refers to the spring constant, r is the current and r_0 the initial distance between atom groups `group1` ($g1$) and `group2` ($g2$)

All of these operations happen within the `openmm_run.py` file responsible for running the openMM simulation. After creation, these forces are then injected into the openMM system where they were evaluated alongside the regular forces during runtime.

3.3 Simulations

All inputs were generated using CHARMM-GUI[25], and used the CHARMM36m forcefield[31]. For easier handling and allowing larger timesteps, CHARMM-GUI's facility for hydrogen mass repartitioning[27] was used, unless noted otherwise. Equilibration was done using the openMM - Inputs provided by CHARMM-GUI[24, 32]. To modify the ligand, CHARMM-GUI's ligand designer[26] was used alongside the commercial program Maestro[33]. The simulations themselves were carried out using openMM 7.5[23].

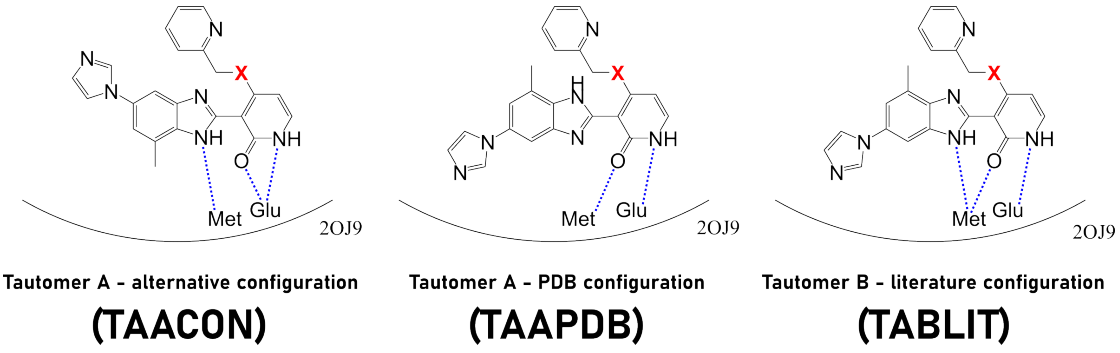
Calculations were conducted via distributed computing utilizing consumer-grade Nvidia GPUs of the RTX 2080, RTX 1080, and RTX 1060 series.

3.3.1 Molecules/systems studied

Code	Associated Protein	Description
TAAPDB	IGF-1R	BMI with two possible binding modes; methyl group anti to binding modes
TABLIT	IGF-1R	BMI with three possible binding modes; methyl group anti to binding modes
TAACON	IGF-1R	BMI with three possible binding modes; methyl group syn to binding modes
ZN148	VIM-2	Base ZN* structure with nitrogen in every aromatic ring
ZN222	VIM-2	ZN* structure with nitrogen removed from both peripheral aromatic rings
ZN223a	VIM-2	ZN* structure with nitrogen removed from one peripheral aromatic ring
ZN223b	VIM-2	ZN* structure with nitrogen removed from alternate peripheral aromatic ring
ZN228	VIM-2	ZN* structure with nitrogen removed from embedded aromatic ring

Table 3.2: Overview of codes (abbreviations) used to refer to the ligands discussed further below. IGF-1R-associated structures are shown in Fig. 3.1, VIM-associated structures in Fig. 3.2.

IGF-1R with modified BMI as ligand



Tautomer A - alternative configuration
(TAACON)

Tautomer A - PDB configuration
(TAAPDB)

Tautomer B - literature configuration
(TABLIT)

Figure 3.1: Possible binding modes of BMI bound to 2OJ9. The red X is substituted with the modification of interest (Table 3.3), in standard BMI X = NH. Dashed bonds indicate likely interactions with protein amino acids.

The PDB structure 2OJ9[1] contains coordinates for a complex of the IGF-1-R insulin growth factor with 3-[5-(1H-imidazol-1-yl)-7-methyl-1H-benzimidazol-2-yl]-4-[(pyridin-2-ylmethyl)amino]pyridin-2(1H)-one (BMI) as inhibitor, first discovered by Velaparthi et. al. in 2007 [34]. IGF-1-R has continued to gather medical attention, especially due to its likely role in tumor propagation [35]. Velaparthi et. al. discuss how a variety of substitutions affects inhibition potency of BMI. Three of these proposed substitutions were

[1] https://www.rcsb.org/structure/2oj9

Code	**Substitute**	**Reported IC_{50} [µM]**
24	NH	0.39
25	S	>25
26	O	>25

Table 3.3: Substitute molecules as used by Velaparthi et al. along with their reported IC_{50} values [34]

simulated. These are referred to by their numeric identifiers (see Table 3.3). Compound 24 - NH was used as the baseline due to its high reported binding affinity to which 25 - sulfur and 26 - oxygen were compared to (refer to Fig. 3.1 for the location of the substitutions). Of particular interest are the possible spatial configurations BMI may adopt in the binding site (Fig. 3.1) - the structures reported by Velaparthi [34] (TABLIT), X-ray crystallography in the PDB (TAAPDB) and a possible alternate configuration (TAACON).

All derivatives of BMI bound to IGF-1R were prepared using the standard procedure outlined in section 3.3, starting from PDB structure 2OJ9.

VIM-2 with zinc ligands

VIM-2 is a Carbapenem-Hydrolyzing Metallo-β-Lactamase, responsible for a subtype of bacterial drug resistance.[36]. ZN148, ZN223, and their derivatives were first presented by Samuelsen et al.[37]. These compounds inhibit Vim-2 activity (likely by chelating the zinc present) and suppress its drug-resistant activity, thus providing a treatment vector for strains containing it.

To prepare the structures for use in Transformato, protonation states were determined by use of Protoss[38, 39] and manual observation, after which ligand and protein were solvated using CHARMM-GUI's solvation builder[25], including patching of the C-Terminus. As certain binding modes require the presence of an OH^- group nearby, but CHARMM-GUI's solvation builder is unable to process the system with it, a roundabout way was chosen where a separate .crd file for the OH group was created and patched in after solvation using `macha`[40]. As this also complicated HMR, Hydrogen Mass Repartitioning was set up using `parmed`[30]. Afterwards, processing with `Transformato` continued as normal.

Tests included a total of five different ZN* - ligands as shown in Fig. 3.2. The nomenclature follows the work by Samuelsen [37].

3.3.2 Binding site dynamics

Initial efforts focused on the implementation of restraints into openMM. To that effect, a number of simulations were conducted using 2OJ9. These did not utilize `Transformato` and were not used to compute free energy differences. Instead, the trajectories saved during these calculations were analyzed with regard to the distances allowed by various restraints and parameters, the effects of their applications on the complex' total potential

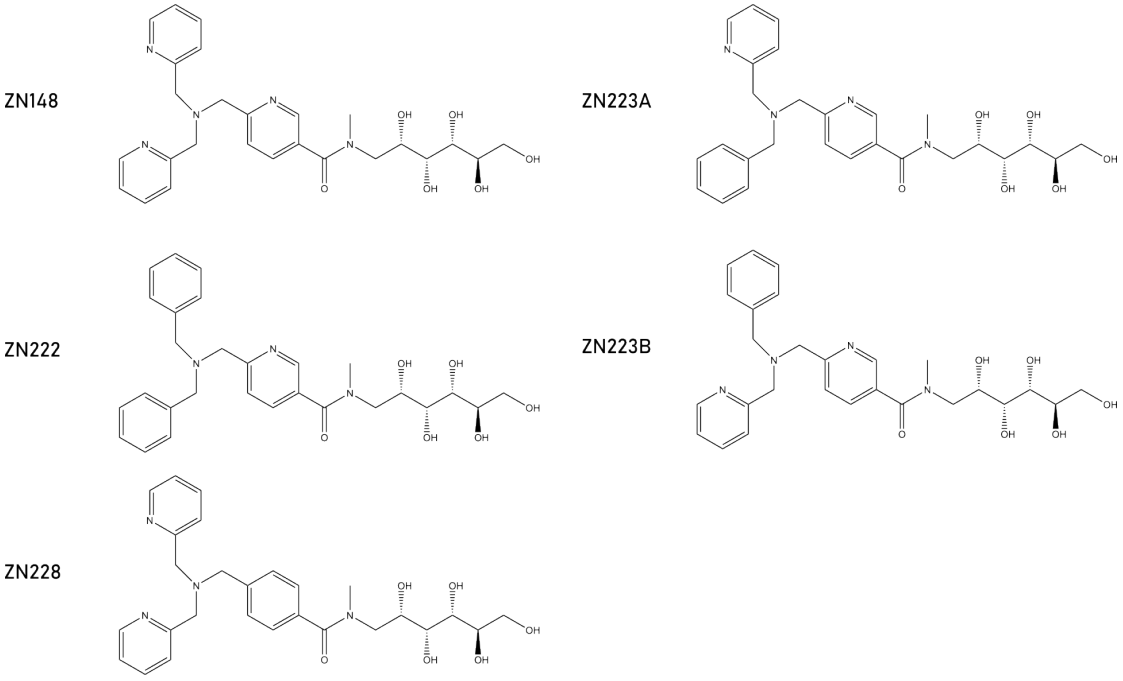


Figure 3.2: The various ZN* - ligands tested against VIM-2.

energy and its root mean square deviation of atomic position (RMSD), a method to measure positional changes of non-hydrogen protein atoms against the original structure [41].

For the 2OJ9 systems, we measured/monitored the distances between the likely interaction partners depending on the structure used (cf. to Fig. 3.1). A smaller number of analogous simulations were also undertaken using VIM-2. Here, the distance between the nitrogen in the heterocycles to the next acidic hydrogen was measured. See Tables 6.1–6.4 for the details of the anchor points.

3.3.3 Restraint dynamics simulations

To ensure correct application of the restraints during use with `Transformato`, the length of the "virtual bonds" created by the restraints (an example is shown in Fig. 3.3) were logged and aggregated over the intermediate states. These lengths are the distance (r) terms in the potentials underlying the restraints (see Sec. 3.2.2). This enabled comparative analysis of the relative restraint length as parameters were changed and in comparison to the unrestrained systems. Restraint length measurements are relative to their starting position, meaning a restraint at the initial distance r_0 of the components would have a relative distance of 0, a bond decreasing in length a negative value, and a bond increasing in length a positive value. In practice, mostly positive relative distances were observed.

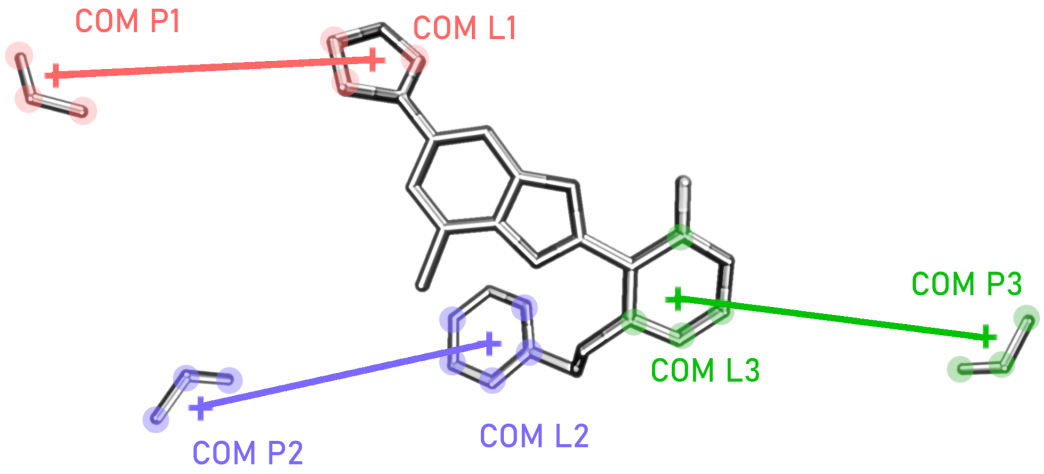


Figure 3.3: Schematic view of a three-extremities (ex3) restraint acting on the BMI ligand in 2OJ9. The binding site is simplified, the twig-like structure represents C_α carbons close enough to be considered for the particular restraint. For each extremity, the carbon selected by the algorithm described in section 3.2.2 gets grouped with surrounding carbons as the ligand group (**L***). It then selects surrounding protein alpha-carbons as the anchor group (**P***, corresponding anchor and ligand groups are colored identically). The restraint creates a "virtual bond" between the centers of mass of these two groups, with the restraint acting as bonded force according to the functional form selected.

3.3.4 `Transformato` RBFE calculations

Here, the focus lay primarily on probing whether the introduction of restraints had a significant effect on calculated free energy differences for the protein-ligand complex. No free energy simulations were conducted for just the ligand. Free energy simulations for 2OJ9 and VIM-2 with both simple and three-extremities (ex3, see Table 3.4) restraints applied were carried out and the results (free energy differences) compared to those obtained in otherwise identical simulations in the absence of restraints. Three replicates were used for each set of parameters. As the simulations were resource-intensive, simulations were only conducted one-sided, meaning that only one endstate was mutated into the common core, with the other endstate being chosen in such a way that it was structurally identical to the common core. However, due to differences arising from charge compensation in the terminal junction (see Fig. 3.4) this means that despite the identical structure this does not supplant a full two-sided simulation. As such, all results

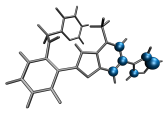
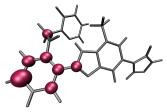
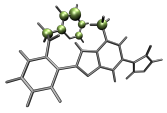
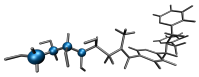
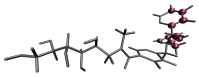
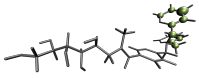
Base Structure	Restraint Name	Core of Origin	Schematic View of Ligand Group
2OJ9	**RO1**	**C16**	
	RO2	**C4**	
	RO3	**C6**	
VIM2	**RV1**	**C07**	
	RV2	**C17**	
	RV3	**C23**	

Table 3.4: Correspondence of three-extremities (ex3) restraint names to their (line) color and core of origin, along with a schematic view of their placement. Restraint particulars are detailed in Tables 6.1 – 6.4.

are qualitative rather than quantitative.

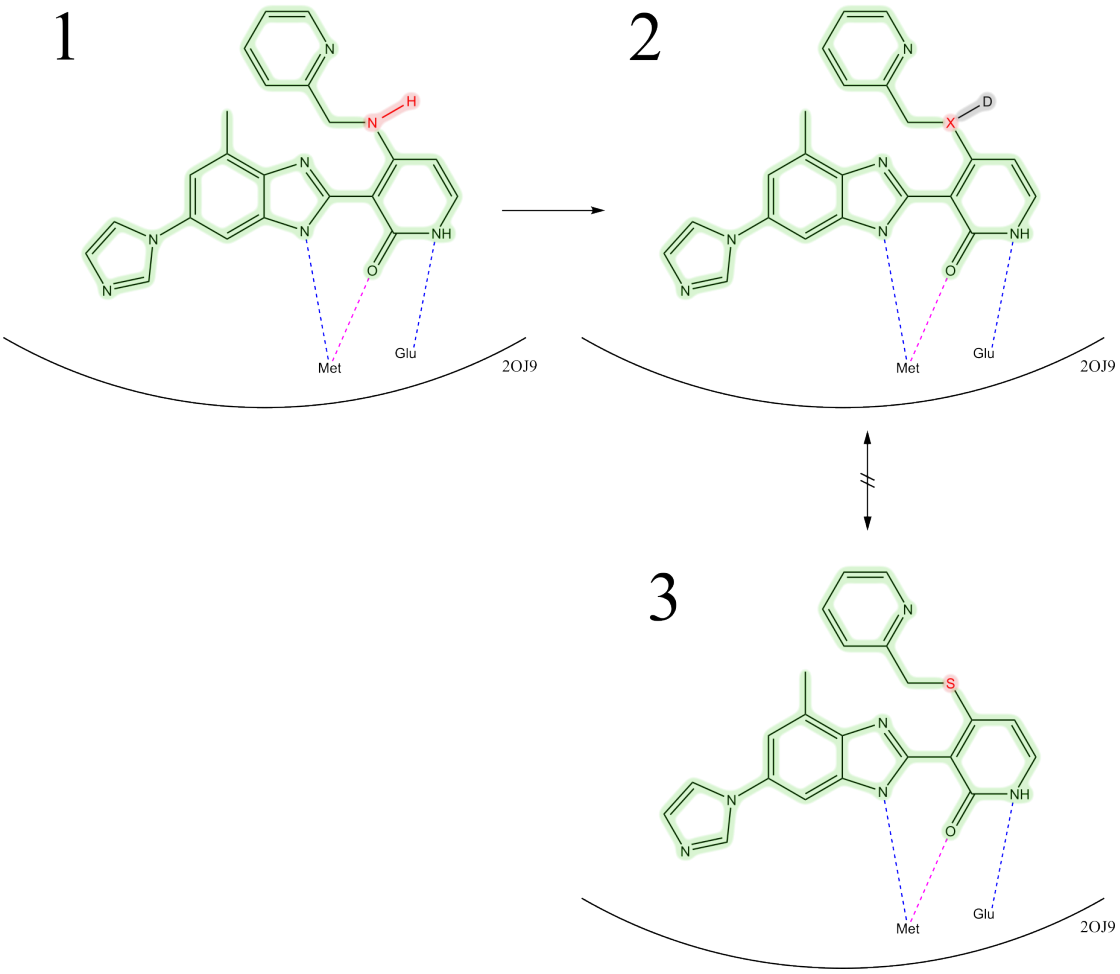


Figure 3.4: Schematic display of mutation for 2OJ9 TABLIT 24 to the common core with TABLIT 25. Green: Common Core. Red: Objects that are mutated. Grey: The dummy region resulting from the superfluous hydrogen. The common core (2) resulting from mutation of (1) is not exactly equivalent to the unmutated structure (3)

4 Results

4.1 Binding site dynamics

These technical integration tests alongside with data reporting from openMM clearly showed the effectiveness of applying restraints; restrained ligands showed significantly less movement away from the binding site, while RMSD and potential energy graphs showed no significant changes from unrestrained systems, indicating no apparent problems with influence on the energy calculations.

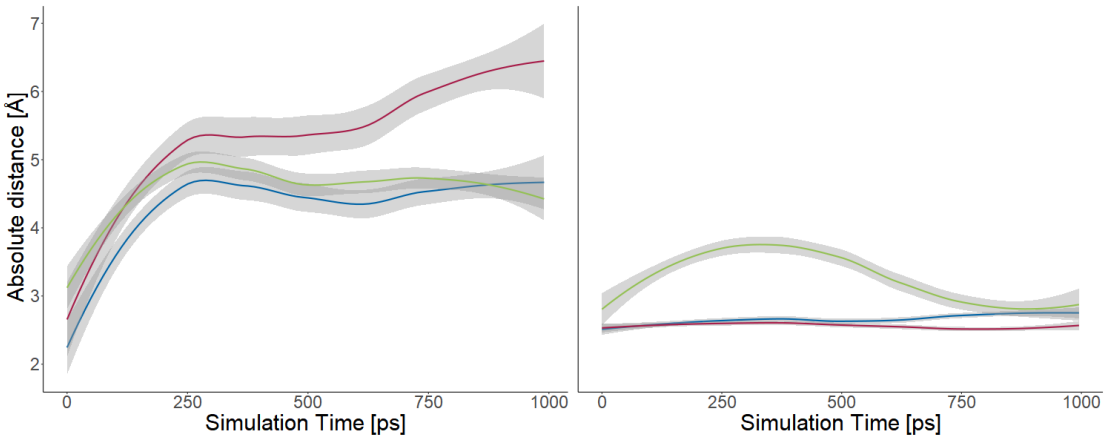


Figure 4.1: Binding site dynamics plot for TABLIT 24 to 25, k=0 (left) and k=400 (right), with harmonic restraints acting directly on the virtual bond anchors. Distances monitored (Color:Type/Index): Blue: H9/4740-O/1518 (Glu), Red: O/4693 - HN/1539 (Met), Green: H14/4735 - O/1554 (Met). Line shows average across all intermediate states, shaded area shows 95% confidence interval.

4.2 Restraint dynamics simulations

A total of 181 simulations were conducted, of which only a small subset is presented here (full data available via the repository). To visualise the relative distances of the harmonic potential encompassing the restraint (exact atoms included in the restraint groups are listed in Tables 6.1 - 6.4), distance data was aggregated across time and intermediate states, and is displayed as arithmetic mean plus the 95% confidence interval as shaded area. A restraint at its initial distance has a value of 0, expansion of the bond gives positive values, contractions negative. *Y - axes equally scaled, all restraints using a harmonic potential.*

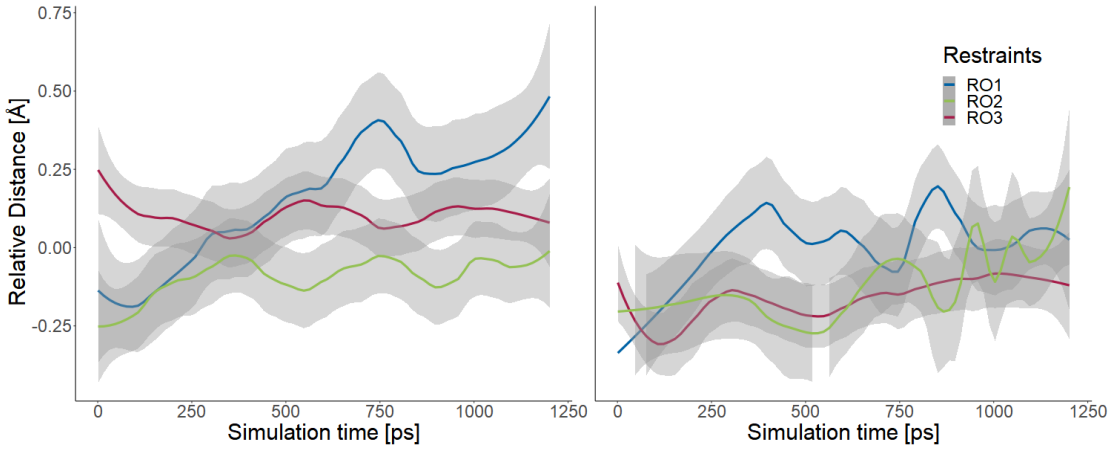


Figure 4.2: Relative distance plot for TAAPDB 24 to 25, k=0 (left) and k=400 (right), three extremities restraint. Colors correspond to restraints as listed in Table 3.4.

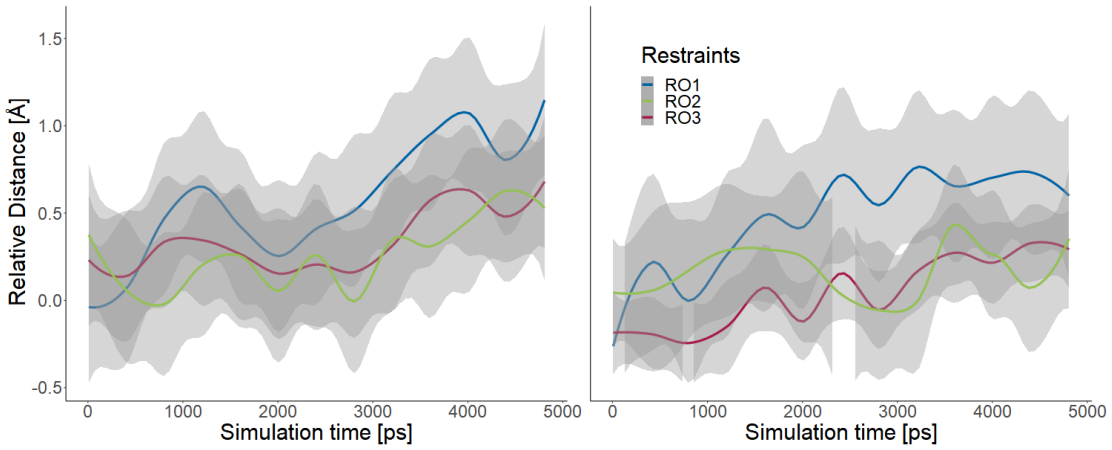


Figure 4.3: Relative distance plot for TABLIT 24 to 25, k=3 (left) and k=100 (right), three extremities restraint. Colors correspond to restraints as listed in Table 3.4.

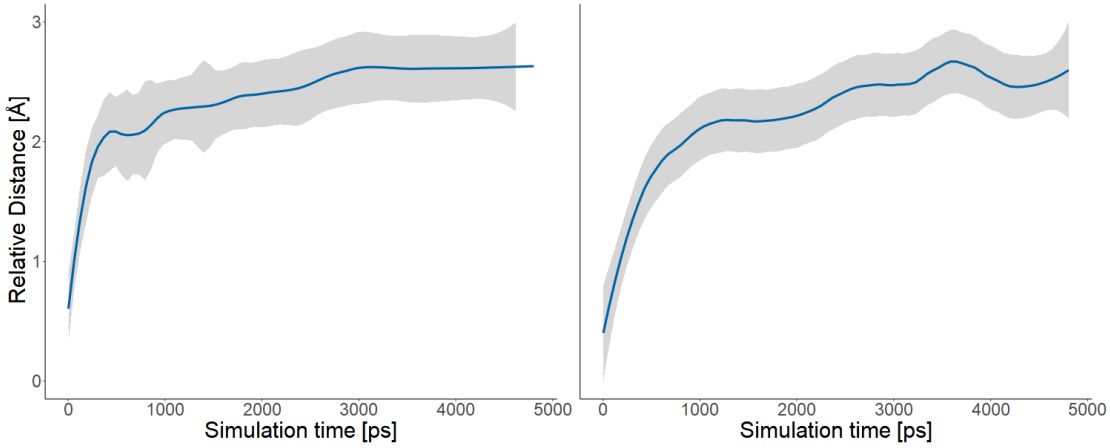


Figure 4.4: Relative distance plot for ZN222 to ZN148, k=3 (left) and k=100 (right), simple restraint.

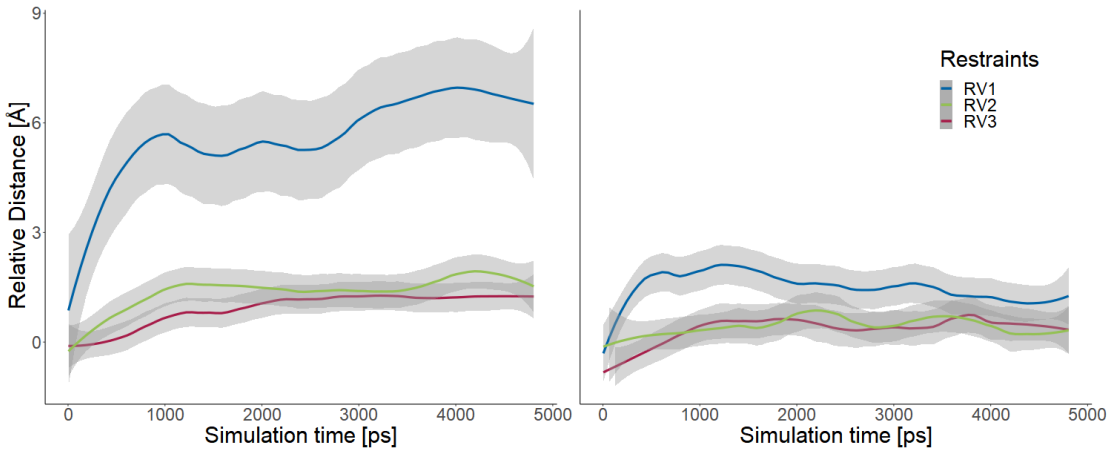


Figure 4.5: Relative distance plot for ZN222 to ZN148, k=0 (left) and k=100 (right), three extremities restraint. Colors correspond to restraints as listed in Table 3.4.

4.3 `Transformato` RBFE calculations

RBFE calculations were conducted as described in section 3.3, varying parameters such as force constant and restraint type. In Fig. 4.6 to Fig. 4.12, results from the simulations utilizing a three-extremities (ex3) restraint are always shown on the top, and results from the simulations utilizing a simple restraint at the bottom of the plot. Unrestrained results are included in both groups to facilitate comparison. Due to time and computational constraints, not all variations for all systems were explored. A minimum of three replicates were used for each data point. Identically colored boxes represent equivalent values of the force constant k. The scale of the Y-axis is fixed across comparative results. Datasets may include both scaled (using the `scaling` option as described in section 2.2.2) and nonscaled results; an unpaired Student's T-Test [42] was applied to test for significant differences between the means of the samples. As it failed to find such differences at a confidence level of 95%, scaled and unscaled results were combined for the purposes of these plots, effectively giving $n = 6$ for parameter sets whenever scaled and nonscaled simulations were conducted.

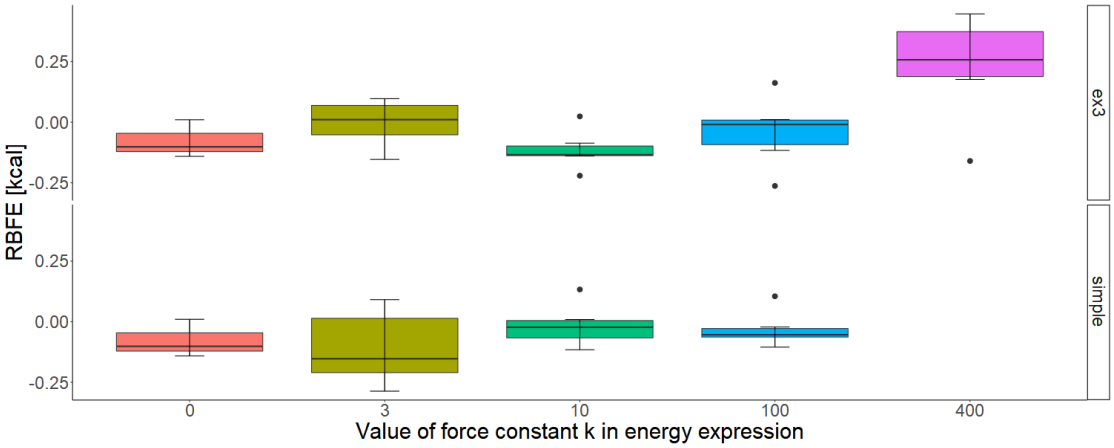


Figure 4.6: RBFE results for TAAPDB 24 to TAAPDB 25 at a runtime of 1.25 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

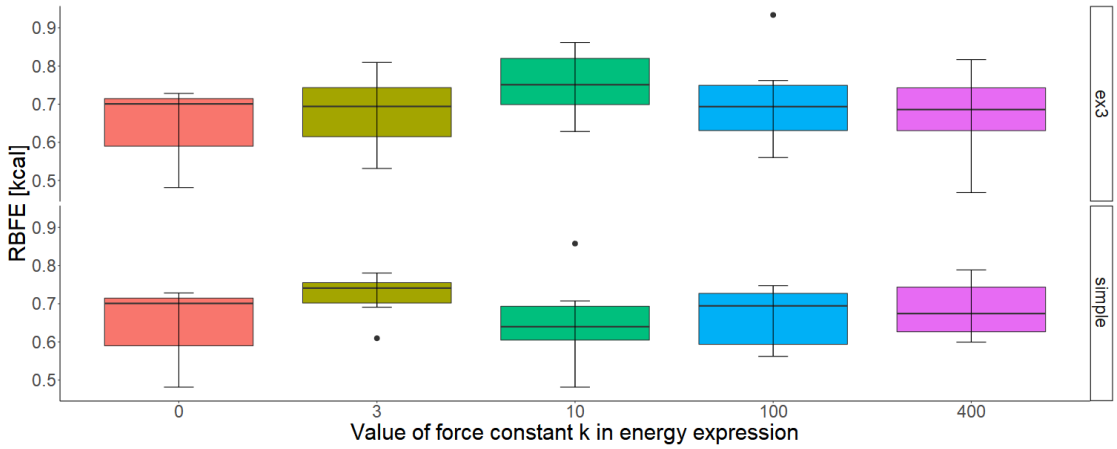


Figure 4.7: RBFE results for TABLIT 24 to TABLIT 25 at a runtime of 1.25 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

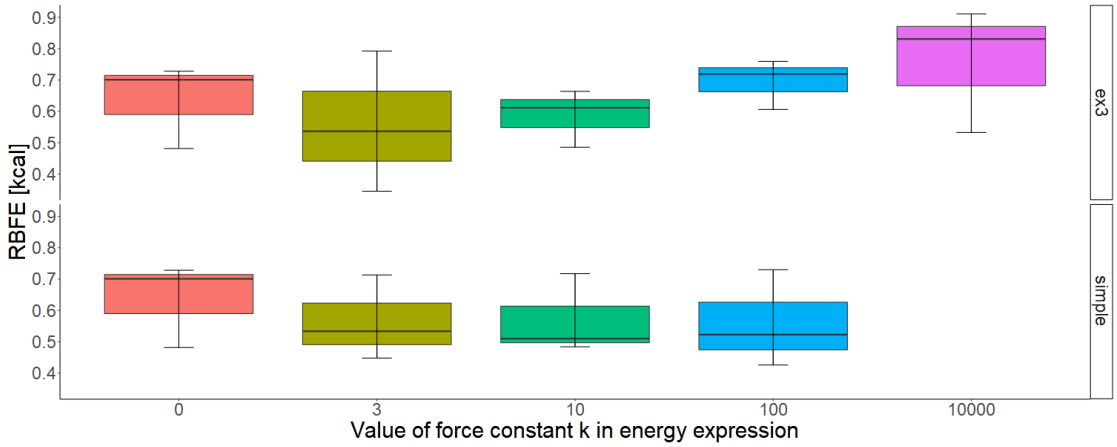


Figure 4.8: RBFE results for TABLIT 24 to TABLIT 25 at a runtime of 5 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

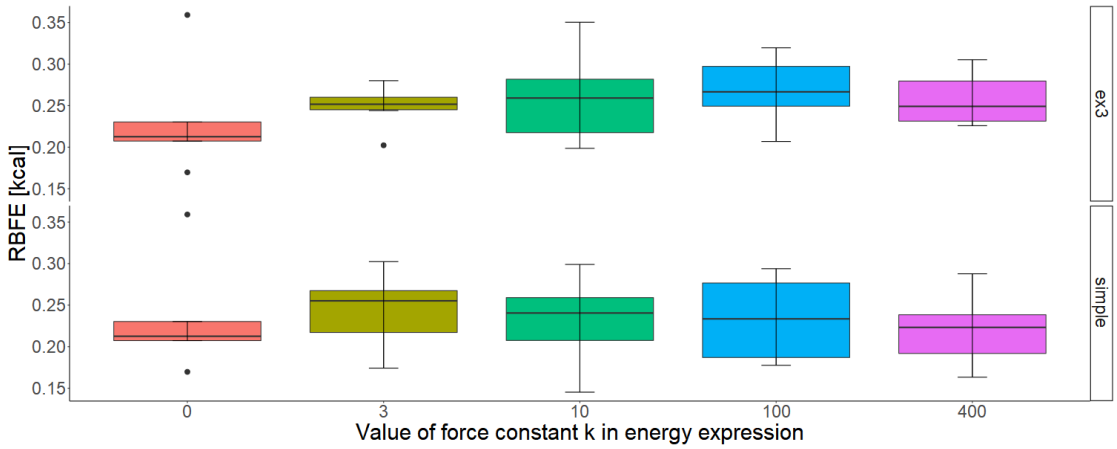


Figure 4.9: RBFE results for TABLIT 24 to TABLIT 26 at a runtime of 1.25 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

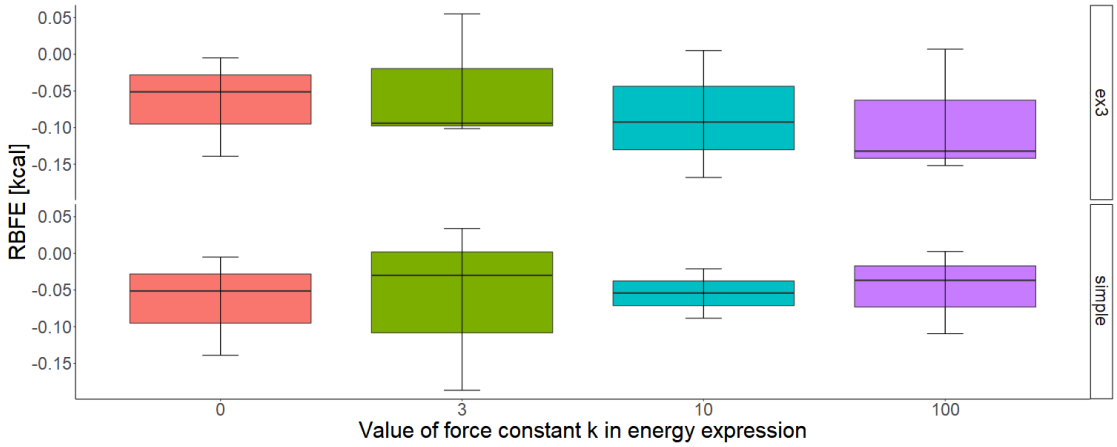


Figure 4.10: RBFE results for ZN148 to ZN222 at a runtime of 5 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

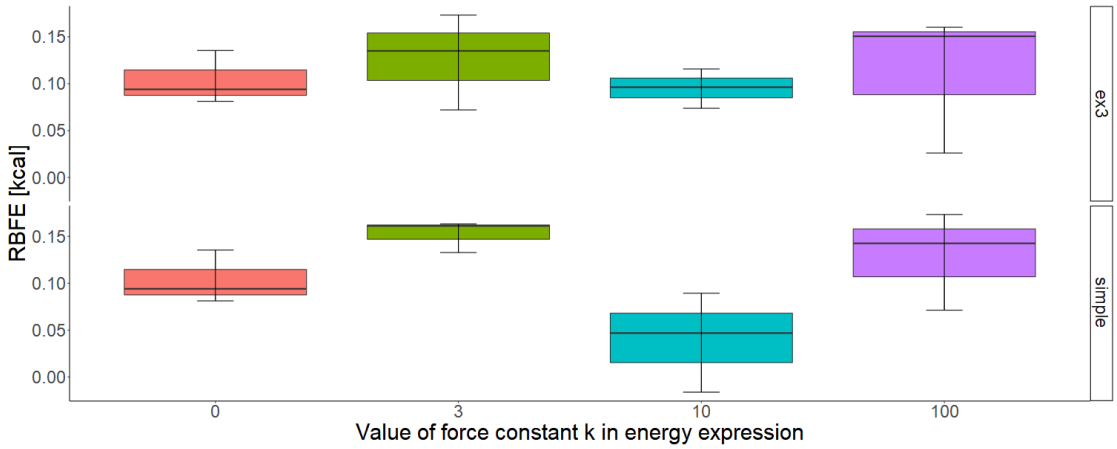


Figure 4.11: RBFE results for ZN148 to ZN223a at a runtime of 5 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

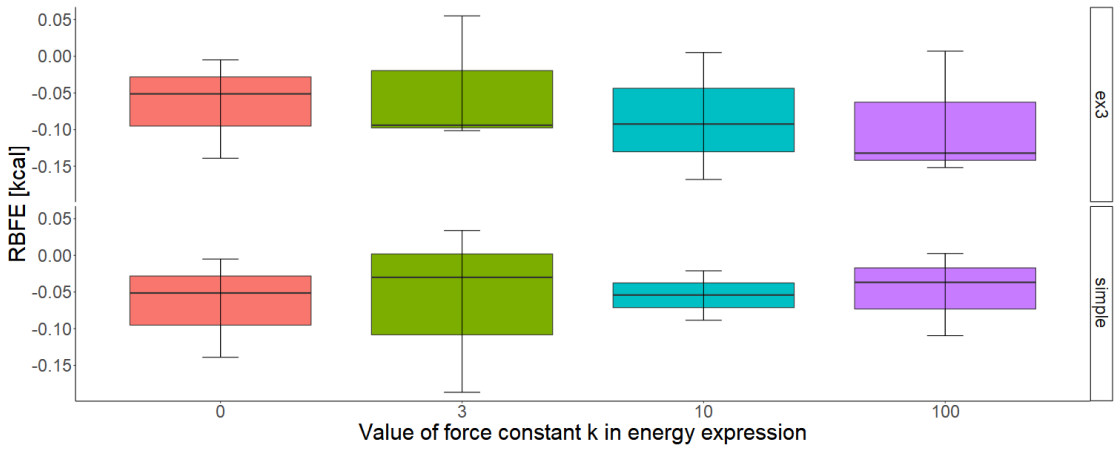


Figure 4.12: RBFE results for ZN148 to ZN223b at a runtime of 5 ns for both simple and three-extremities restraints. $k = 0$ represents the unrestrained comparison system.

5 Discussion

5.1 Comparison of restrained to unrestrained results

Restraint dynamics (Section 4.2) Here, the first obvious result is the fact that restraints themselves work; significant differences can be observed between the dynamics of the unrestrained and heavily restrained systems, seemingly without influencing the RBFE results. This becomes most obvious in Fig. 4.5, showing results for a VIM-2 derived system where an extremity group of the unrestrained ligand moved away on average up to 9 Å during the simulation, whereas in the restrained system the extremity group only moved away 3 Å. Similar, though less drastic results can be observed in the other data presented. As such, it can be concluded that at the basic technical level, the restraints function as designed. All of the data presented here is achieved through the use of harmonic potentials; while flat-bottomed potentials were implemented, significant modifications between the originally tested and now-implemented version along with time constraints prevented their inclusion in the presented dataset. However, the results of the limited number of simulations conducted are aligned with expectations.

Concerning individual systems, ligands in the VIM-2-based systems seem most likely to leave their binding site / initial position, both with and without restraints in place, showing significant deviations even within the limited timeframes simulated. A similar, but much less distinct behavior can be observed for TAAPDB, with TABLIT being the most stable system in comparison.

Transformato RBFE calculations (Section 4.3, Fig. 4.6 – 4.12) When discussing the results of the free energy calculations, TAAPDB is somewhat of an outlier. The most heavily restrained system with $k = 400$ (Fig. 4.6) is the only system to show a significant deviation in computed free energy difference compared to the unrestrained systems and to fail the Student's t-test when compared to the unrestrained system[1]. This would indicate that at a certain point, restraints do impact RBFE calculations. However, it should be noted that this behavior is present in both scaled and nonscaled simulations, that a very high k of 400 is required, and that it is the only set to show such deviation. Notably, TABLIT does not significantly differ from the unrestrained set even at $k = 10000$. It should also be noted that it is only an outlier in the context of the very limited spread of the data presented here, with a mean deviation of 0.19 kcal/mol to the unrestrained set. Given typical uncertainties of 1-2 kcal for RBFE calculations, this does not seem a significant factor.

Overall, however, no impact of restraints on the calculated free energies can be observed. Within the limitations discussed in sections 2.2.1 and 3.3, this would seem to validate the theory that restraints do not have significant impact on RBFE calculations. This, on the other hand also means that there seem to be no significant *benefits* to using

[1] After removal of the statistical outlier aligned with the unrestrained set

restraints; however this may be a consequence of the very limited simulation times (1.25 or 5 ns). Restraints may also find utility in non-standard calculations, such as those approximating a pull-mechanism.

5.2 Limitations of the current approach and future possibilities

While the results thus far look promising, some inherent technical problems do present themselves that may limit applicability in some cases. Firstly, the strict delineation between the common core (to which the restraints are applied) and everything else (where they are not) may present a problem in cases where a significant portion of the ligand is mutated. This problem also arises when connecting parts are mutated if the outlying parts are considered separate by `Transformato`[2], though in this case it may be circumvented by manually modifying the common core. In a similar vein, there currently exist no mechanism to accommodate changes in the protein.

Automation improvements With `Transformato`'s general push towards accessibility and quick setup even for large batches of ligands, the current process to introduce restraints, even using the automated mode, is unsatisfactory. In particular, the productive use of extremity restraints requires prior manual inspection of the ligand in question, and parameter setup at this stage is mostly guesswork. While the default values should be sensible for most use cases, further automation with consideration of molecule and binding site geometry along with charge interactions should make it possible to generate good data without requiring human intervention, significantly increasing its usability for lead generation.

Quality-of-life improvements Currently, the `restraints.py` API does not expose much of the underlying framework, complicating efforts to expand or modify the provided functions. In particular, not all parameters used for the generation of restraints are readily accessible, including the algebraic expression used to calculate restraint forces. Modifying these parameters currently requires modifications of the `restraints.py` itself, necessitating reinstallation. This would complicate multi-platform deployments especially on clusters, troubleshooting etc. Ideally, an API would be provided to expose these parameters directly through the `submit.ipynb`. This, however, would require extensive expansion of the I/O capabilities between the submit and simulation stages, as the current YAML framework seems ill-suited for these kinds of modifications.

Package-independent cluster computation As of now, `restraints.py` is the only `Transformato` module that, in order to function, must be installed on the cluster used to run the simulationS. This also includes some of its dependencies, notably MDAnalysis,

[2] e.g.: in the 2OJ9 - derived structures, `Transformato` by default assumes two seperate CC's, with the mutated atom (X in Fig. 3.1) being outside of both. Instead, it can be included as the terminal junction. While this case only left a single atom out of the common core, if larger parts of the molecule are not in the common core, that part of the molecule cannot be restrained.

which complicates the workflow compared to standard `Transformato` usage, and may preclude deployment in non-permissive environments. At the very least, it introduces an additional layer of complexity with involvement of IT staff. Given the current limitations applied to the restraint, in particular the fact that it is only applied to the immutable common core, it should be possible to frontload the structure analysis underpinning the creation of the restraint, having it be handled at the `submit.ipynb` stage instead of the `openmm_run.py` stage. Assuming a satisfactory mechanism can be implemented to keep the openMM-generated atom idx constant, there would be no need for MDAnalysis or the restraints module being loaded during simulation.

CHARMM integration Currently, restraints are an experimental feature only available when using openMM. However, as otherwise the complete feature set of `Transformato` is available under CHARMM, it could be considered a waste to not have restraints implemented as well. This would be non-trivial, as CHARMM does not have the same type of centroid prototype force openMM possesses which is exploited for these restraints. Given language limitations, it would likely also require creation of dynamic simulation scripts during the submit stage instead of the static script used for openMM, thus creating a significant opportunity for critical bugs to arise.

5.3 Conclusion

A method to introduce restraints into `Transformato` was successfully implemented and performed adequately in testing. In the context of unstable systems, restraints accomplish their goal of retaining ligand-protein closeness without significant impact on calculated RBFE values. Conversely, no significant benefit from using restraints was observed. As testing was not carried out under production conditions, in particular the simulation length per state was considerably shorter, further research is required to draw definite conclusions.

6 Annex

6.1 Definitions

As far as possible, equations in this thesis are notated and styled as in the Alchemistry Wiki maintained by Shirts and Chodera [43]. The symbols and notations used have the following meaning:

Q	the partition function
k_B	Boltzmann's constant
T	Temperature
P	Pressure
V	(Box) Volume
U	Total internal energy in a MD simulation, both potential and kinetic energies
N	Number of particles in the system
β	Substitute for $(k_B T)^{-1}$ by convention
Γ	Phase Space Volume
λ	the *alchemical variable* or *coupling parameter*. Used to describe the progress of a transformation along the alchemical path
K	The *state variable*, describing all conditions and parameters of the thermodynamic system
k	A specific state found in K
u	The *reduced potential* $u = \beta(U + PV - \sum \mu N_i)$. For NPT ensembles, $\sum \mu N_i = 0$

6.2 List of anchor atoms for restraints

These tables give an overview of the exact molecular restraints calculated by the automatic restraint generation used for this thesis. **Please note: The residue numbers and atom identifiers listed below originate from the modified PDBs** used for the calculations and are not necessarily identical to those found in e.g. the RCSB Protein Database. While the protein residue numbers stay identical, the IDs between the various ligands vary as they are reshuffled during the ligand building process, and, thus, are not given here. The corresponding structure files may be found in the data repository.

	PDB Atom Name/ID		
Core	C16	C4	C6
Ligand Group Extremities	C12 C13 C14 C15 C16 C17	C1 C2 C3 C5 C11 C18	C5 C7 C8 C9 C10 C22
Protein residue numbers (all named CA)	332, 346, 365, 373, 390, 452, 468, 489, 1541, 1558, 1572, 1596, 1603, 1656, 1667, 2359, 2375	347, 390, 452, 468, 489, 714, 730, 740, 759, 1239, 1255, 1279, 1298, 1455, 1471, 1490, 1507, 1521, 1540, 1558, 1596, 1603, 2359, 2520, 2527, 2566	347, 366, 373, 390, 397, 408, 428, 436, 452, 759, 1596, 1603, 2310, 2334, 2359 , 2527, 2559, 2566, 2583, 2597, 2621, 2633

Table 6.1: Three - extremities (ex3) restraint anchor points for all structures derived from 2OJ9, grouped by the core of origin.

Ligand Group (Atom Names)	Protein Anchor Group (protein residue numbers, all named CA)
C1, C2, C3	280, 297, 308, 332, 347, 366, 373, 390
C4, C5, C6	397, 408, 428, 435, 452, 468, 489, 504
C7, C8, C9	690, 714, 730, 740, 759, 781, 795, 1053
C10, C11, C12	1223, 1239, 1255, 1279, 1436, 1455, 1471, 1490
C13, C14, C15	1507, 1522, 1541, 1558, 1572, 1603, 1615, 1634
C16, C17, C18	1656, 1667, 1688, 1731, 2300, 310, 2334, 2348
C19, C20	2359, 2376, 2392, 2402, 2429, 2449, 2463, 2479 2501, 2520, 2527, 2539, 2559, 2566, 2583, 2597 2621, 2633, 2652

Table 6.2: Simple restraint anchor points for all structures derived from 2OJ9.

	PDB Atom Name/ID		
Core	C07	C17	C23
Ligand Group Extremities	C04	C14	C14
	C05	C15	C15
	C06	C16	C16
	C15	C18	C17
	C16	C19	C20
	C17	C21	C21
		C22	C22
		C23	C24
		C24	C25
		C27	C28
Protein Anchors (all named CA)	852, 876, 883	549, 2199, 2464	456, 473, 484
	893, 915, 1317	2471, 2481, 2491	504, 523, 533
	1329, 1381, 1388	2510, 2531, 2576	549, 572, 584
	1760	2600, 2614, 2625	1329, 2471, 2576,
		2642, 2656, 2672	2625, 2635, 2642
		3086, 3098, 3105	2656, 3098, 3105
		3122	3122

Table 6.3: Three - extremities (ex3) restraint anchor points for VIM-2 - derived structures, grouped by the core of origin.

Ligand Group (Atom Names)	Protein Anchor Group (protein residue numbers, all named CA)
C1, C2, C3	87, 162, 178, 413, 432, 442, 456, 473
C4, C5, C6	484, 504, 516, 523, 533, 549, 572, 584
C7, C8, C9	595, 609, 816, 828, 842, 852, 876, 883
C10, C11, C12	893, 915, 929, 1238, 1249, 1263, 1280, 1300
C13, C14, C15	1317, 1329, 1341, 1365, 1381, 1388, 1395, 1697
C16, C17, C18	1738, 1753, 1760, 1774, 2199, 2216, 2464, 2471
C19, C20, C21	2481, 2510, 2625, 2635, 2642, 2656, 2672, 2682
C22, C23, C24	2694,3086,3098,3105,3122
C25, C26, C27	2621, 2633, 2652
C28	

Table 6.4: Simple restraint anchor points for all structures derived from VIM-2

List of Figures

List of Tables

References

(1) Cournia, Z.; Allen, B.; Sherman, W. Relative Binding Free Energy Calculations in Drug Discovery: Recent Advances and Practical Considerations. *J. Chem. Inf. Model.* **2017**, *57*, 2911–2937.

(2) Klimovich, P. V.; Shirts, M. R.; Mobley, D. L. Guidelines for the analysis of free energy calculations. *J. Comput. Aided Mol. Des.* **2015**, *29*, 397–411.

(3) Karwounopoulos, J.; Wieder, M.; Boresch, S. Relative binding free energy calculations with transformato: A molecular dynamics engine-independent tool. *Front. Mol. Bio.* **2022**, *9*.

(4) Braunsfeld, B. Implementation and Testing of CHARMM as Backend to the Free Energy package Transformato. *Master's Thesis, University of Vienna* **2021**, DOI: `10.25365/thesis.66300`.

(5) Wieder, M.; Fleck, M.; Braunsfeld, B.; Boresch, S. Alchemical free energy simulations without speed limits. A generic framework to calculate free energy differences independent of the underlying molecular dynamics program. *J. Comput. Chem.* **2022**, *43*, 1151–1160.

(6) Fleck, M.; Wieder, M.; Boresch, S. Dummy Atoms in Alchemical Free Energy Calculations. *J. Chem. Theory Comput.* **2021**, *17*, 4403–4419.

(7) Shirts, M. R.; Chodera, J. D. Statistically optimal analysis of samples from multiple equilibrium states. *J. Chem. Phys.* **2008**, *129*, 124105.

(8) Bennett, C. H. Efficient estimation of free energy differences from Monte Carlo data. *J. Comput. Phys.* **1976**, *22*, 245–268.

(9) Tzeliou, C. E.; Mermigki, M. A.; Tzeli, D. Review on the QM/MM Methodologies and Their Application to Metalloproteins. *Molecules* **2022**, *27*, 2660.

(10) Vanommeslaeghe, K.; Hatcher, E.; Acharya, C.; Kundu, S.; Zhong, S.; Shim, J.; Darian, E.; Guvench, O.; Lopes, P.; Vorobyov, I.; MacKerell, A. D. CHARMM General Force Field (CGenFF): A force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *J Comput Chem* **2010**, *31*, 671–690.

(11) Dror, R. O.; Dirks, R. M.; Grossman, J.; Xu, H.; Shaw, D. E. Biomolecular Simulation: A Computational Microscope for Molecular Biology. *Annu. Rev. Biophys.* **2012**, *41*, 429–452.

(12) Hollingsworth, S. A.; Dror, R. O. Molecular dynamics simulation for all. *Neuron* **2018**, *99*, 1129–1143.

(13) Kumar, S.; Nussinov, R. Close-Range Electrostatic Interactions in Proteins. *ChemBioChem* **2002**, *3*, 604–617.

(14) Orabi, E. A.; Öztürk, T. N.; Bernhardt, N.; Faraldo-Gómez, J. D. Corrections in the CHARMM36 Parametrization of Chloride Interactions with Proteins, Lipids, and Alkali Cations, and Extension to Other Halide Anions. *J. Chem. Theory Comput.* **2021**, *17*, 6240–6261.

(15) Jensen, F., *Introduction to Computational Chemistry*, Third edition; Wiley: Chichester, UK ; Hoboken, NJ, 2017; 1 p.

(16) Mey, A. S. J. S.; Allen, B.; Macdonald, H. E. B.; Chodera, J. D.; Kuhn, M.; Michel, J.; Mobley, D. L.; Naden, L. N.; Prasad, S.; Rizzi, A.; Scheen, J.; Shirts, M. R.; Tresadern, G.; Xu, H. Best Practices for Alchemical Free Energy Calculations. *LiveCoMS* **2020**, *2*, DOI: `10.33011/livecoms.2.1.18378`.

(17) Straatsma, T. P.; Berendsen, H. J. C.; Postma, J. P. M. Free energy of hydrophobic hydration: A molecular dynamics study of noble gases in water. *J. Chem. Phys.* **1986**, *85*, 6720–6727.

(18) Boresch, S.; Bruckner, S. Efficiency of alchemical free energy simulations. *J. Comput. Chem.* **2011**, *32*, DOI: `10.1002/jcc.21713`.

(19) Tan, Z. On a Likelihood Approach for Monte Carlo Integration. *J. Am. Stat. Assoc.* **2004**, *99*, 1027–1036.

(20) Beutler, T. C.; Mark, A. E.; van Schaik, R. C.; Gerber, P. R.; van Gunsteren, W. F. Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations. *Chem. Phys. Lett.* **1994**, *222*, 529–539.

(21) Boresch, S.; Bruckner, S. Avoiding the van der Waals endpoint problem using serial atomic insertion. *J. Comput. Chem.* **2011**, *32*, 2449–2458.

(22) Li, Y.; Nam, K. Repulsive soft-core potentials for efficient alchemical free energy calculations. *J. Chem. Theory Comput.* **2020**, *16*, 4776.

(23) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L.-P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; Wiewiora, R. P.; Brooks, B. R.; Pande, V. S. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Comput. Biol.* **2017**, *13*, e1005659.

(24) Brooks, B. R. et al. CHARMM: The biomolecular simulation program. *J. Comput. Chem.* **2009**, *30*, 1545–1614.

(25) Jo, S.; Kim, T.; Iyer, V. G.; Im, W. CHARMM-GUI: A web-based graphical user interface for CHARMM. *J. Comput. Chem.* **2008**, *29*, 1859–1865.

(26) Guterres, H.; Park, S.-J.; Cao, Y.; Im, W. CHARMM-GUI ligand designer for template-based virtual ligand design in a binding site. *J. Chem. Inf. Model.* **2021**, *61*, 5336–5342.

(27) Gao, Y.; Lee, J.; Smith, I. P. S.; Lee, H.; Kim, S.; Qi, Y.; Klauda, J. B.; Widmalm, G.; Khalid, S.; Im, W. CHARMM-GUI supports hydrogen mass repartitioning and different protonation states of phosphates in lipopolysaccharides. *J. Chem. Inf. Model.* **2021**, *61*, 831–839.

(28) Michaud-Agrawal, N.; Denning, E. J.; Woolf, T. B.; Beckstein, O. MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *J. Comput. Chem.* **2011**, *32*, 2319–2327.

(29) Gowers, R. J.; Linke, M.; Barnoud, J.; Reddy, T. J. E.; Melo, M. N.; Seyler, S. L.; Domański, J.; Dotson, D. L.; Buchoux, S.; Kenney, I. M.; Oliver Beckstein In *Proceedings of the 15th Python in Science Conference*, ed. by Benthall, S.; Scott Rostrup, 2016, pp 98–105.

(30) Shirts, M. R.; Klein, C.; Swails, J. M.; Yin, J.; Gilson, M. K.; Mobley, D. L.; Case, D. A.; Zhong, E. D. Lessons learned from comparing molecular dynamics engines on the SAMPL5 dataset. *bioRxiv* **2016**, 077248.

(31) Huang, J.; Rauscher, S.; Nawrocki, G.; Ran, T.; Feig, M.; de Groot, B. L.; Grubmüller, H.; MacKerell, A. D. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods* **2017**, *14*, 71–73.

(32) Lee, J. et al. CHARMM-GUI input generator for NAMD, GROMACS, AMBER, OpenMM, and CHARMM/OpenMM simulations using the CHARMM36 additive force field. *J. Chem. Theory Comput.* **2016**, *12*, 405–413.

(33) Schrödinger release 2022-2, 2021.

(34) Velaparthi, U. et al. Discovery and initial SAR of 3-(1H-benzo[d]imidazol-2-yl)pyridin-2(1H)-ones as inhibitors of insulin-like growth factor 1-receptor (IGF-1R). *Bioorganic & Medicinal Chemistry Letters* **2007**, *17*, 2317–2321.

(35) Chiu, Y.-J.; Hour, M.-J.; Jin, Y.-A.; Lu, C.-C.; Tsai, F.-J.; Chen, T.-L.; Ma, H.; Juan, Y.-N.; Yang, J.-S. Disruption of IGF-1R signaling by a novel quinazoline derivative, HMJ-30, inhibits invasiveness and reverses epithelial-mesenchymal transition in osteosarcoma U-2 OS cells. *Int J Oncol* **2018**, *52*, 1465–1478.

(36) Poirel, L.; Naas, T.; Nicolas, D.; Collet, L.; Bellais, S.; Cavallo, J.-D.; Nordmann, P. Characterization of VIM-2, a carbapenem-hydrolyzing metallo-ß-Lactamase and its plasmid- and integron-borne gene from a pseudomonas aeruginosa clinical isolate in france. *Antimicrob. Agents Chemother.* **2000**, *44*, 891.

(37) Samuelsen, Ø. et al. ZN148 is a modular synthetic metallo-ß-Lactamase inhibitor that reverses carbapenem resistance in gram-negative pathogens in vivo. *Antimicrob. Agents Chemother.* **2020**, *64*, DOI: 10.1128/AAC.02415-19.

(38) Lippert, T.; Rarey, M. Fast automated placement of polar hydrogen atoms in protein-ligand complexes. *J. Cheminf.* **2009**, *1*, 1–12.

(39) Bietz, S.; Urbaczek, S.; Schulz, B.; Rarey, M. Protoss: a holistic approach to predict tautomers and protonation states in protein-ligand complexes. *J. Cheminf.* **2014**, *6*, 1–12.

(40) Åsmund Kaupang macha: MAnual CHArmm, 2022.

(41) Yusuf, D.; Davis, A. M.; Kleywegt, G. J.; Schmitt, S. An Alternative Method for the Evaluation of Docking Performance: RSR vs RMSD. *J. Chem. Inf. Model.* **2008**, *48*, 1411–1422.

(42) Lüroth, J. Vergleichung von zwei Werthen des wahrscheinlichen Fehlers. *Astr. Nachr.* **1876**, *87*, 209–220.

(43) Shirts, M. R.; Chodera, J. D.; Naden, L.; Mobley, D. L. AlchemistryWiki `http://alchemistry.org/wiki/Main_Page` (accessed 07/07/2022).