Università
della
Svizzera
italiana

**Software
Institute**

Design and Implementation of a Visual and Interactive Web Platform for Drug Binding Prediction Pre-Processing

Andrea Brites Marto

September 2024

Supervised by
Prof. Dr. Michele Lanza
Prof. Dr. Vittorio Limongelli

Co-Supervised by
Dr. Roberto Minelli
Dr. Stefano Raniolo
Daniele Angioletti

Abstract

In pharmaceutical research, drug development spans over many years of studies. To speed up the discovery process and reduce the costs, drug binding affinity is crucial as it directly correlates with the strength of the binding between the drug and a target.

To predict the binding, several computational techniques have been developed, each with different levels of accuracy and performance. The aim of these techniques is to compute an estimate of the binding free energy, or else the affinity between a ligand with its target, by evaluating the interactions that they can establish. In particular, in computational chemistry molecular dynamics—through state of the art computational binding free energy methods such as Funnel-Metadynamics (FM) are employed. However, these methodologies often present several design limitations which limit the scope and practicality of their usability. For example, the tooling support for the FM is legacy code with a considerable amount of technological debt and poor documentation, which renders it difficult to support and to deal with. Furthermore, the currently available implementation is unintuitive and cumbersome to use.

In order to address these issues, we present a new web-based software solution for the first step of the FM protocol. The system allows the loading and storage of the biological configuration of both the ligand and the target, as well as all the relevant parameters, to create the necessary inputs to perform FM simulations. To achieve this goal, we first focus on designing the application. Finally, we develop a web-based platform to enable an entry point for the entire system and allow to interactively work on the FM protocol.

"Make everything as simple as possible,
but not simpler."
Albert Einstein

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Dr. Michele Lanza and Prof. Dr. Vittorio Limongelli, for giving me the opportunity to work on this thesis and for their guidance and teaching throughout the project. In particular, I appreciated how both of you constantly inspired me to keep working, with new great ideas for improvements every time.

Secondly, I want to express my gratitude to Dr. Roberto Minelli for encouraging me to do this interdisciplinary master thesis.

An immense thanks to Dr. Stefano Raniolo, who assisted me with continuous feedback on the software during the development and also for supporting me with invaluable feedback on the thesis report. I really enjoyed the meetings with you because I learned a lot in the molecular life science field. Thank you again. I also want to thank Daniele Angioletti for his help throughout the thesis, being from a similar background, we both could understand each other very quickly and this helped me a lot.

To my dear friend Gianmarco De Vita, you are an amazing person, I cannot thank you enough. Your assistance was unbelievable, as you motivated me, helped me, and, most importantly, were honest and clear with me in everything.

To my family, who supported me and gave me the will and the strength to become what I am today. To my friends Sasha, Stefano and Samantha, who boosted my morale during hard times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the main focuses of drug design studies is the understanding of the way a drug binds to its target and the assessment of the *affinity* between them which is defined as a measure that describes the strength of the interaction between two or more molecules that bind reversibly. In the past, these challenges were bestowed upon experimental assays, which are in general costly and prodigal since they require high-throughput screening of different ligand-target complexes. A more friendly and inexpensive way to gain this knowledge allows us to speed up the early stages of the drug discovery and development process. As shown in Figure 1.1, the method leading to the development of a drug consists of five successive stages:

1. In the *Pre-discovery* stage, researchers identify the disease through clinical observations aiming to understand the molecular mechanisms involved and discover potential molecular targets.

2. *Drug Discovery* focuses on identifying therapeutic targets associated with the disease and finding therapeutic agents that can modify the function of these disrupted targets using in silico[1] and in vitro experiments[2].

3. The *Preclinical Studies* is where drugs are tested in various animal models (in vivo [3]) to evaluate their toxicity and viability.

4. *Clinical Studies* is a stage during which the drug is tested on a ever-increasing number of humans (starting from healthy individuals and gradually switching to clinical cases) to determine its safety and efficacy, ensuring the highest benefit-to-risk ratio for patients.

5. *Approval* is where the study results are submitted to regulatory agencies for review. These agencies evaluate the documentation and decide whether to approve the drug for market release. This entire process, from *Pre-discovery* to *Approval* and the eventual release of the drug in the market, can take up to 10 to 15 years and this makes it an important investment [1].

Previously, there have been exceptions to this thorough procedure, for instance during the COVID-19 pandemic and its race to find drug candidates or seasonal vaccination. However, accelerated procedures are closely regulated and sometimes viewed unfavourably by the community.

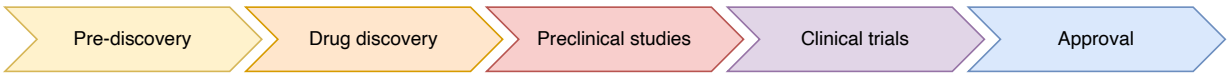


FIGURE 1.1: Drug development process.

[1]Computer-based bioinformatics and/or computational chemistry algorithm

[2]Experiments performed with microorganisms, cells obtained from tissues of infected organisms, or 3D models of engineered tissue, or biological compounds isolated from their biological surroundings.

[3]Studies performed on living organisms, such as animals.

As shown by DiMasi [2], their study examines financial benefits for drug developers from improvements in the drug development process, quantifying effects on development costs from faster development, earlier failure decisions, and higher approval success rates. Utilizing data from a study on R&D costs for new drugs, the research simulates potential cost reductions. They discovered that enhancing drug development efficiency through faster development times, quicker termination decisions, or higher success rates—whether via public policy, better management, or new technologies—can substantially reduce R&D costs. This increased efficiency may lead to more innovations and quicker availability of new therapies.

In this scenario, it is possible to intervene during the *Drug Discovery* stage, where we can employ increasingly reliable in silico techniques to reduce in vitro assays' experimental costs by reducing the number of compounds that must be synthesized and tested in order to have the final drug candidates [3]. In this regard, once the target of the drug has been discovered in the *Pre-discovery* phase, and we have several potential drugs, i.e. small molecules, we can predict how these two entities bind together. This prediction is called the drug binding affinity, in which we can understand how the drug interacts with the target, the strength of the binding, and potentially the toxicity. Therefore, it is a fundamental step in drug discovery. For a better understanding of the document, three distinct definitions need to be considered:

(i) **binding site** which is the region of the target molecule where the ligand binds.

(ii) **binding mode** which represents the ligand in its binding conformation in the binding site.

(iii) **binding mechanism** defined as the path traversed by the ligand to reach the binding site.

Historically, drug efficacy has been measured by affinity for specific molecular targets, focusing on thermodynamic parameters, such as dissociation constant, inhibition constant, half maximal effective concentration (EC50), half maximal inhibitory concentration (IC50), and kinetics rate measurement [4]. Consequently, numerous experimental techniques have been developed. Recent drug discoveries emphasize the importance of kinetic properties, leading to a focus on optimizing association and dissociation rates. Current methods measure these rates by monitoring a specific signal over time. Nonetheless, experimental methods take time and are expensive, not considered the possibility for errors since these studies are done in a laboratory. As a result, computational methods have been increasingly utilized to complement the limitations of experimental methods, filtering the number of drug candidates and accelerating drug discovery. This is generally done by calculating the free energy of binding, a crucial thermodynamic quantity for understanding complementarity of chemical species and identify in the process how they associate or react. [4] This physical quantity, also referred to as the binding affinity, can be estimated using a plethora of free energy calculations of various accuracy. However, the higher the accuracy; the more demanding the computational time due to the intensive molecular simulations required and proper setup to study case by case, including the parameters to run those simulations properly. The complexity of these computational techniques can lead to potential errors and a considerable waste of time when an improper configuration is applied, potentially requiring a level of experience that not all user might have. Thus, limiting the whole process to be used and having a rather complex protocol to follow does not make them financially profitable for pharmaceutical companies. In this regard, the scientific community created advanced methods to mitigate the aforementioned problems. Among these, Funnel-Metadynamics (FM) [5] is a widely recognized computational method employed in the field of computational chemistry and molecular dynamics simulations. Its primary purpose is to accurately estimate the binding free energy of a ligand, defined as the free energy difference between the bound[4] and completely unbound states[5], disclosing the main binding conformations, and providing qualitative information about the kinetics of the binding mechanism [6]. This is extremely important because in a single calculation we can obtain substantial information to better understand the target-ligand interaction.

[4] target-ligand complex formed

[5] ligand and protein free in the solvent

As shown by [5], the FM is a binding free energy method employed to predict the thermodynamic properties of the binding mechanism of a drug to its molecular target, combining a metadynamics bias potential with a funnel-shape potential restriction. To accomplish this, we need the ability to position a funnel-shaped filter in the three-dimensional space over the target molecule such that the region on which we want the restriction is well-defined. The bias potential used in metadynamics is constructed as a sum of Gaussian functions in the space of the chosen collective variables (CV), a parameter representing a determined degree of freedom (i.e., an independent metric that describes a state of a physical system subject to study). Although, in FM simulation this force is applied to the target structure and when the ligand approaches the edge of the funnel, a repulsive potential is used such that the system is discouraged from visiting the region outside the funnel again. To ease the user experience and promote the technique, FM was introduced in a full-fledged protocol, presenting general information able to lead experienced or inexperienced users to set up a FM simulation [7]. The protocol is called Funnel Metadynamics Advanced Protocol (FMAP), composed of three different phases: *pre-processing*, *simulation*, and *post-processing*. The pre-processing step is of paramount importance, given that possible errors at this stage may jeopardize results obtained later in the procedure due to the positioning of the funnel-shaped filter, which has to be done carefully and in the simplest way with a clear 3D visualization of both the target molecule and the ligand. In that regard, interfaces were created to mitigate the issue with the Funnel Metadynamics Advanced Protocol Graphical User Interface (FMAP GUI) software to tackle the pre-processing and post-processing phase. This remains one of the weaker steps of the methodology as it requires the most steps by the user and is the most prone to error because it could be daunting to follow for initiates, as stated in [7].

Last but not least, the steps are disconnected from each other by switching in and out of different software windows, and if you accidentally stop the program, you need to set all the parameters again. As a result, this turns in a tedious and repetitive task considering the number of parameters required.

1.1 Contributions

In this section, we present the thesis contributions spacing from the design and development of a software solution implemented under a web based approach, its deployment as a tooling support for the molecular simulations that have to be run in supercomputers, and an experimental user evaluation allowing us to get feedback from users belonging to the domain.

As a background, foundational motivation, we base upon the fact that several tools implementing and supporting computational methods tend to have certain issues due to its creators focusing more on the domain specific aspects rather than designing a sound software architecture [8, 9]. In their extensive review, Verma et al. [10] found that although the applications of bioinformatics are significantly growing both in academia and industry, the integration of software engineering practices in the implementation of appropriate tools is not experiencing improvements. In the fields of life sciences, in particular in biology and chemistry, scientists tend to produce software that is more aimed to functionality rather than manageability and user ease of use [11]. Indeed, they often lack of a proper graphical user interface (GUI) [12] or proper optimization that would speed up the calculations and make them portable on different—smaller in scale—hardware [13]. Therefore, the developed tools may suffer in terms of usability, especially for users that are non experts in those particular fields but could take advantage of using such software, like students or inexperienced scientists [14].
FM is part of this class of techniques. Although the `Tcl` interfaces improved its user-friendliness and accessibility to the broader scientific audience between the first version in 2013 and its latest implementation in 2020, they depend on a third-party software called Visual Molecular Dynamics (VMD), and they are not optimised, possibly causing frustration in the user that is forced to use particular software.

Basing upon these premises, the goal is to design and develop an innovative software system. A fundamental step for starting to design the software is to acquire a deep understanding of the methodology

beneath the functionalities we want to implement. Thus, we conduct a full review of the requirements based on the state-of-the-art solutions both on a theoretical and practical level such that we can study novel features and integrate them into the new solution. Once the domain specific requirements are clear, we plan on defining a novel architecture accessible for all researchers in this field through a web platform. Furthermore, the FMAP-GUI will be fully renovated following the expert feedback from past users to improve the workflow with an independent and accurate visualization. Last but not least, the overall goal is to offer a comprehensive tool to aid and guide the user throughout the completion of the protocol. Such implementation could represent the base module for an end-to-end procedure to apply the FM in molecular systems, rivaling state-of-the-art software currently employed in pharmaceutical industries. [15, 16]

1.2 Document Structure

The rest of this document is structured as follows:

In Chapter 2, we provide an overview of the evolution of analytical software in the computational biology field both from a methodological and computational cost perspectives, as well as a comprehensive review state-of-the-art solutions to the problem of drug binding prediction. More in particular, we focus on Funnel Metadynamics (FM) technique allowing to extract various information in a binding process, thermodinamically and kinetically speaking, with manageable computational complexity.

In Chapter 3, we explain the assumptions and the technologies underlying the implementation of our tool, a web-based software tool for protein visualization and principally, among its features, setting up inputs for FM simulation.

In Chapter 4 we describe the architecture of our software, with insights and details on its strengths and on things that could be improved or change over time.

In Chapter 5, we illustrate a sample working functional example exposing the potential of the new tool and reports the design and the results of a user evaluation survey conducted on a sample of experts able to provide feedback on the software tool from a domain-specific perspective.

In Chapter 6, we present an abridgement with highlights of the most relevant results of this work, with a critical focus on its limitations and on future research directions as well as open challenges and possibilities.

Chapter 2

State of the Art

Over the years, during the *Drug discovery* phase of the development of a new drug, experimental and computational techniques have become increaingly accurate in predicting the interaction between ligand and protein. As seen in the introduction, the binding properties are useful information that are used to improve the whole process of drug production and are studied by observing the so called ligand-binding interactions. Ligand binding occurs when a compound (be it a small molecule, peptide, lipid, DNA, etc.), referred to as a *ligand*, binds to a receptor, referred to as a *target*, affecting its behaviour by influencing few key hot spots or outright changing the target conformation. This generally promotes a signal cascade that promotes or disfavour certain cellular mechanisms. Consequently, when we refer to ligand-protein binding (LPB), we are studying the drug mechanism of action, which is the discovery of its molecular target and its binding mechanism. The ligand binding mechanism describes the complete path of the ligand as it transitions between different binding states. The first state is the bound state, where the ligand is close to the protein's binding site. The second state is the unbound state, where the ligand detaches from the target and returns to the solvent. During the interaction between a ligand and a protein, the ligand denoted in Figure 2.1 with **L** can assume different binding poses, where each pose has a related energetic state when interacting with the protein denoted with **P**. The binding poses have different energy states since each pose has its own atom dispositions in space that promotes a different network of bonds.

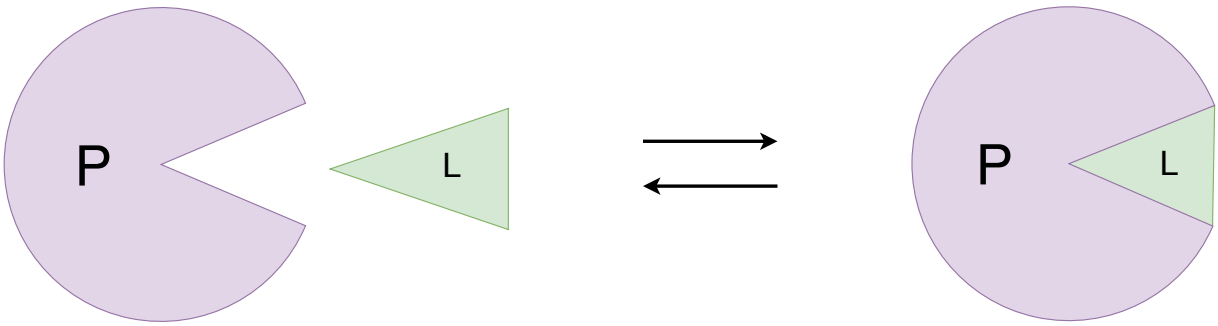


FIGURE 2.1: Ligand Protein Binding (LPB)

A ligand can be a small molecule or even peptide structure that can bind to a specific site in a target, altering its conformation and eventually, its function. As an example, G protein-coupled receptors (GPCRs)—the largest group of membrane receptors in eukaryotes, play a key role in a variety of functions in the human body—are classified as allosteric proteins meaning that the agonist binding, a chemical compound that through a receptor activates a biological response, at its binding site. When a ligand binds with a GPCR it triggers conformational change, and this has an effect on the G-protein bound to the GPCR. The G-protein, known as guanine nucleotide-binding proteins, is a family of proteins devoted to transmitting signals from outside the cell to the inside. There are two classes of these proteins: monomeric small G-proteins and heterotrimeric G-proteins. We are interested in the latter class, which is composed of three different subunits, alpha (G_α), beta (G_β), and gamma (G_γ). When a conformational change occurs in a

GCPR, the alpha subunit dissociates from the beta-gamma dimer ($G_{\beta\gamma}$). Then, G_α and $G_{\beta\gamma}$ can activate the signaling cascade throughout the cell. Proteins attached to receptors and involved in this kind of process are also called effector proteins.

There are two kinds of binding sites. One is the allosteric site, which is a location on the protein different from the active site where a molecule can bind and change the functionality of the protein (e.g., changing the binding site, promoting a given conformation, etc.). This is essential because some drugs may also bind to those modified structures. On the other hand, the orthosteric site is the location where an endogenous ligand can bind. Up until now, we spoke about ligands in a general way but we can now classify them based on their function. Agonists are molecules with the ability to bind in different locations and ways, inducing the receptor to a conformational change that increases its efficacy. Not all molecules can elicit full activity though, and those are called partial agonists. On the contrary, inverse agonists have the opposite effect since they bind with a constitutive active site, stabilizing though an inactive conformation and thus, reducing its activity by blocking the signal transmission. Finally, the last class is the antagonist class, which locks the binding site and stabilizes the basal activity of receptor. With this activation and signaling process from drug to protein, when we analyze ligand-protein binding, we need to focus on three different definitions related to binding. Firstly, the binding mode includes the position of the ligand within the binding site relative to the receptor and the specific residues or functional groups on the target molecule that interact with the ligand, which is the conformation when bound to each other. Secondly, the binding site, also known as an active site or binding pocket, can be rearranged to better suit the binding modes of different compounds with high affinity. Last but not least, the binding mechanism and its interactions with its target molecule to form a stable complex. Understanding the binding is essential for designing drugs with optimal binding affinity, selectivity, and efficacy.

For this reason, the effectiveness of a drug is assessed based on its binding mechanism for specific molecular targets. As introduced by Bernetti et al. [4], various methods and practical strategies have been developed for measuring thermodynamic parameters, such as the equilibrium dissociation constant (Kd) and the half-maximal inhibitory concentration (IC50). Consequently, numerous established experimental techniques are now available for addressing various specific problems. Recent advancements in drug research have highlighted the importance of characterizing kinetic properties when identifying suitable novel drug-like ligands. Current methods for measuring these parameters involve monitoring a specific signal over time in response to the binding event. The literature outlines several experimental techniques for determining binding kinetics, which can be categorized into three classes: (a) assays using a label for detection, (b) label-free techniques, and (c) enzyme activity-based assays. In general, experimental techniques have limitations to which computational can overcome. Regarding the computational methods, molecular dynamics (MD) simulations are employed to measure the drug binding kinetics. Over the years, MD simulations have become a primary tool to animate biomolecular structures, providing insights into the natural dynamic behavior of biomolecules in solutions across different timescales. Existing experimental methods measure kinetic quantities but do not provide a mechanistic interpretation of the underlying processes at the atomic level. In that regard, simulations which capture all configurations of a system over time at atomic-level detail are well-suited for reconstructing the kinetics of biomolecular processes. The underlying assumption of MD simulations is the ergodic hypothesis, formulated in equation 2.1.

$$\bar{A}_i(r) = \langle A_i(r) \rangle_{NVE} \tag{2.1}$$

This hypothesis states that the average of a certain observable over a time period of $A_i(r)$ is equal to its microcanonical ensemble—i.e. statistical ensemble[1] for which the total number of particles in the system (N), the system's volume (V), as well as the total energy in the system (E), are assumed to be constant. Simply put, the probability of discovering a state depending on its energy aligns with the average of all the various conditions and states observed throughout a sufficiently long time, thus implying that the average behavior

[1] Ensembles are statistical models consisting of a set of particles that together attempt to describe the structure of a system.

of the system over time matches what is typically observed in the system as the average over the statistical ensemble. More importantly, it should be independent of the chosen initial state, meaning that assuming this hypothesis holds, over some time molecular dynamics simulations that started from different starting positions will experience the same averaged behaviour. This is important since the validity of the results obtained by a simulation is guaranteed by the fact that the ensemble system pondering all possible states is statistically equivalent to a system evolving through the different states. Thus, the ergodic hypothesis allows us to compute thermal averages of molecular properties through MD simulations, computed as the sum of all the physical quantity values at different energy levels weighted by their respective probabilities of occupation. By simulating a molecule and its surroundings over a period, time-averaged properties closely resemble experimentally measurable ensemble averages. This approach is valuable for calculating bulk properties of fluids and free energy differences for processes such as ligand binding [17]. Using a potential energy function, MD simulations evolve molecular systems by integrating Newton's equations of motion for all atoms in the model. By calculating the forces between pairs of atoms and repeating this process for a series of timesteps, researchers generate numerous configurations over time, known as an MD trajectory. Below, we review how these computational methods are used to determine the kinetic properties.

As presented by Limongelli [18], the first class of techniques is the direct estimation of rate constants via brute-force sampling, which estimates rate constants by counting the number of transitions between stable states divided by the effective time spent in each state. The mean residence time, or the average time the system remains in the initial state, is the inverse of the rate constant. The mean first passage time is the time it takes for the system to enter a second state from the initial one. Assuming instantaneous transitions, the mean residence time equals the mean first passage time. Biologically relevant events occur on much longer timescales than typical MD timesteps, making sampling computationally demanding. To reliably estimate rate constants, numerous transitions must be observed, a requirement often not reachable in dynamic docking simulations. Dynamic docking flexibly explores protein-ligand binding without rigorously determining thermodynamics and kinetics, making it suitable for industrial applications. This method that is based on MD, accounts for the full conformational flexibility of molecular species, overcoming the limitations of rigid molecular docking methods.

Moreover, we have the Markov state model and adaptive sampling approaches, which instead of analyzing full long trajectories, take many shorter trajectories and build a Markov state model (MSM), running multiple short simulations in parallel. More specifically, when constructing the MSM, it determines the transition probability of the `n` sampled microstates. A microstate is recognized based on a geometrical clustering of the configuration sampled, and in the decomposition phase, each configuration is appointed to a microstate. For this reason, it was named adaptive since the model can improve progressively.

The third class is about Brownian dynamics (BD) and multiscale modeling. This is a normal MD simulation in which each atom of the system evolves over time following Newton's second law of motion. However, the dynamics of a solute in a solvent can be defined by the Langevin equation. More generally, with this equation, we reduce the computational cost since the solvent will be represented by the implicit model of solvation, and only the solute of the system will be simulated at an atomistic level. The solute is treated as a fixed body allowing only translations and rotations, and, with this negation of internal degrees of freedom, larger timesteps and higher performance are provided. Lastly, we have the class of Free-Energy surface–based biased approaches, which are MD-based methods to ameliorate the exploration of the configuration space at a fully atomistic level. They are referred to as biased MD, which by introducing external forces, the observation of rare events is encouraged, such as Umbrella sampling (US) and Metadynamics (MetaD). The goal of these methods is to reconstruct the free-energy surface (FES) or to accelerate the sampling of slower processes.

In order to obtain significant results, all of these methods running MD simulations computational requirements are highly demanding in terms of time, as pointed out by Barducci et al. [19], since the simulation can be considered reliable when all possible relevant states have been covered by the run. Metadynamics introduces a method to improve sampling in molecular dynamics simulations, since in a standard MD simulation the time step is in the order of 10^{-15} seconds (femtoseconds) whereas to sample a protein requires the order of 10^{-4} seconds, that would require 10^{11} MD steps. This is not affordable due to the time required for computation even for high performance hardware and the cost of the energy consumption.

Figure 2.2 illustrates as a Gantt diagram the time scales of the different physical and biological processes that the shown computational methods can reach. As we can observe, enhanced sampling techniques allow the study of the drug binding [20].

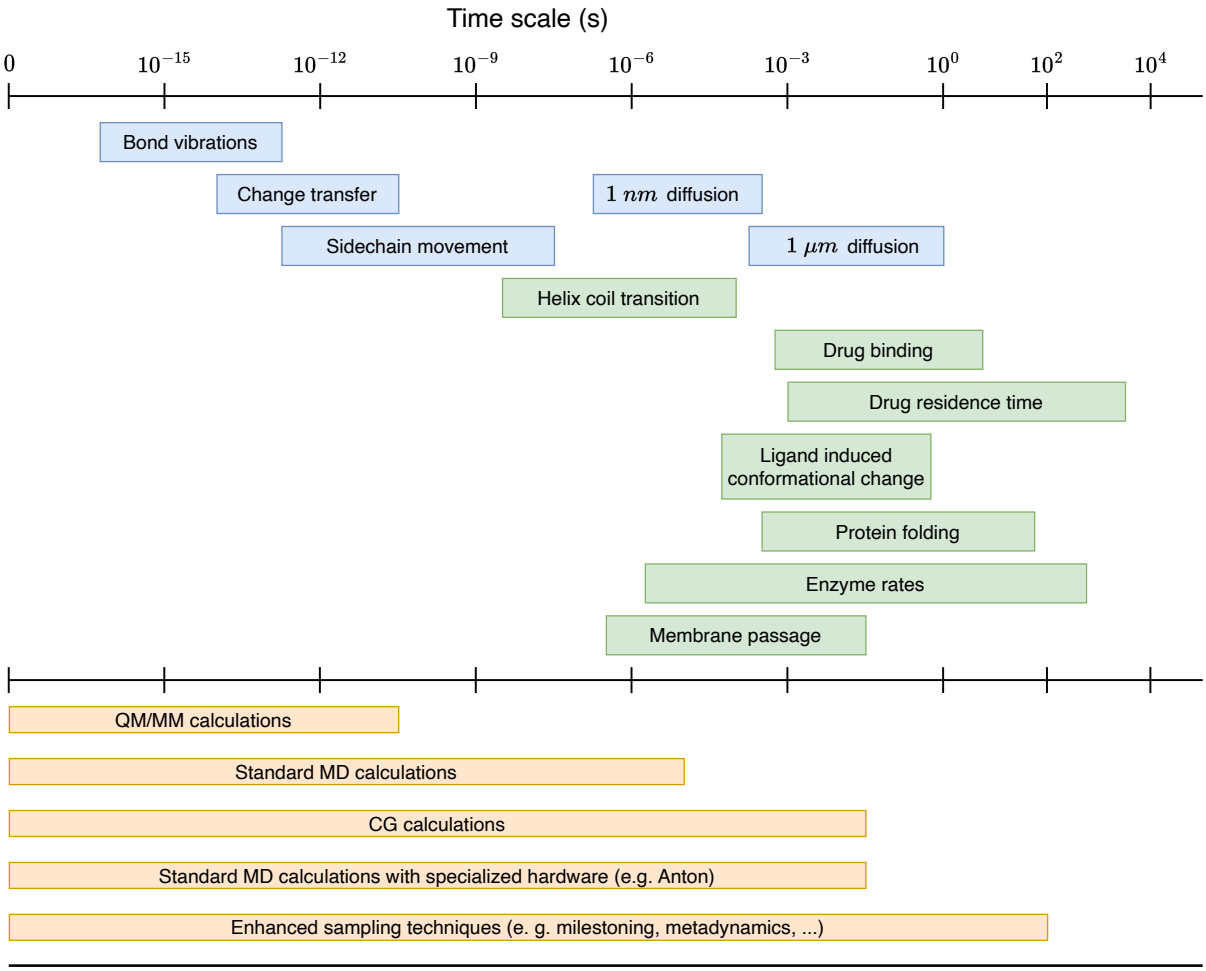


FIGURE 2.2: Covered time scales of physical, biological processes and computational techniques.

As shown in Figure 2.2, blue boxes refer to physical processes, green boxes refer to biological processes, and orange boxes represent computational techniques. These boxes aligned with the time scales they cover in their physical and biological processes are in the order of milliseconds and, as stated by Shaw et al. [21], only a few MD runs have reached time scales such as microseconds while the majority of simulations stop at nanoseconds, this is due to the process of computing the motion of all atoms by all other atoms repeatedly on the order of times 10^9 just to cover microseconds. This is why, only the most powerful clusters, supercomputers or specialized ones are needed to perform such simulations.

In recent years, a lot of progress has been made and many methods that are referred to as enhanced sampling techniques have been proposed to address this issue. Metadynamics is a class of methods that enhance sampling by applying an additional bias potential to the selected number of degrees of freedom also called collective variable (CV) [22]. Generally speaking, collective variables are parameters that represent a particular structural state or property of the system which can be measured throughout the simulation, such as a combination of distances, angles, and dihedrals. In addition, when we need to observe a target sampling, we can reduce the dimensionality of variables by introducing CVs. Since these states are rare, we can favor the sampling during the simulation by applying some bias (enhanced sampling). With good CVs, we can discriminate essential states to which we are interested in and thus, achieving results on very slow processes in the system. Furthermore, the CV is very important since it can give key information about the system if chosen correctly. The first version of metadynamics introduced a bias to influence coarse-grained dynamics in the collective variables space, based on a series of constrained MD simulations. In a later version, the bias was used in every step of the MD simulation by applying an extended Lagrangian formalism [23] or on the microscopic coordinates of the system.

2.1 Funnel Metadynamics

In 2013, Limongelli et al. [5] designed the *Funnel Metadynamics* (FM) technique by developing a metadynamics based approach to predict accurate protein-ligand binding free energy in which they discuss the importance of understanding the interactions between drugs and their target molecules for drug development. It highlights the challenges in accurately estimating the binding affinity of a drug to its target, crucial for processes like drug design and optimization.

Various methods, from empirical scoring functions in docking protocols to more computationally intensive endpoint and pathway methods, have been developed to estimate binding affinities [7]. The distinction lies in the fact that in the first scenario, the system explores specific states chosen by the user, whereas in the other scenario, the system can sample the entire binding mechanism. This is why we have a difference in terms of machine time steps. Additionally, it is possible to assess potential activation energy barriers to move from an energetic state to another and qualitatively gather more kinetic information. In this way, using simulations we are able to acquire kinetics insights between ligand-protein binding. To achieve that, these methods involve techniques that range from gradually separating the ligand from the protein to applying external potential, such as metadynamics, to enhance sampling and compute the free-energy surface.

Current methods lack the ability to accurately determine binding energy due to the rare and complex binding events. The weakness lies precisely in the fact that many of the techniques can measure only singular states and not the overall binding path. Therefore, we can not estimate the process kinetics, which is one of the main parameters when studying the potency of a drug. This led to the proposal of a new approach called "Funnel Metadynamics" (FM). FM aims to overcome these limitations by applying a funnel-shaped restraint potential, significantly reducing the exploration space in the unbound state. This results in enhanced sampling of both bound and unbound states, allowing for a more accurate estimation of the binding free energy within a reasonable simulation time. As a result of the nature of FM, knowing the binding mode in advance is not required since it is guaranteed to be sampled eventually, making FM a suitable technique for almost any case study.

The FM approach was successfully applied to study the binding process of few paramount systems, revealing agreement with experimental data and providing insights into binding mechanisms. [5] Importantly, FM does not require any prior knowledge of the binding mode, it retains exploration capabilities, and can investigate buried binding sites while considering slow protein motions and solvent effects. The method's relatively low computational costs make it appealing for industrial applications where speed is essential.

This approach integrates the MD bias potential with a funnel-shaped restraint directed at the target. The bias potential is formed on the system's trajectory taking into account the space of the selected degrees of freedom (CV) and progressively adapting a constructed sum of Gaussian functions. The restraint is cone-shaped which includes the ligand binding site integrated with a cylindrical shape directed to the solvent making it a funnel-shaped geometry. In general, when the ligand explores inside the funnel-shape area, the system does not get any funnel potential, but when the ligand reaches the limits of the region, a repulsive potential is applied to discourage it from scouting outside the funnel. In this manner, the sampling is accelerated, and multiple binding and unbinding events are observed, leading to a quick convergence of the free energy. As a consequence, we get a well defined binding free-energy surface (BFES) from which we can identify the ligand binding mode represented by the lowest free-energy minimum. From the BFES, we compute the absolute binding free-energy of the protein-ligand ΔG_b^0 formulated in equation 2.2.

$$\Delta G_b^0 = -k_B T ln(C^0 K_b) \tag{2.2}$$

Where k_B is the Boltzmann constant [2]; T is the temperature of the system; C^0 is the state standard concentration of $1M$; K_b is the equilibrium binding constant derived from the difference between the free-energy in the bound and unbound state. Therefore, the binding free energy depends on the free-energy values of these two states (bound and unbound) regardless of the path connecting them.

2.2 Software Evolution in Bioinformatics

The first molecular dynamics computer simulation was developed in the early 1950s with Monte Carlo simulations (MC) and the Molecular Dynamics method (MD). With the progress in computer power and speed, computers became more available to all researchers who started to publish various simulation results. Many other algorithms, for instance the Verlet algorithm, were invented to speed up the calculations. As molecular dynamics techniques advanced, the software evolved with it and many programs started to appear to perform such tasks. The first program was CHARMM (Chemistry at HARvard Macromolecular Mechanics) developed in 1982 and after that, another important software was published which is still the most widely used nowadays, GROMACS [24] (GROningen MAchine for Chemical Simulations). Several programs can be used to run and analyze MD simulations such as NAMD [25] (Not (just) Another Molecular Dynamics program), AMBER [26] (Assisted Model Building with Energy Refinement), or DESMOND [15]. To gain more insights while performing simulations with molecular systems, an open source plugin known as PLUMED [27, 28, 29] has been developed by Bonomi et al. [29] that is designed for free-energy calculations. The program consists of a set of routines that can be integrated with well-known classical molecular dynamics (MD) code by patching them. This flexibility enables users to select from a range of MD engines depending on the particular system being simulated and the computational resources available. The program enables free-energy calculations using multiple collective variables, with a focus on biological applications. It is written in ANSI-C and is compatible with both Fortran and C/C++ code. The FM protocol implementation is based on PLUMED, which includes the code to compute free energy landscapes, following the theory behind the FM simulation.

As discussed by Neerincx and Leunissen [30], bioinformaticians often need to link many data sets and tools in order to understand biological systems due to their complexity. Therefore, to integrate all these data sets and tools, in recent years web servers started to become popular with web-based interfaces to overcome the problem of interoperability. Thus, an evolution of tools paired with increasingly sophisticated web interfaces, ranging from basic HTML-based user interface to web services able to perform pipelines to ease the workflow, emerged.

Already in 2009, the European Bioinformatics Institute (EMBL-EBI) stated that the adoption of web service technologies had significant advantages for the common life science scientist such as not having to

[2] The Boltzmann constant (k_B) links the average kinetic energy of particles in a gas to the temperature of the gas.

invest too many resources on the installation, maintenance and execution of all the variety of tools [31]. Furthermore, these technologies can simplify the functionalities and can be easily accessed. Their user interface is simple and efficient, their interoperability is simplified and they are multiplatform. The disadvantage is the dependency on an Internet connection [31] that can affect the user experience for instance due to bad connection or loss of connection which may potentially lead to loss of processes' advancement and the inability to work.

2.2.1 Evolution from a Software Engineering Perspective

Besides that, progresses in the sophistication of the quality of the experimental results derived from the simulations run thanks to those tools, do not directly correlate with the improvement of the implemented software engineering practices. For instance, according to Verma et al. [10] users in this field tend to trade off structure software quality for just obtaining "something that works". Among the causes for that, the authors address some restraints on financial budget for development and timetables, as well as resources allocated to the software verification and testing. Also, they implicitly claim that most of the effort and resources flow into the assessment of the reliability of the experimental results, as incorrect outcomes would negatively affect the validity of the implemented experimental framework.

Likewise, Umarji et al. [32] mention among the main causes the way the software is designed and developed. Furthermore, they individuated some patterns across the two different ideal groups of developers—namely, software engineers and bioinformatics programmers—with the former being more prone to use eXtreme Programming (XP) practices according to the recommendations of relevant literature. They base this claim upon the fact that computational biologists apply encapsulation and polymorphism significantly less than professional programmers. As a consequence, they claim that bioinformaticians do not employ the full advantages of object-oriented programming. Additionally, the authors further explored how software evolution and maintenance is perceived among the two categories of programmers, with biologists being unaware of the relation between change-prone software and age (software evolution). In this regard, the authors studied the importance of software documentation in bioinformatics software. Generally, literature states that biology-related software is rather complex, with many end-users and it falls in a constantly evolving field, thus source code documentation is deemed to be very important. Upon that, the authors found out that many bioinformatic developers acknowledge the importance of documentation, recognizing it both as crucial and beneficial. However, they stress the fact that the most frequent approach undergone by bioinformaticians to structure the documentation of their software follows the style of "User Guides", a kind of technical manual dedicated to support users in using an application tool, therefore resulting in a lack of clear system and software documentation. Similarly, Rother et al. [33]—after conducting a survey over 22 bioinformatics and biomedical projects—point out the lack of formal training in programmers within these domains with respect to good software implementation and documentation practices and mention the unsystematic development pipeline as the main cause for the resulting scarce software maintainability. The integration of Agile frameworks in the architectural design and development of biological software is well-received and encouraged by scholars [34, 35], drawing out parallelism between the iterative natures of both the framework and the domain-specific scientific approaches they are implementing.

Besides the problems related to the development and maintainability of the implemented software tools, relevant research also focuses on the end-user perspective. In particular, we mention the work of Bolchini et al. [36], in which they studied the efficiency and the effectiveness of the usability of bioinformatics tools on the researchers when they interact, share and work with biological information. They found that user experience impediments can slow down researcher activities. The study results depicted a set of usability issues that involved in the design of the navigation and search process. Particularly, they denote a major issue from the users to the ability to find the information they need to complete their tasks. To that end, the authors stress on the growing need in ensuring high standards of usability. In a similar way, other authors [37, 38, 39, 40] assess that bioinformatics software tools have misleading interfaces to use

and navigate for which they propose metrics and methodologies such as user-centered-design (UCD) to mitigate this problem.

Finally, from this survey of the relevant literature we can derive that authors generally agree on the fact that the development of effortlessly maintainable and well-documented software with a user-friendly interface is a primary need in bioinformatics.

2.2.2 From a Data Engineering Perspective

While software plays an important role in the bioinformatic field, so does data management. In the era of open science, the key to gaining knowledge discovery relies on good data management in terms of data integration and data reusability. In this regard, Wilkinson et al. [25] emphasizes a growing requirement from the science community for improving robust data management and stewardship plans. This does not only consider the proper collection, annotation, and archival but includes the concept of the long-term management of digital information. The outcome of such advancements, are high quality publications that ease the process of discovery, evaluation, and reuse in following studies.

To this end, the authors define and clarify what *good data management* constitutes with four fundamental principles, introducing the so called *FAIR data principles*: **Findability**, **Accessibility**, **Interoperability**, and **Reusability**. In detail, **Findabiltiy** is the principle that assigns a globally unique and persistent identifier, described with rich metadata. The latter must clearly include the identifier of the data it describes. Additionally, both data and metadata should be indexed in a searchable resource to ensure easy discovery and access. The **Accessibility** states that data and metadata should be retrievable using a standardized, open, and universally implementable communications protocol that supports authentication and authorization when needed. Moreover, metadata should remain accessible even if the data is no longer available. While data has to be accessible, it should have the **Interoperability** principles. On this matter, data and metadata should use a formal, accessible, shared, and widely applicable language for knowledge representation. In addition, employ vocabularies that adhere to FAIR principles and include qualified references to other related data and metadata. Finally, the **Reusability** principle where data and metadata should be richly described with accurate and relevant attributes, released with a clear usage license, associated with detailed provenance, and meet community standards relevant to their domain.

The FAIR principles are correlated but independent guidelines intended to strengthen the discovery and reuse of data resources, tools, vocabularies, and infrastructures. With the minimum application of these principles, the entry barrier for making data FAIR for data producers is low. Consequently, the principles can also be applied incrementally and in various combinations as publishing environments evolve incrementing the degrees of *FAIRness*. More importantly, the FAIR principles are not a standard or specification but rather a model for data producers and stewards to guide their implementation choices. For instance, the *wwPDB*[41] is a web server that hosts information on 3D structures of proteins and nucleic acids determined by experiments. Ultimately, from *FAIR* principles we derive that there is a general agreement on having well defined data structures and infrastructure to allow better discovery processes.

2.3 FM Advanced Protocol (FMAP)

FM simulation is a complex computational technique that involves many steps, configurations, and parameters. For this reason, there is a protocol named FM Advanced Protocol (FMAP), which provides a well-defined step by step guideline and a flexible user-friendly graphical user interface (GUI) to perform funnel metadynamics. We present a detailed overview of the FMAP, a workflow with three different main subsequent phases: *pre-processing*, *simulation* and *post-processing*. In Figure 2.3 we represent the workflow of the Funnel Metadynamics Advanced Protocol.

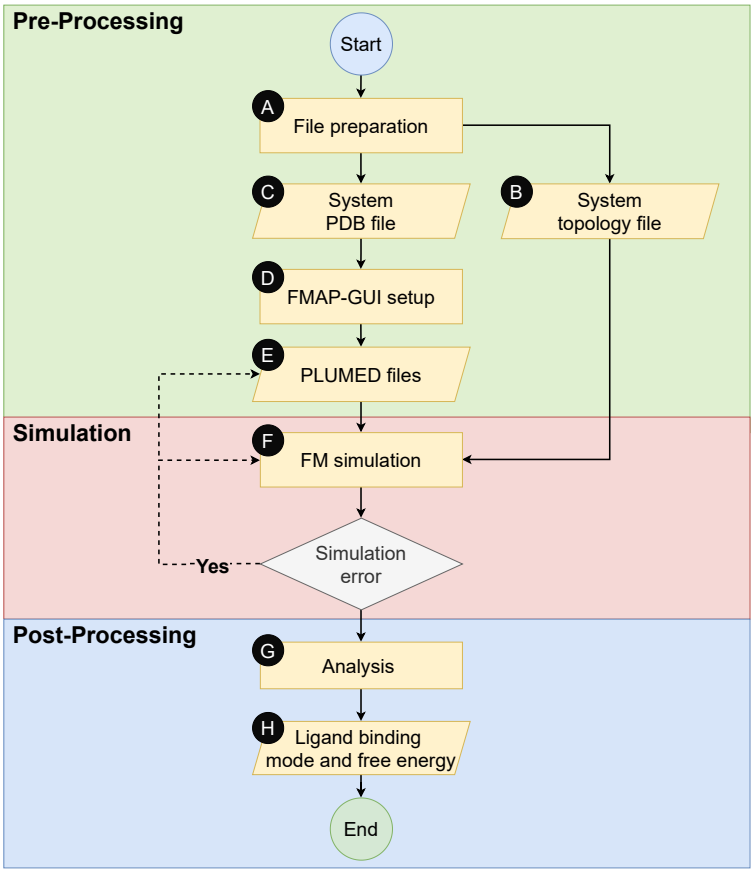


FIGURE 2.3: Flowchart representation of the FM protocol.

Figure 2.3 **A)** shows that begin of the pre-processing phase of the FM calculation with the preparation of the files needed for the protocol, as showed in Figure 2.3 **B)**, the system PDB file which represent the molecular information of the ligand and the target, and Figure 2.3 **C)**, the molecular topology specification for the system under study. Figure 2.3 **D)** involves creating a funnel-shaped potential, achieved through the FMAP GUI, providing a Graphical User Interface, allowing the end user to load the initial structure of the system and define the Cartesian coordinates for placing the potential funnel in the system. Afterwards, this phase ultimately generates a customized PLUMED input file for FM by incorporating user-defined options as shown in Figure 2.3 **E)**. In the simulation phase, Figure 2.3 **F)**, we provide instructions for conducting the funnel metadynamics (FM) simulation using one of the following three different molecular dynamics (MD) engines. Within the simulation folder, the automatically generated 'BIAS' file represents the funnel-shaped potential defined in the pre-processing phase. In Figure 2.3 **G)**, it begins the post-processing phase where users analyze funnel metadynamics (FM) results to retrieve structural and energetic data on ligand-protein binding. The FMAP-GUI facilitates various post-processing data types as shown in Figure 2.3 **H)** such as creating Binding Free Energy Surfaces (BFES) and collecting system statistics.

2.3.1 Pre-Processing

The pre-processing phase is the first step in which the user has to set up the funnel shape, configuring defined parameters to properly position the funnel in the three dimensional space on the molecular target after generating the topology and coordinates files of the studying system (can be done with GROMACS, NAMD or AMBER). In addition, the user has to define the necessary FM parameters.

The idea of the FM is to apply a funnel shape on the ligand site shown in Figure 2.4 as **L**, on a protein represented by the gray circle **P**.

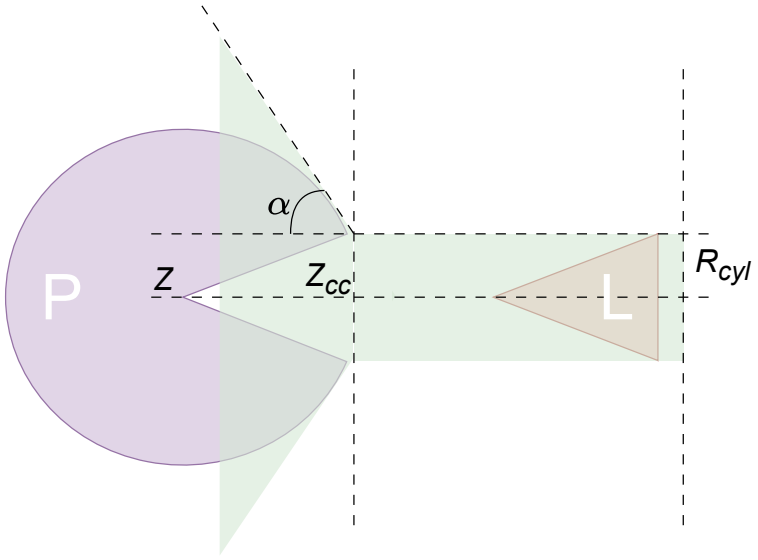


FIGURE 2.4: Funnel disposition and structure as conceived by FM.

Parameter	Unit	Description
Point A	(x_A, y_A, z_A)	Point in the 3-D space setting the origin of the funnel.
Point B	(x_B, y_B, z_B)	Point in the 3-D space setting—together with the origin—the direction of the funnel.
α	Radians	Amplitude of the cone section with respect to the Z_{cc} and R_{cyl} values.
Z_{cc}	Å	Cone and cylinder switching point given by the distance from the Point A.
R_{cyl}	Å	Radius of the cylinder.
Radius	Å	Radius of the cone section.

TABLE 2.1: Parameters to define the funnel.

The position of the funnel shape is done through defined parameters as shown in table 2.1 and it is geometrically modeled as a cylinder shape attached to a cone section. In the planar visualization provided by Figure 2.4, it can be seen that the cone section of the funnel enfolds the binding site of the protein, while the cylinder extends on the same direction of the binding site, but towards the opposite side. In order to set up the funnel, **Point A** acts as the origin for the funnel axis and needs to be specified using three-dimensional Cartesian coordinates. Similarly, **Point B**, also defined in Cartesian coordinates, represents the direction and orientation of the funnel axis relative to Point A. The cone amplitude α, expressed in radians, represents the amplitude of the cone section, and R_{cyl} indicates the radius in angstrom (Å) of the cylinder section for the unbound region. **Zcc**, again in Å, denotes the transition point between the cone and cylinder regions of the funnel. The **Radius** is a parameter that is adapted automatically on changing the **Zcc**, R_{cyl} and α values.

Parameter name	Description
Lower wall	The minimum value that can be sampled as the projection of the ligand's center of mass along the funnel axis.
Upper wall	The uppermost value as projection of the ligand's center of mass along the funnel axis that is sampled.
Min fps.lp	The lowest value for `fps.lp`, used to construct the funnel-shape restraint potential.
Max fps.lp	The highest value for `fps.lp`, used to construct the funnel-shape restraint potential.
Anchor point	An atom of the target molecule used to reconstruct the ligand together with it in the same periodic image.
K_funnel	The value for the spring constant of the funnel-shape restraint potential.
Arg_meta	The list of CVs that will be enhanced by the metadynamics Gaussian functions d1 during FM.
Sigma_meta	The width of the metadynamics Gaussian function.
Height_meta	The height of the metadynamics Gaussian function.
Pace_meta	The deposition stride of the metadynamics Gaussians functions as number of simulation steps.
Biasfactor	The bias factor that is a rescaling factor of the Gaussian function's height when using well-tempered metadynamics.
Grid_min	The lowest value to sample in the CV space.
Grid_max	The highest value to sample in the CV space.
K_uwall	The spring constant for the 'Up. Wall'.
K_lwall	The spring constant for the 'Low. Wall'.

TABLE 2.2: FM simulation parameters.

The FM is provided with a tool named **FMAP-GUI**, which helps the user configure the FM parameters to run the simulation described in table 2.2. Firstly, to run FMAP-GUI it is required to install VMD, a software designed to display and analyze molecular assemblies, with the FMAP-GUI and PLUMED binary files. Secondly, by opening the VMD console it can launch the FMAP-GUI with the following commands:

- Load *funnel.tcl* script into VMD command: `source /path/to/file/funnel.tcl`
- Run *funnel.tcl* script command: `funnel_tk`

This opens the Funnel Metadynamics Advanced Protocol GUI (FMAP-GUI) with a funnel in a default position on the VMD interface. Figure 2.5 shows the FMAP-GUI. Figure **4a)** employs option keys for dynamically adjusting the positions of points A and B by selecting an atom on the VMD interface. In Figure **4b)**, the position and orientation of the potential funnel can be interactively modified either by editing the values in the displayed boxes or by using the arrows. The current coordinates of points A and B are provided in corresponding boxes.

Moving to Figure **4c)**, it is dedicated to customizing the shape of the funnel potential on the target structure. This involves adjusting the switching point (in Å) between the cone and cylinder sections (Zcc), the angle (in radians) defining the section cone amplitude, and the radius (in Å) of the cylinder section (RCyl). Achieving optimal funnel positioning is facilitated by seamlessly switching between invisible and transparent visualization modes.

The inputs in Figure **4d)** enable the implementation of lower and upper walls for the projection of the ligand's center of mass along the funnel axis. Figure **4e)** allows users to specify the 'Min fps.lp' and 'Max fps.lp' values, representing the lowest and highest projections of the ligand's center of mass along the funnel axis, essential for generating the funnel-shaped potential.

In Figure **4f)**, users can establish an anchor point on the target structure to address issues related to periodic boundary conditions (PBC) during simulation. This anchor point should be positioned as close

as possible to the ligand. Figure **4g)** provides a dropdown menu with various options for preparing the PLUMED input file. Each entry is documented in the PLUMED input displayed in the canvas below, adhering to the format and unit of measure of PLUMED.

For reference, an example of the PLUMED input is available in Figure **4h)**. Users can scroll down with the mouse to read the entire text and make modifications at their convenience. In this section, the user can define the remaining FM parameters, **K_funnel**, **Arg_meta**, **Sigma_meta**, **Height_meta**, **Pace_meta**, **Biasfactor**, **Grid_min**, **Grid_max**, **K_uwall** and **K_lwall**. Table 2.2 provides an extensive description of these simulation parameters. The PLUMED input can be saved in a file using the 'Export' in the dropdown menu with the button Figure **4i)**, while the GUI can be reset to default values using the 'Reset' option in Figure **4j)**.

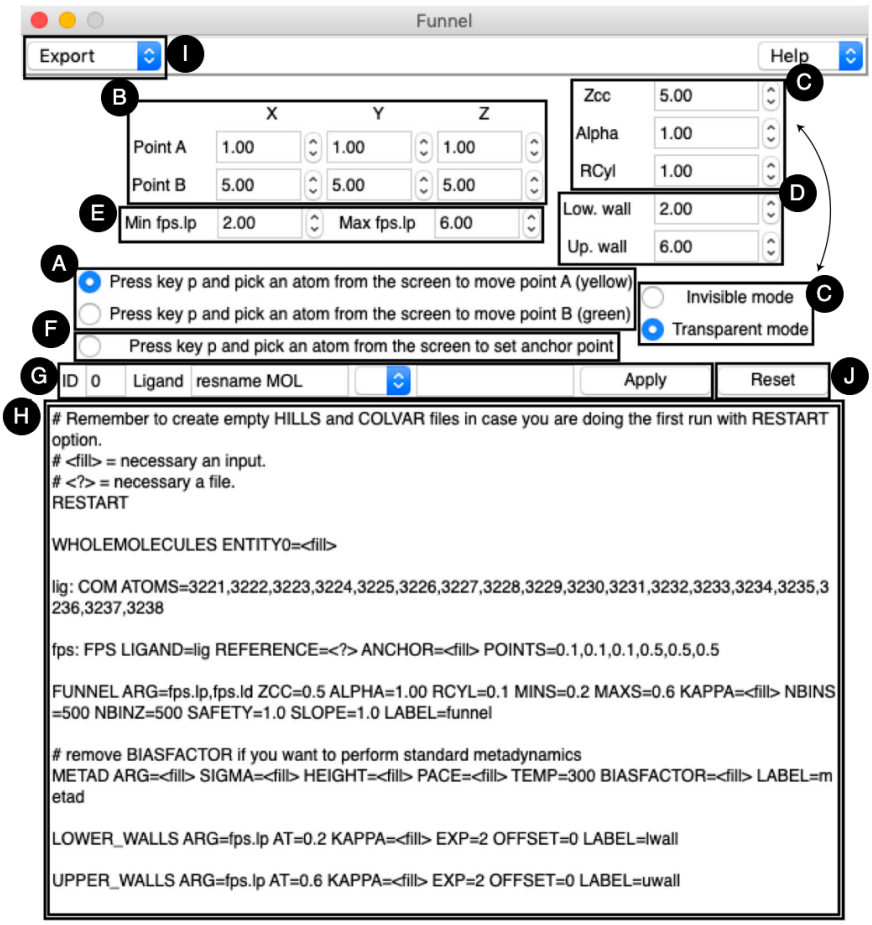


FIGURE 2.5: FMAP-GUI

2.3.2 Simulation

In the simulation phase, we can run the FM simulation with three different molecular dynamics (MD) engines: Gromacs, Amber and NAMD. As already mentioned, the MD software has to be patched with the PLUMED-2 plugin to perform the FM computations. The FM will produce the file 'BIAS' in the simulation folder and represent the funnel shape potential as defined in the pre-processing phase. During the FM simulation, multiple ligand protein binding and unbinding states are observed. There are two versions of the FM. In FM version 1.0, the target molecule was restrained in the three-dimensional space to allow the application of the funnel potential on the system. In version 2.0, the protein can freely move in the

Cartesian space inside the funnel considering the roto-translational entropy of the target molecule. This was implemented by applying the alignment matrix to automatically replace back to the original position.

2.3.3 Post-Processing

In the post-processing phase, users analyze the FM results to extract structural and energetic data on the ligand protein binding process. The FMAP GUI facilitates various types of post-processing data such as generating the BFES, gathering statistics on the system and provides data analysis tools. To perform post-processing analyses, PLUMED-2 binary files are patched into the GUI which allow the computation of the BFES automatically displayed in the GUI where the user can interactively switch between two different views: BFES and system FM trajectory. In this regard, the user can perform several analyses:

- The evolution of the system given by the trajectory expressed in frames by timesteps can then be observed as well as their free-energy values that are interactively available at the BFES.
- An interval of interest can be selected in the BFES view using the right button of the mouse to retrieve the ligand-protein complexes belonging to the selected range of interest in the BFES. This region can be used to extract the structures related to the lowest free-energy, which represents the ligand binding mode.
- The GUI displays the final free-energy value after computing the absolute protein-ligand binding free energy by choosing the interval corresponding to the bound state and the reference state representing the unbound state in the BFES. This calculation accounts for the entropy correction resulting from utilizing the funnel-shape potential.
- The free-energy convergence is directly evaluated through the FMAP GUI by computing the absolute protein–ligand binding free energy as a function of the simulation time.

2.4 FM Advanced Protocol Architecture

Figure 2.6 shows the current FMAP architecture describing the overall scheme from the start of the pre-processing from the user to the end of the post-processing. The protocol system has five software components representing all the phases of the FMAP where users have to download each of these software, install, and configure them on their machine.

The first component is Visual Molecular Dynamics [42] (VMD), a software devised for modeling, visualization, and analysis of biological systems such as proteins. In general, it is used to view molecules through the reading of standard files coming from the Protein Data Bank (PDB) and display the given structures. The PDB[3] is an on-line archive containing structural information at the atomic level for proteins and other important biological macromolecule interacting with proteins—i.e., complex large molecules such as carbohydrates, lipids and nucleic acids whose functions again pertain to biological processes—described by scientists outlining the topology of such biological structures through methods and lab experiments such as X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and cryogenic electron microscopy. Each PDB file has a unique PDB ID and contains meta information summarizing the protein, citation information, and the details of the structure solution in the header section. Right after the header section, it starts with a sequence of information dedicated to the atoms of the structure row by row in the coordinates section with the column entries described in Table 2.3. Regarding the PDB information in the scope of the FM, during the pre-processing phase every column is involved in the rendering of the atoms

[3] https://www.rcsb.org/

as well as the protein secondary structures. During the simulation and post-processing phase, this information is also employed but from a computational point of view while computing the free energy in each time step.

Column name	Description	Type
ATOM	Specify the atom type (ATOM or HETATM)	Character
Atom serial number	Atom index in the file	Integer
Atom name	Atom name described in the file (differs from element)	Character
Residue name	Specify the residue name	Character
Chain ID	Identifier for the molecular chain	Character
Residue number	Residue sequence number	Integer
X coordinate	Three dimensional coordinates w.r.t. the x-axis in Å	Float
Y coordinate	Three dimensional coordinates w.r.t. the y-axis in Å	Float
Z coordinate	Three dimensional coordinates w.r.t. the z-axis in Å	Float
Occupancy	Indicate the fraction of molecules that have each of the conformations	Float
Beta factor	Displacement of the atomic positions from a mean value	Float
Element	Specify the element name (i.e. C for carbon)	Character

TABLE 2.3: Coordinates PDB section.

In addition, VMD provides multiple features for rendering and coloring molecules and many molecular representations such as licorice, cartoon, etc. Lastly, VMD can animate and analyze the trajectory of a molecular dynamics (MD) simulation. It is used during the pre-processing by providing the initial reference structure of the ligand with its target, and by loading the plugin `funnel.tcl` where the user can configure the simulation. A `.tcl` file contains the source code to extend VMD features written in Tcl programming language (Tool Command Language), a high-level, interpreted, and dynamic language created in 1988 by John Ousterhout at the University of California, Berkeley. Tcl is a command-driven language with operations written in prefix notation. It supports variadic commands and dynamic redefinition of everything, including control structures. It operates with automatic type conversion for variables, although non-defined variables cause errors. It allows object-oriented programming, event-driven capabilities for sockets/files, and lexical scoping by default. Tcl emphasizes extensibility via various languages, bytecode interpretation, Unicode support, regular expressions, and cross-platform compatibility with GUI integration via Tk.

The output resource comes in the form of a PLUMED file, which is a file with **.dat** extension containing instruction information for the simulation. It is structured by rows where—depending on what kind of simulation the user needs to run—several instructions must be defined. The first row is **WHOLEMOLECULES**, and is used to cause the periodic boundary conditions reconstruction for the atom serials indicated. The next rows are dedicated to the CV selection such as distance, angle, dihedral and center of mass (COM), but there are many others[4]. Each CV has an associated name which is then used for later instruction and a key instruction with **DISTANCE** for distance, **ANGLE** for angle, **DIHEDRAL** for dihedral and **COM** for center of mass. For instance, as shown in 2.1, we can see one distance **d1** and one center of mass of the ligand under the study named as **lig**.

Then, we start with the parameters required for the FM simulation, where we need to specify the ligand's center of mass, the reference file structure (the PDB), an anchor point, which is the atom index representing the closest protein atom to the ligand, and the points denoting the PointA and PointB. Subsequently, the **FUNNEL** argument calculates the funnel shape restraint potential defined on a grid that is read during

[4] https://www.plumed.org/doc-v2.8/user-doc/html/_colvar.html

the setup. The **METAD** is used to perform standard metadynamics on one or more collective variables. **LOWER_WALLS** and **UPPER_WALLS** define a barrier for one or more collective variables, this will limit the explorable region during the simulation. Finally, since we need the measurements to be written in a file we will process later, we need to print them with a user defined value indicating the frequency to write. In this case, it will print the **ARG** argument defined in the **FUNNEL** instruction into the file named **COLVAR**.

LISTING 2.1: PLUMED file structure and syntax.

```
#RESTART

WHOLEMOLECULES ENTITY0=[VALUES]

d1: DISTANCE ATOMS=[ATOM1],[ATOM2]
lig: COM ATOMS=[ATOMS]

fps: FUNNEL_PS LIGAND=lig REFERENCE=[FILE] ANCHOR=[VALUE] POINTS=[VALUES]

FUNNEL ARG=fps.lp,fps.ld ZCC=[VALUE] ALPHA=[VALUE] RCYL=[VALUE] MINS=[VALUE] MAXS=[VALUE]
    KAPPA=[VALUE] NBINS=[VALUE] NBINZ=[VALUE] FILE=[FILENAME] LABEL=[LABELNAME]

METAD ARG=[CVNAMES] SIGMA=[VALUE] HEIGHT=[VALUE] PACE=[VALUE] TEMP=[VALUE] BIASFACTOR=[
    VALUE] LABEL=metad GRID_MIN=[VALUE] GRID_MAX=[VALUE] GRID_WFILE=grid_w.dat
    GRID_WSTRIDE=[VALUE]

LOWER_WALLS ARG=fps.lp AT=[VALUE] KAPPA=[VALUE] EXP=[VALUE] OFFSET=[VALUE] LABEL=lwall
UPPER_WALLS ARG=fps.lp AT=[VALUE] KAPPA=[VALUE] EXP=[VALUE] OFFSET=[VALUE] LABEL=uwall

PRINT STRIDE=[VALUE] ARG FILE=COLVAR
```

In the simulation phase, we first patch GROMACS, an open-source software suite for high-performance molecular dynamics simulation and analysis with PLUMED, a plugin that works with a large number of molecular dynamics codes and can be used to analyze features of the dynamics while performing a variety of free energy methods, and then we pass the output from VMD as input to start the simulation. Once the execution of the simulation ends, we load the `ffs.tcl` plugin into VMD and provide the output of the simulation as input we can move to the post-processing phase analyzing the results.

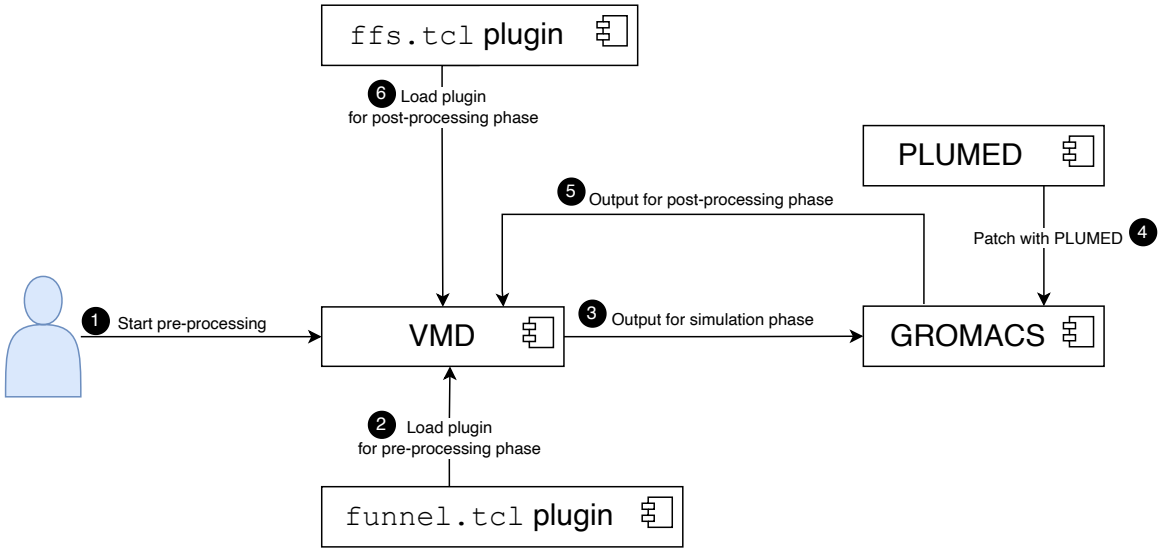


FIGURE 2.6: Scheme of the software architecture of the FM protocol.

2.5 Problems and Improvements

Software is complex in the life sciences field, and effectively identifying and addressing problems is crucial for successful improvement. This chapter addresses the current issues with the actual FM tooling support and how we plan to solve them.

We begin by exploring the importance of data persistence, which is not just about storing data, but also about how to use it to facilitate the retrieval of past changes and future works including other systemic issues that could affect performance or user experience. Understanding the nature of these problems whether they are functional, performance related, security issues, or user experience flaws—is the first step toward designing effective solutions.

Following this, we will discuss various solutions that are required for the aforementioned problems. These include, but are not limited to, data persistence, portability and interoperability, workflow guidance and manual tasks.

2.5.1 Data Persistence

In the FM system, the configuration parameters do not persist after the execution of the tool. If the user encounters an erroneous run of the simulation phase, everything from the pre-processing is lost, only the PLUMED input file is still there. In that case, if the user wants to perform a rerun of the simulation with a lightly-tuned set of the original configuration parameters, a full restart of the pre-processing phase has to be done by reopening the tool and starting again from the beginning configuring every parameter.

Furthermore, the tool does not keep any history of previous research of the ligand proteins. Prior successful runs of the tool might be useful to the user for later similar simulations. An archive can prove useful because the user can compare previous configurations between each other as well as use them as inspiration for new ones. This history feature is also very useful since the user has the possibility to revert an action or reset to a complete configuration state in the past.

Finally, the wrong input in the FMAP-GUI can lead to a full rerun of the tool because some input parameters can't be modified, even though they should. As a result, this makes the tool less user-friendly and more tedious to use.

2.5.2 Portability and Interoperability

The setup process for performing a complete FM process involves several components and can be quite dispersive and complex due to its dependence on multiple software tools and operating systems. Specifically, users must install three different software: VMD, PLUMED, Gromacs (or an alternative Molecular Dynamics (MD) engine). Each of these software has specific system requirements and installation steps, which add layer complexity. For VMD, not only is a separate download and installation necessary, but the user must also manually install additional packages specific to their task or environment. This manual installation process can prove problematic, especially for users who may not have extensive technical knowledge or experience with such tools. The introduction of a web approach provided with a dedicated graphical user interface significantly improves this setup and makes the process more user-friendly. By utilizing such a web application, users can bypass the need to install VMD and its associated packages altogether. The new system, featuring the FMAP-GUI a PLUMED plugin essential for this tool allows users to engage directly with the pre-processing phase without concern for the underlying operating system. Since the entire setup operates within a web browser, it eliminates common installation hurdles related to software compatibility and system configurations. The web platform not only simplifies the initial setup but also enhances accessibility, enabling users to focus more on their research and less on the technical difficulties of software setup. Ultimately, this leads to a more efficient and user-friendly environment for conducting FM processes.

2.5.3 Workflow Guidance

The user interface of the plugin is currently challenging and counter-intuitive, firstly due to a lack of sufficient documentation and a design that does not stick to state-of-the-art UI and UX principles. As a result, users need to own dual expertise: a deep understanding of the (FM) and specific training on how to operate the tool effectively. This learning curve significantly limits the tool's accessibility and usability. Improving the user experience could be straightforward and impactful. One effective method would be to incorporate comprehensive instructions directly within the tool, guiding the user through its various tasks step-by-step. Another significant issue with the current tool is its lack of immediate feedback on errors. Currently, errors in input data are not revealed until the simulation phase, at which point correcting them before can be time-consuming and costly in terms of both time and computational resources. To address this, we plan to implement a robust system for input validation. By introducing constraints on the data entered by users and strictly verifying the correctness of the output files, the tool will be able to identify and notify users of errors earlier in the process. This will help prevent the propagation of errors into later stages, saving valuable time. By focusing on these enhancements, improving documentation and user guidance, and implementing early error detection mechanisms, we can enhance the functionality and user-friendliness of the tool, making it more efficient and accessible for a broader range of users in the FM community.

2.5.4 Manual Tasks

As detailed in Section 2.3.1, the current configuration process requires users to manually set certain parameters that cannot be directly modified via the graphical user interface. For instance, the setting of collective variables (CVs) is a crucial parameter in molecular dynamics simulations. To configure CVs, users must first identify the relevant atoms within VMD software and then manually input their atom serial numbers into a configuration file. This procedure typically involves opening multiple windows in VMD as well as the FM tool to verify atom serial numbers. For instance, if a user needs to set a distance based CV they must locate the specific atoms in VMD, determine their respective serial numbers, and then accurately enter these details into the configuration file. This process is not only tedious and time consuming but also prone to errors, particularly due to the manual entry of data. Such tasks are repetitive and can lead to significant inefficiencies, impacting the overall user experience negatively. To address these issues, automating the configuration of CVs could provide a substantial improvement. By integrating a feature that allows users to select atoms directly within the GUI, and automatically populating the configuration file with the correct parameters, the tool could significantly facilitate this process. Automation would not only enhance the user experience by making the setup quicker and less cumbersome but would also reduce human error, therefore increasing the reliability of the simulation outputs and speeding up the entire protocol.

2.6 Conclusion

In drug development, studying the interaction between a ligand and a protein is fundamental. We have seen how complex it is and how we can gain insights on ligand-protein interaction with computational techniques. Among these, FM protocol is employed to which every phase has its own software component as detailed in Figure 2.6. In the life science field, software has been evolving rapidly and is increasingly becoming an essential pillar along with the application of software and data engineering practices. In this regard, the tooling support of the pre-processing phase of the FM protocol presents some issues from a software and data perspective as explored in 2.5. In this thesis, we focus on addressing these issues by designing and implementing a new tool applying software engineering best practices. Moreover, we will develop a web platform that allows researchers to prepare FM simulations in a guided user-friendly environment connected to a database to store the configurations. The latter is strongly related to the *FAIR* principles in which FM configurations are accessible through a web platform, researchable within the tool, operable in all operating systems and potentially reusable.

Chapter 3

Approach

In order to approach a web-based solution and provide a software tool in such an environment to perform the pre-processing phase of the FM protocol, we need a way of rendering small and large protein-ligand structures. Furthermore, the FM also requires rendering and modifying a funnel shape—i.e., a cylinder attached to a cone section—, representing the molecular elements, in the scene. Thus, we need to choose the molecular visualization library properly so that, given a biological structure, it returns a rendered view and allows its exploration and interaction.

3.1 React environment

We will implement the software within the React[1] environment, a library to develop web and native interfaces by building components written in Javascript (JS). Although it supports TypeScript (TS), which is a strongly typed programming language built on top of JS, it is more suitable for handling complex interfaces in terms of type safety and maintainability of software. Therefore, the chosen library must integrate well with the React environment.

3.2 Molecular Visualization Libraries

For visualization purposes, several libraries have already been developed and are currently employed. We will see here a general overview, with short descriptions of the libraries, where they are used, and their hypothetical application for our project.

3.2.1 Protein Viewer

Protein Viewer[2] (PV) is a lightweight JavaScript WebGL-based software tool to visualize protein structures in the three dimensional space on web browsers. It is a computationally efficient and portable application which can be easily integrated into a website and does not require any additional plug-ins to be installed. It is the default protein viewer in SWISS-MODEL[3].

While being a good option, it has an important limitation for what we need to do. Since we are required to build custom geometry shapes such as cones, cylinders and spheres. PV does not have a direct API to render cones but only Tube and Sphere and we would then need to find a workaround to integrate custom geometries within the WebGL environment. Furthermore, it is not straightforward the integration of the library into the React environment.

[1] https://react.dev/
[2] https://biasmv.github.io/pv/
[3] https://swissmodel.expasy.org/

3.2.2 Jmol

Jmol[4] is an open-source viewer for three dimensional biological structures written in Java. From it, the authors developed JSmol which is an implementation of Jmol that does not require Java and runs in standard web technology. It does not have any direct feature to render custom 3D shapes on the viewer. It does support compatibility with JQuery but we need to integrate it with React environment. Nevertheless, it has a great documentation and a dedicated community forum to resolve general problems.

3.2.3 UnityMol

UnityMol[5] is a molecular viewer written in C# within the Unity3D game engine. It has very poor documentation but it has great potential in building molecular representations and custom geometries. Despite that, it has few problems, the first is that the development cycle of Unity3D is longer and harder than building a web application within a React environment. Additionally, it is still in early release, with the latest version being **0.9.6**, and the last support news dates back to 2016.

3.2.4 NGLViewer

NGLViewer is a library that properly renders 3D molecular structures in the web environment. It is implemented in Javascript and supports Typescript. It is a collection of tools specialized for web-based molecular graphics using WebGL technology to display various molecules such as DNA/RNA, proteins with many function utilities and representations. In Figure 3.1 is shown an example of a 3D molecular structure, a turkey β1 adrenergic receptor shown in ribbon representation and stick+ball representation for the other small molecules (e.g., the ligand dobutamine, cholesterol, lipids, etc.).

FIGURE 3.1: NGLViewer example showing the structure with PDB ID 2Y00

[4] https://jmol.sourceforge.net/
[5] https://www.baaden.ibpc.fr/umol/index.html

NGL selection language

The library provides a selection language that can be used in many interfaces to limit which atoms or residues are shown in the molecular representations or trajectories. It has keywords to make it easier the selection of biological groups of atoms such as **backbone** and **sidechain**. It is possible to make a selection with molecular information such as atom name, residue name, chain name, atom index, residue number and element name.

In addition, the selection language has the expression feature, which can combine keywords and molecular information to allow more complex selections to express ranges with comma separated values or with logical operators. For instance, the string to select a specific atom, like the Cα atom, of the residue number 10 from the chain F would be as follows:

```
10 and :F and .CA
```

This selection language feature is limited to selecting molecular entities with identifiers such as residue names or serials, but one requirement we have to fulfill is to be able to select a number of atoms within a specified distance range from a specific atom.

3.2.5 Considerations

From the aforementioned libraries to integrate molecular viewers in a web technology environment, we extracted the key requirements for our needs described in Table 3.1. Each solution has **full** molecular features that include various representations, molecular selection language and measurements. Nevertheless, not all libraries are compatible with the React environment and this is a problem since we want to develop the tool within it, therefore we can already discard UnityMol. For Jmol, the workaround to allow the integration with React is not a best practice to do. The final choice is between PV and NGLViewer, the first one having limited capabilities for rendering custom geometries, and, it would be harder to do so by applying WebGL shapes from the ground up, while the second having an API to render many different custom geometries.

Library	Molecular Features	React environment	Custom Geometries
Protein Viewer (PV)	Full	Compatible	Limited
Jmol	Full	Limited	Not possible
UnityMol	Full	Not possible	Full
NGLViewer	Full	Compatible	Full

TABLE 3.1: Key feature of molecular viewer libraries.

Accordingly, we will go for the NGLViewer since it fits most of our needs. It is compatible with the React environment, has everything we need for molecular features, and allows the rendering of custom shapes.

3.3 Technologies

The platform is a web application provided with its own toolchain stack and Docker configuration to run the entire system, this will ease the integration in existing systems. For instance, GROMACS simulation software can also be used with docker and it would be possible to integrate and pipeline them together. The system is divided into four different components: Backend, Frontend, Traefik and the Database.

3.3.1 Backend and Frontend

We plan on developing a backend side with Go (Golang) programming language using the Go-Gin[6], a lightweight HTTP framework written in Go that allows to build web applications and microservices providing a set of functionalities to reduce boilerplate code. On the other hand, the frontend will be developed within the React environment incorporating the NGLViewer for the molecular rendering and Material UI (MUI) for the composition of the entire user interface.

3.3.2 Traefik

Traefik is an open-source Edge Router that helps publish services receiving requests on behalf of your system and redirects the request to the responsible components for handling them. It is very useful since it automatically discovers the right configuration for your services by inspecting the infrastructure, where it finds relevant information and discovers which service serves which request. Traefik is natively compliant with every major cluster technology, such as Kubernetes, Docker and many others.

3.3.3 Database

The database employed to store the configurations is called ScyllaDB, a NoSQL database to support key-value and wide column use cases written in C++ where the data is distributed across clusters with outstanding speed, high availability and scalability.

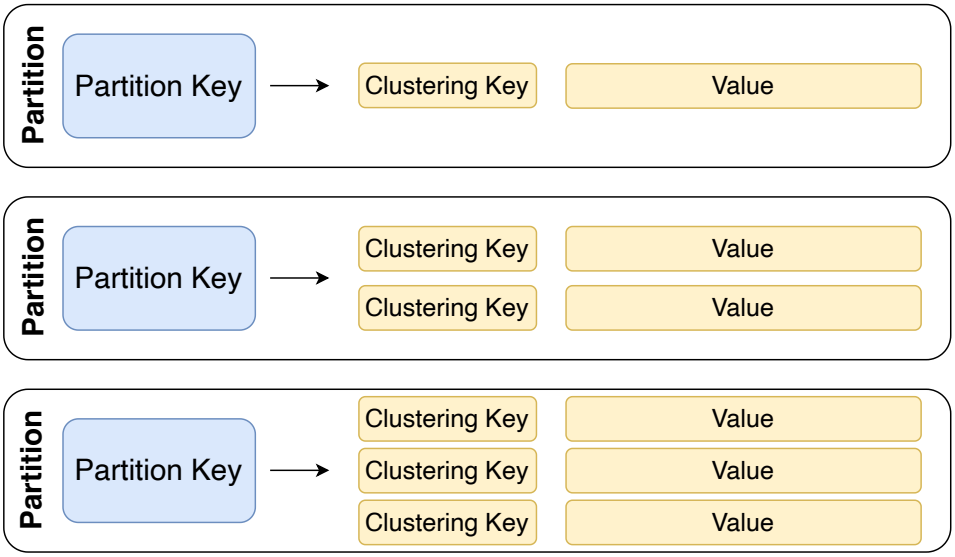


FIGURE 3.2: ScyllaDB partition and clustering key.

Figure 3.2 shows how ScyllaDB manages partition and clustering keys, which is the same way how Apache Cassandra works. Clustering keys have an important role in the data model used to structure data within a partition. The primary key in ScyllaDB is composed of two specific keys: partition and clustering keys. The partition key has its own hash value that represents the node that will store the data and can distribute the data through the nodes in the cluster. In that regard, every row in the same partition key will be stored in the same node. The clustering key has the role of managing the data within a partition, thus sorting the rows and allowing the definition of the order based on the clustering key values.

[6] https://gin-gonic.com/docs/

This can prove very useful when we need to store the user's history of configurations for each research case. By inserting each time new configuration value with a different time value as clustering, the result will be a history already sorted by time. Considering other available database solutions, ScyllaDB is the easiest and simplest to achieve such a goal. Further technical details about this will be explained in Chapter 4.

3.3.4 Evaluation and Feedback

We plan on doing a user evaluation to assess the tool's validity and the improvements we require from the pre-processing phase. Thus, as described in Figure 3.3, after the development stage we start testing the tool with close expert users who know deeply the FM protocol and then once we have good results, we will perform a user evaluation with users in this field asking them to answer a questionnaire and do specific tasks. The process of user evaluation and feedback collection is further explained in Chapter 5.

FIGURE 3.3: Evaluation pipeline.

Chapter 4

Design and Implementation

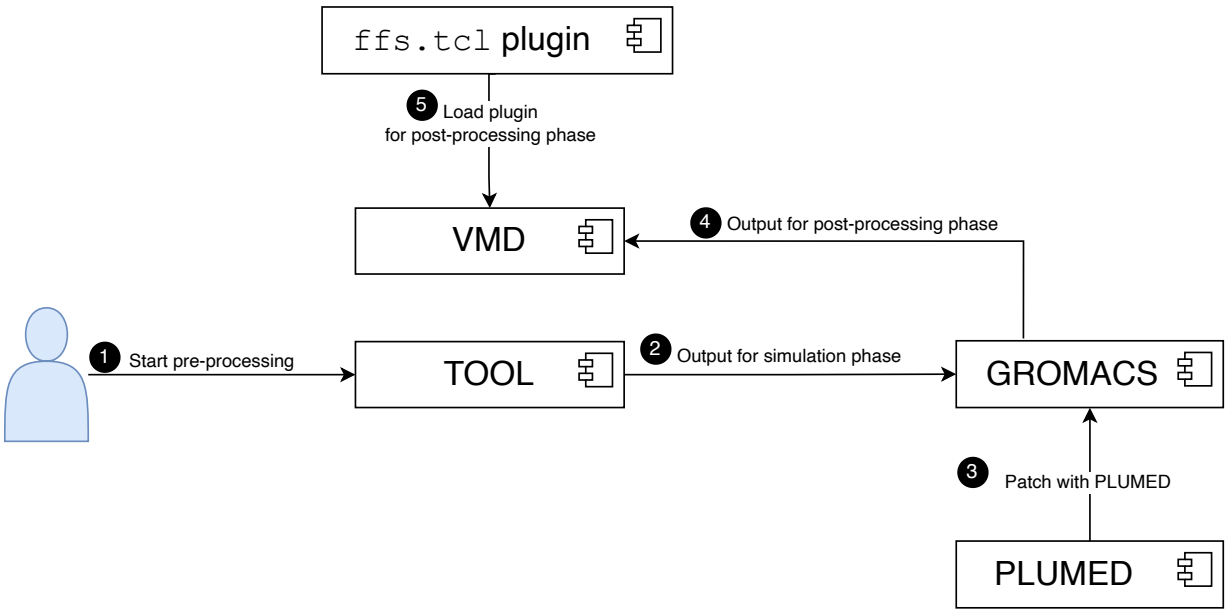


FIGURE 4.1: New scheme of the software architecture of the FM protocol

In this chapter, we provide a detailed overview of the revised procedure, adding the architecture and the implementation of the tooling support to configure a Funnel Metadynamics simulation. Figure 4.1 shows the new scheme of the FM protocol software architecture with the developed tool's introduction. The designed tool substitutes the VMD plugin *funnel.tcl* without any installation requirement and improves the configuration of the input parameters. Being a web platform, it also avoids any incomprehension related to the different OS mapping. It should be noted that, VMD is still necessary for the post-processing phase since the current solution does not analyze simulation results. Moreover, we depict the data model used to store and manage the configurations and we explain the use cases in detail with a technical implementation description of the following:

- **Use Case 1**: Loading molecular structure
- **Use Case 2**: Definition of the funnel and bounds
- **Use Case 3**: Definition of the CV
- **Use Case 4**: Definition of the simulation parameters
- **Use Case 5**: History state and action management

4.1 Context View

The context view diagram in Figure 4.2 illustrates the architecture for running FM simulations. At the center, we have the software system, a web application that is used during the preparation of the simulation. The tool offers an interface where users, referred to as "User" in the diagram, study the ligand-protein interaction and need to configure simulations. The user needs to access the tool environment with credentials to start loading the 3D molecular structure. It directly interacts with the Protein Data Bank, an external web service that provides essential 3D molecular structures. This interaction is done via an API, enabling the software to retrieve the necessary molecular data to display properly through the PDB ID and let the user build the configuration. If the 3D is not found within the PDB service or no 3D structure is available for the system of interest, the user can still load a local structure in PDB format (e.g. in the case of modelled proteins or coarse-grained structures). After the user configures all the parameters for the simulation, the system prepares and provides a PLUMED configuration file essential for running the simulation and sends it to the user. Finally, the user can move the file to the dedicated hardware and launch the simulation.

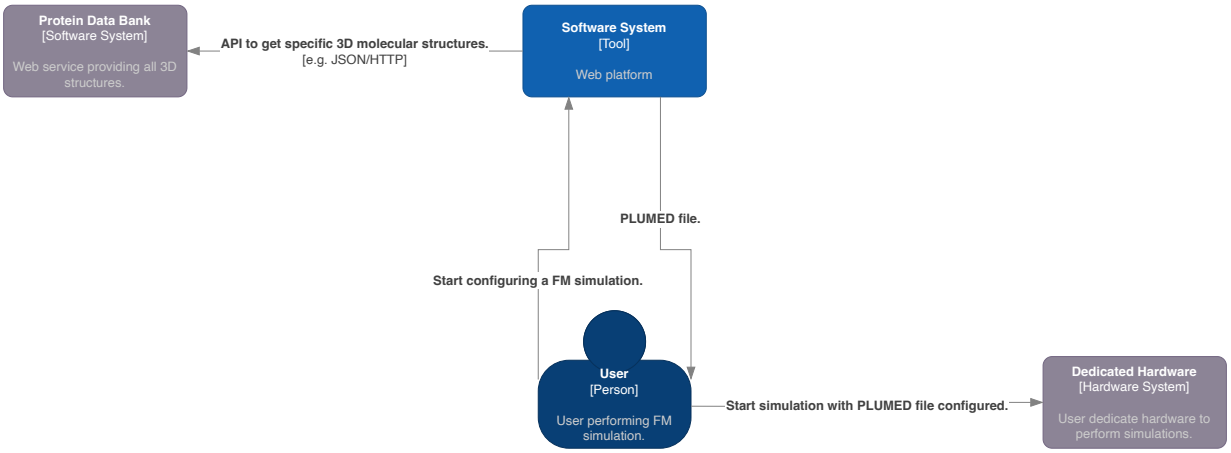


FIGURE 4.2: Context View

4.2 Container View

The container view shown in Figure 4.3 depicts the interaction of containers with the end users. First of all, the Traefik container acts as a reverse proxy, handling connections and directing the traffic to either Frontend or Backend containers based on the request. In addition, Traefik offers a dedicated view to monitor the status of the whole network. The Backend container runs a backend service in Golang using the Gin framework and provides an API that supports web GUI functionalities. It handles basic authentication and authorization as well as query operations accessing the database. The Frontend container is responsible for the graphical user interface and transaction of the simulation data. It allows the retrieval of molecular structure data from the external Protein Data Bank to be used configured and used in simulations. The Database container runs the ScyllaDB database to store user and system data, including molecular configurations supporting the backend's query operations.

The dashed line region (1) contains two containers used during the development and deployment phases. The first is a webhook server [1] written in Golang that gets notified when a new image of the tool is pushed to the Docker Hub, and in that case, it restarts the services with the updated image. Through the GitHub actions, a new tag version triggers the build and push of the backend and frontend docker images.

[1] https://github.com/adnanh/webhook

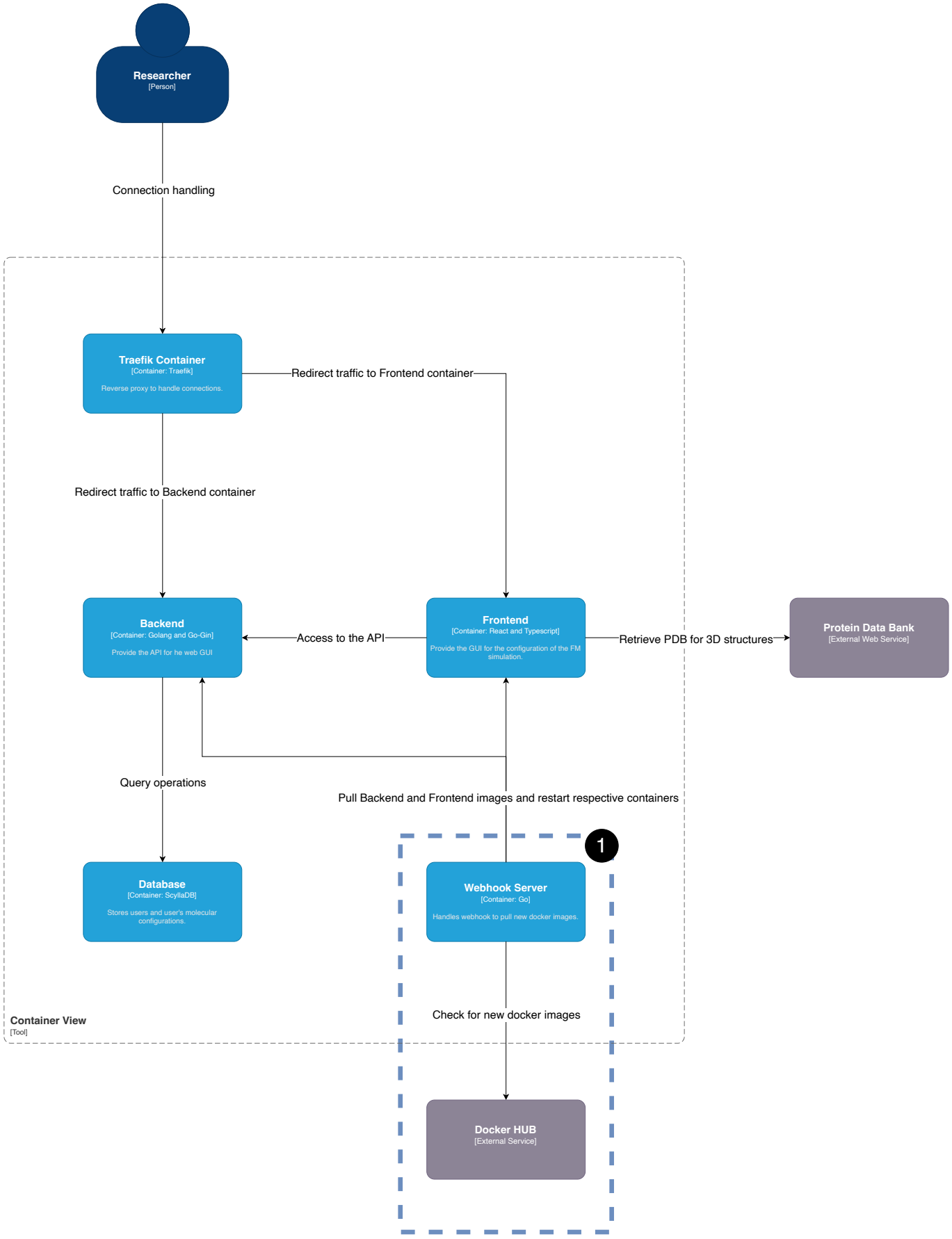


FIGURE 4.3: Container View.

4.3 Deployment

The deployment of the system is divided as seen in section 4.2. The Frontend and Backend containers have their own Docker file and are described in Listing C.2 and Listing C.1. To easily run the system with all its containers, a *docker-compose.yml* is depicted in Listing C.3. We have set up a Continuous Integration (CI) and Continuous Development (CD) pipeline using GitHub Actions, as described in Listing C.4. This pipeline triggers the action when the repository version is tagged. It builds the backend and frontend docker images and pushes them to the DockerHub. A webhook is configured on the web server, as depicted in Listing 4.1, with two entry points. The first one allows the redeployment of the Docker images, while the second serves to check the status of the webhook itself. When the Docker images are pushed, the webhook triggers and initiates the *redeploy-docker* command. This command executes the bash script presented in Listing 4.2 to pull the new Docker images and redeploy them. This feature was extremely helpful during development as it eliminated the need for manual deployment every time, we just needed to tag a new version.

LISTING 4.1: Server config hooks.json file for webhooks.

```
[
  {
    "id": "redeploy-docker",
    "execute-command": "/home/andrea/hooks.sh",
    "command-working-directory": "/home/andrea/FunnelMD",
    "response-message": "Ok"
  },
  {
    "id": "status",
    "response-message": "Ok␣hooks"
  }
]
```

LISTING 4.2: Server bash script hooks.sh.

```
#!/bin/bash

cd /home/andrea/FunnelMD
docker-compose pull
docker-compose up --build -d
docker system prune -f
```

4.4 Data Model

In this section, we present the data model designed to store the configuration of the representation of the funnel on the protein and ligand with the setup of the CVs, but also the modeling for the user's account environment. The first table, depicted in Table 4.1, represents the discovery structure with the minimal data to render the whole configuration without accounting for many rendering computations to the client side. The PRIMARY KEY is composed of the `account_email` and the `id` with a CLUSTERING KEY as `lastupdatetimestamp`. The second is shown in Table 4.2, representing a simple table for managing the account of a user.

Field	Type
account_email	text
id	uuid
lastupdatetimestamp	timestamp
alpha	double
anchor_point	int
angle_atoms	list<frozen<list<int»>
b_point	list<double>
com_atoms	list<frozen<list<int»>
creationtimestamp	timestamp
d_point	list<double>
dihedral_atoms	list<frozen<list<int»>
discovery_name	text
distance_atoms	list<frozen<list<int»>
fps	list<double>
protein_name	text
r_cyl	double
radius	int
status	int
t_point	list<double>
walls	list<double>
zcc	double

TABLE 4.1: Fields of Table discoveries in Database.

Field	Type
email	text
creationtimestamp	timestamp
discoveries	map<uuid, int>
password	text
status	int

TABLE 4.2: Fields of Table accounts in Database.

4.5 User Interface structure

The tool's core UI component is the preparation view, which allows the user to prepare the molecular structure. Figure 4.4 shows how the page is designed with five different elements.

1. On top of the page, we have the top bar dedicated to navigating the tool
2. The biggest element contains the molecular view such that the user has the necessary space to explore and move around the molecular structure.
3. The left sidebar which has six tabs to switch between different functionalities.
4. The bottom sidebar is the component for the command line interface where the user can input defined commands to work on the configuration.

5. In the right sidebar all inputs required to prepare the FM are located, such as all funnel parameters, and by switching the tab from component **3)**, this view changes inputs.

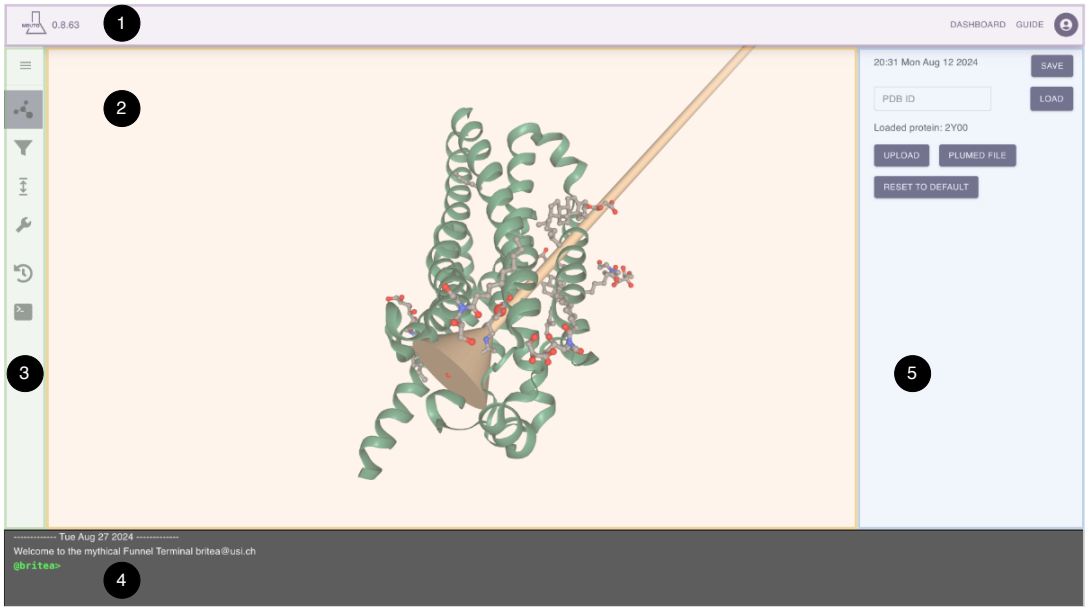


FIGURE 4.4: General UI sections.

We designed a general user flow that serves as the basic user usage. The diagram in Figure 4.5 shows the process from entering the home page to downloading the final PLUMED file. During this process, the user needs to authenticate with their credentials, create a new discovery on the dashboard page, and load the molecular structure. After this, a series of activities should be completed in sequence:

1. preparing the funnel shape
2. defining the bounds
3. adding the collective variables
4. setting the simulation parameters

The flow is designed such that the user can easily and clearly define all the necessary parameters and values for the FM simulation without forgetting anything. In this way, the UI which originated from this user flow guides the user through all steps.

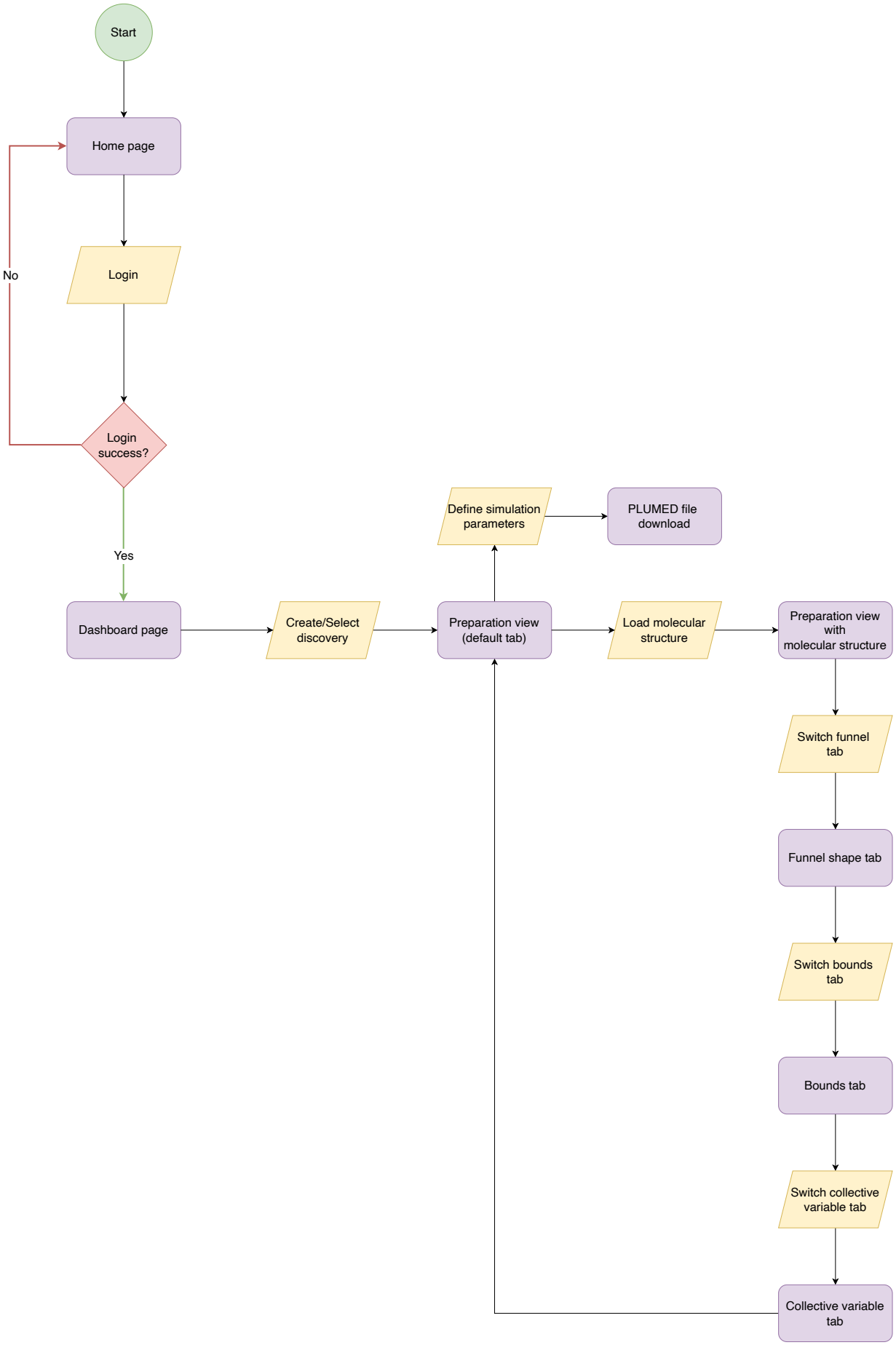


FIGURE 4.5: User flow diagram.

4.6 Loading molecular structure

The first use case is how to load molecular structures into the tool. There are several extensions that store molecular structure information. Many visualization software (VMD being one of the most famous and taken as a reference hereafter) support popular file formats such as topology files PSF and PARM, a format containing atom connectivity and information but without data on coordinates. However, trajectory files such as DCD, CRD, TRR, and XTC files, mainly containing coordinates and energies of a simulation over time. Finally, coordinate files PDB, GRO, G96, and XYZ files contain cartesian structure information. In VMD, only the PDB, GRO and G96 files can be loaded alone, retaining all the information, whereas others could miss some data without pairing them with additional files. With the new tool, we opted to selectively work with PDB and GRO files. This is done on purpose since we restrict the allowed extensions' file structure to the ones we use to perform FM simulations, but potentially we could allow even others similarly to VMD.

The first step for a user preparing an FM simulation is to add a new discovery to his account after logging in and giving it a name, as shown in Figure 4.6.

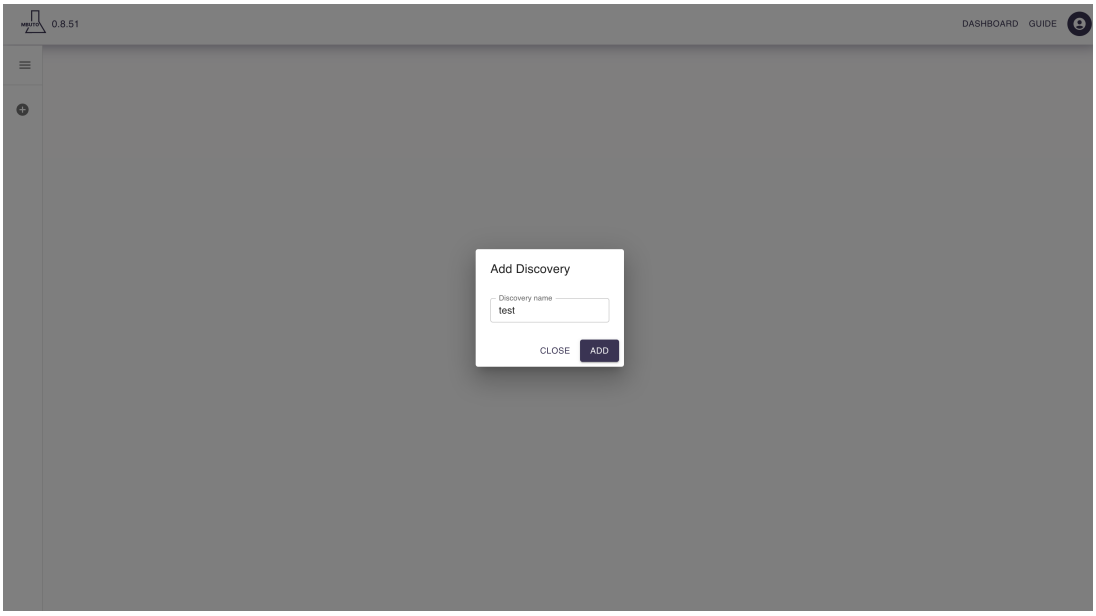


FIGURE 4.6: Add a new discovery

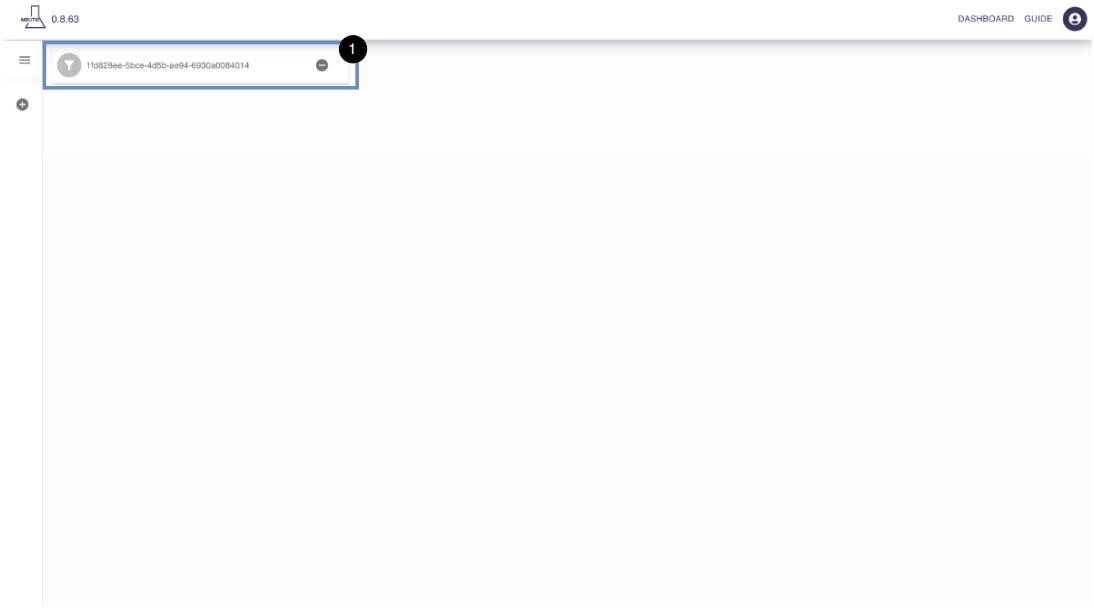


FIGURE 4.7: Dashboard panel.

In the Dashboard panel, a new discovery item will appear, and the user can click it to enter the preparation phase depicted in Figure 4.7. The discovery item can still be deleted by using the minus button on it. At this point, as Figure 4.8 shows, the **2YOO** default protein is loaded and can be changed with the button highlighted in box **1**. That is the quickest way to load a molecular structure because the user must enter the PDB ID from the Protein Data Bank and the application will crawl it from the Protein Data Bank API and load it into view. Another way to load the molecular structure is provided through box **2**, where the user can upload a local file. This is extremely useful since many researchers have modified molecular structures coming from their experiments, and they might want to do simulations with it and not with standard structures published in the PDB database.

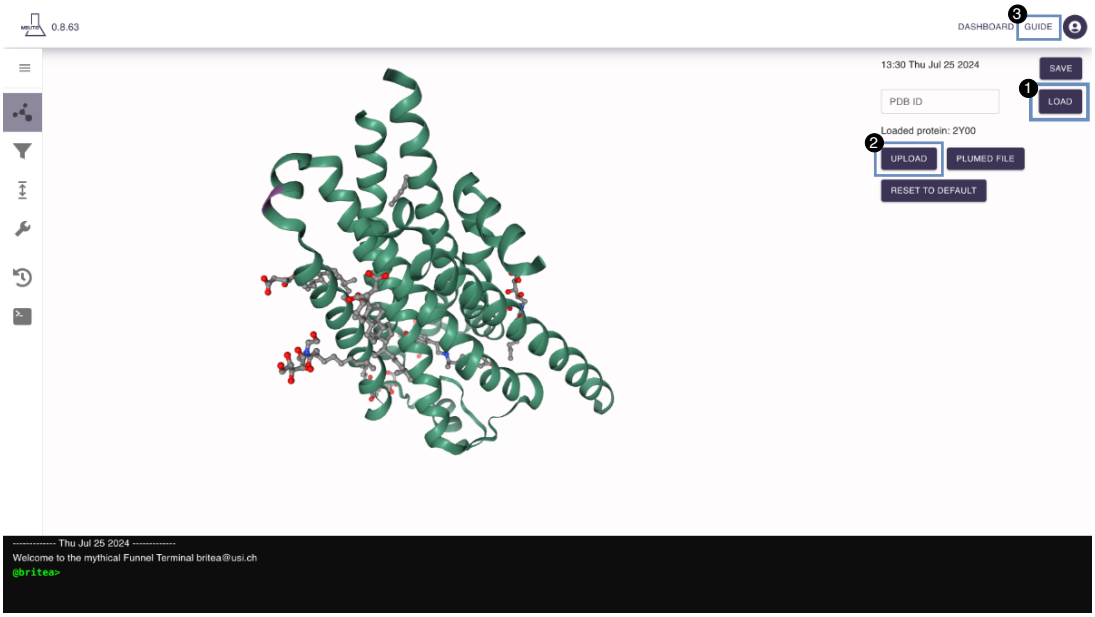


FIGURE 4.8: Preparation view.

In the above view, the user is able to perform several keyboard and mouse events that help the user navigate the molecular structure. In Table 4.3, we depict all possible events and their effect on the view. Furthermore, a user guide can be accessed through the button in box **3**. Once prompted, a full-screen modal window appears containing the guide information. For further details on the main commands explained in the user guide, we refer to Appendix D.

Key	Event
ctrl+drag	translate the view in the direction of the drag.
shift+scroll	clip near the view based on the scroll delta.
shift+left-click	recenter the view on the protein center.
scroll	zoom in/out of the view.
w | a | s | d	respectively move the view down, up, left, and right.
q | e |	rotate the view to the left and to the right.
hover	highlight the amino acid on the protein ribbon representation.

TABLE 4.3: Funnel parameters example.

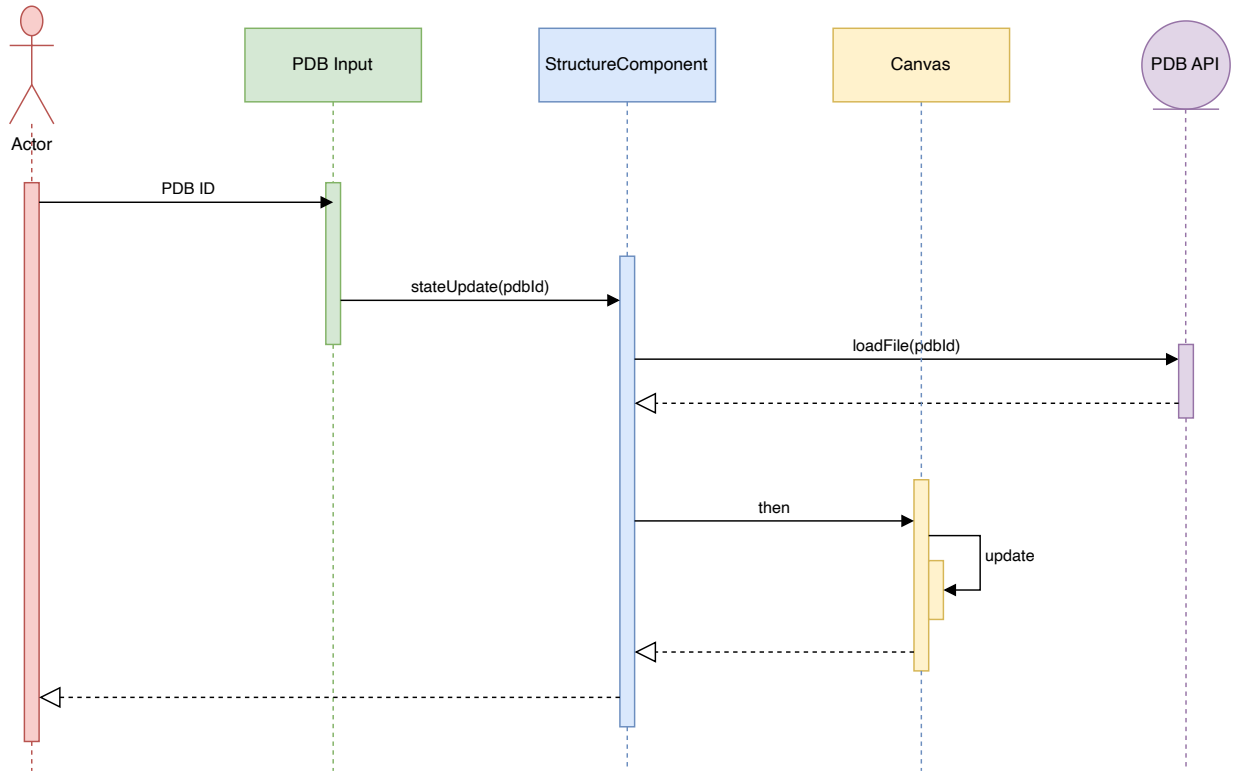


FIGURE 4.9: Loading structure sequence diagram.

In this use case, the `StructureComponent` is the core element, where we use the Stage class from the molecular library, which is central for rendering the molecular scene, and assign it to the element `ID` where we want to attach the rendered canvas on the web page. The class offers a method to load a file to render the molecular structure onto the stage named **loadFile**, which takes two parameters. The first is the *path*, a String in case of a URL or for the file data in File or Blob object, the second is the *params* loading parameters to configure how the structure has to be rendered. In this way, the user can provide the PDB `ID` or upload a file containing a molecular structure that will trigger a `stateUpdate` by requesting the molecular structure

and rendering it on the canvas inside the `StructureComponent`, providing the molecular view on to the user as depicted in Figure 4.9.

4.7 Definition of the funnel and bounds

As a second use case, we describe the main part of the FM configuration in the pre-processing phase, the setup of the funnel shape in the molecular structure with its bounds. All parameters have a default value besides the point A and point B. Figure 4.10 shows the view for the definition of the funnel, and the interested component is the right sidebar where the user can actively define through the inputs. From top to bottom, there are eight different UI components. It is divided into three main parts, the funnel view visualization, funnel axis settings, and funnel shape. First, **1)** is a button to hide or show the funnel shape. It is useful to let the user explore the molecular structure without having the funnel object inside the view. For instance, the user might want to look at a specific location to consider precisely the atoms that need to be inside the funnel. This feature is also available through the CLI at the bottom of the page using the commands:

- Show command: `show funnel`
- Hide command: `hide funnel`

Then, in box **2)** we find the radio buttons to set point **A** and point **B** of the funnel. Option A is selected by default, this allows to pick the 3D position by clicking an atom in the molecular view. The same happens when **B** is selected, but instead, it updates the point B of the funnel. Another way of setting those points is presented in box **3)**, in which the user can directly interact with text inputs to modify the x,y, and z coordinates of the points. These changes, including the position picking, trigger automatic updates on the view such that the user is aware of the modifications.

After tackling the cone axis settings, the tool needs to build the funnel shape based on three parameters shown in box **4)**: `alpha`, `rcyl`, and `zcc`. The alpha parameter is a text input that accepts only numbers and has steps of 0.1 radians. It updates the view with the modified amplitude of the cone region. The second, the `rcyl`, is the radius of the cylinder and it is also a text input accepting only numbers. When the latter is updated, it scales the whole funnel shape accordingly so that the user does not have to compute the values of the shape again. The third input is the `zcc` parameter, which describes the distance between the cone base and the starting point of the cylinder. It is represented with a slider input to ease the changes since it is subject to many tries in order to get the desired funnel form. Finally, the box **5)** contains the input for defining the anchor point, which is the serial number of an atom in the molecular structure. It has a checkbox, and once it has been checked, the user can click an atom in the view and the value updates with its respective serial number. An example of a funnel restraint configured is presented in Figure 4.11.

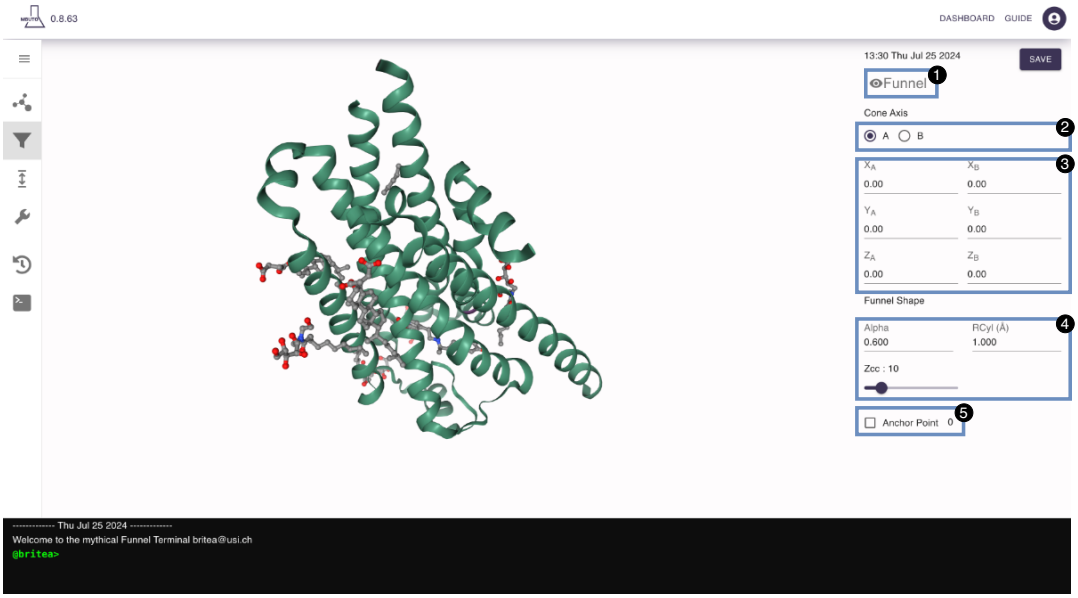


FIGURE 4.10: Funnel shape configuration view.

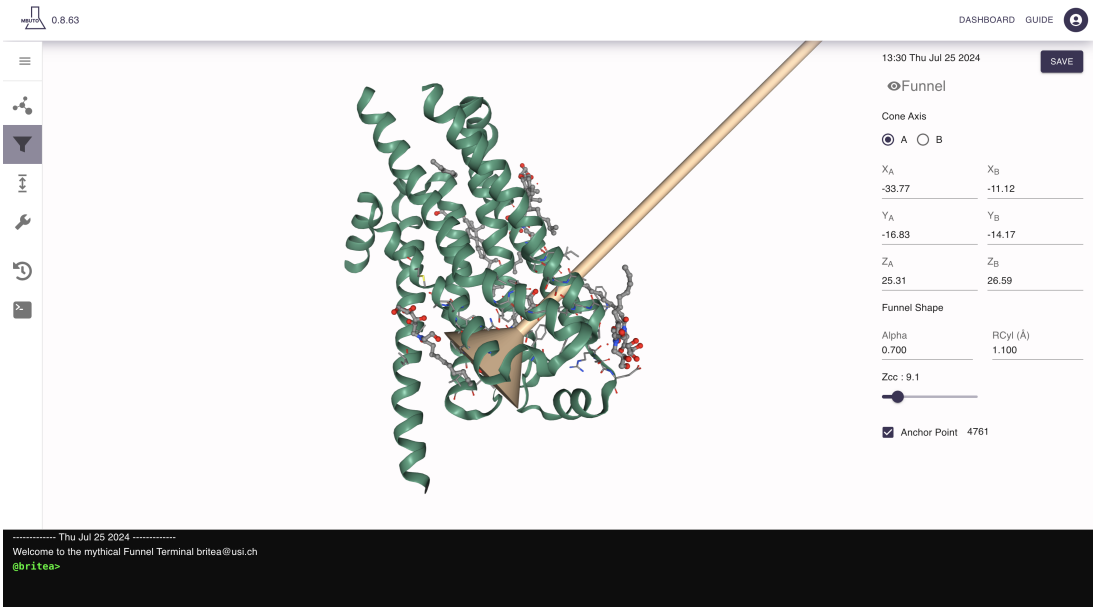


FIGURE 4.11: Funnel shape configured example.

To keep the view updated, the client side needs to compute all values every time to provide real-time geometric visualization of the funnel shape. The funnel configured through the user interface described above is a custom shape that merges a cone with a cylinder. We wanted to create a funnel like structure using a frustum shape attached to a cylinder shape. However, the library we are using does not allow us to specify the top and bottom radius for the `Cylinder` geometry class, making it impossible to achieve a conical frustum. The authors of the library acknowledged this limitation in their GitHub repository issue [2], where they mentioned that the implementation of the cylinder geometry only accepts one radius instead of two. Technically, the end result is correct and does not affect any value. Graphically, we have to address the fact that we can see the small top part of the cone inside the cylinder.

[2] https://github.com/nglviewer/ngl/issues/952

The funnel shape is built with the support of the molecular library by instantiating a `Shape` object. To it, we use the class method `addCone` which takes the base point and the top point as a Float32Array, the color, a `Color` object instantiated with a String containing the RGB values, and the radius value. Secondly, we add the cylinder on top of the cone using the class method `addCylinder` with the required fields such as base and top point similar to the `addCone`, a `Color` object, and the radius of the cylinder. Finally, to visualize the custom geometry, we attach the Shape object to the `Stage` class using the method `addComponentFromObject`, which creates a component from the Shape object and adds it to the Stage.

Merging the parameters used in FM simulation and the requested values to build the funnel shape from the library class, we need to compute the base and top point on every update. The base point is given by the user once it is selected, while the top point requires computation and we can exploit the height of the cone to compute it. As shown in Figure 4.12, the height of the cone is given by the sum of the `zcc` value and the distance between the cylinder starting point and the cone top point. We then compute the height as formulated in 4.1.

$$height = \frac{r_{cyl}}{\tan(\alpha)} + zcc. \tag{4.1}$$

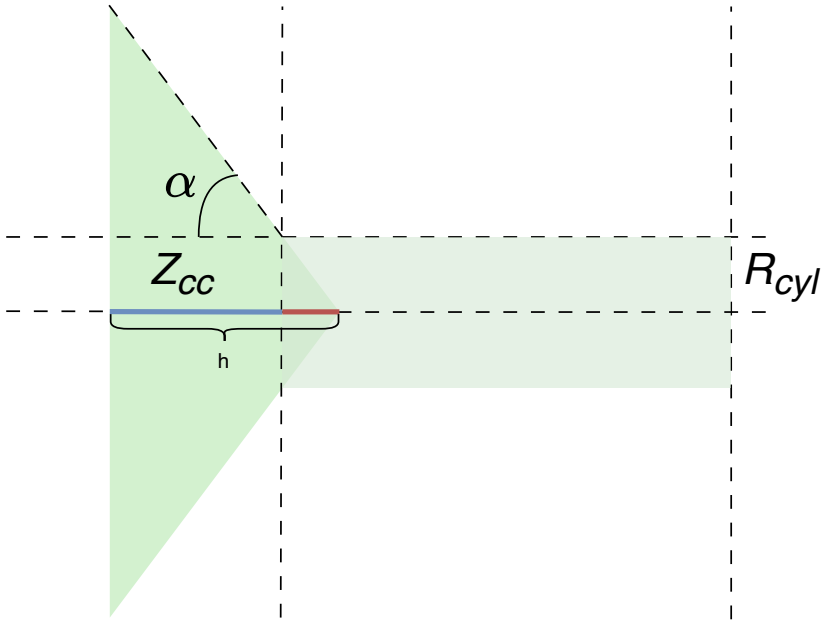


FIGURE 4.12: Funnel height representation.

Afterward, we can compute the top cone point of the funnel shape and, given the point for the base of the cone x_b, y_b, z_b, the direction point that will give the funnel direction x_d, y_d, z_d, the top cone point is calculated with the formula 4.2.

$$(x_t, y_t, z_t) = \left(x_b - \frac{(height \cdot (x_b - x_d))}{distance_{bd}}, y_b - \frac{(height \cdot (y_b - y_d))}{distance_{bd}}, z_b - \frac{(height \cdot (z_b - z_d))}{distance_{bd}} \right). \tag{4.2}$$

With the top cone point coordinates, we are now able to use the class method to build a cone shape. For the cylinder shape, we compute the starting point based on the distance to the base point, which is equal to the `zcc` value. With it and the radius of the cylinder value, we can build the cylinder shape. By combining a cone and cylinder we then form a funnel shape. The radius of the cone base is not the responsibility of the

user since we can derive it from the other cone's values and for each update on the funnel, we compute the radius cone as shown in formula 4.3 based on the `alpha` value and the cone height computed previously.

$$r_{cone} = |h \cdot \tan(\alpha)|. \tag{4.3}$$

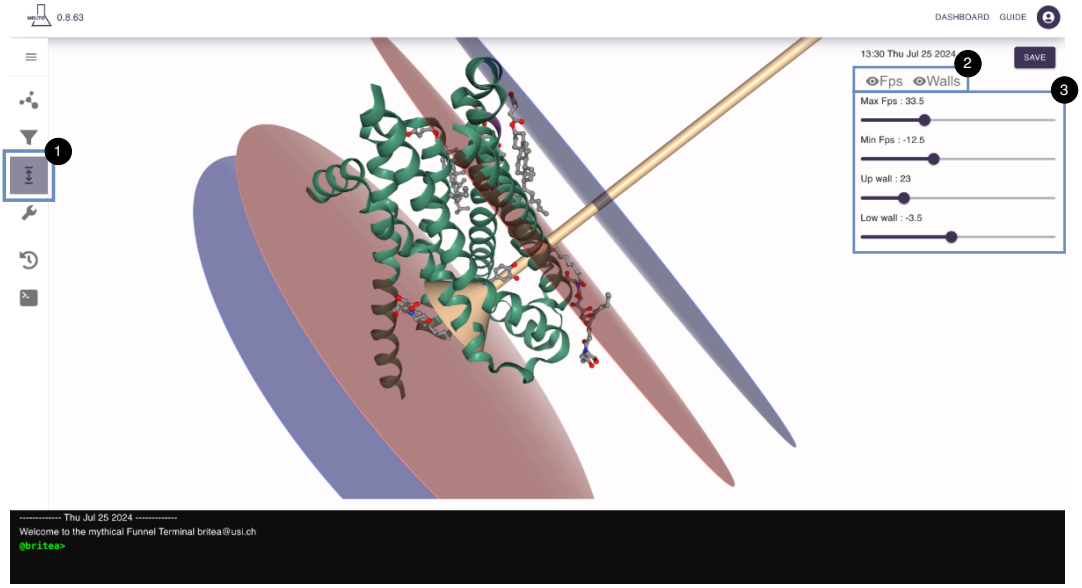


FIGURE 4.13: Bounds definition view.

Once the user sets the funnel, bounds need to be defined to disallow sampling of unnecessary regions inside the funnel and the view can be accessed by switching to the third tab positioned in box **1)** in Figure 4.13. We have four different bounds related to the simulation parameters: `max.fps`, `min.fps`, `low wall`, and `upper wall`. The ones below the base cone point are, respectively, the `low wall` in red and `min.fps` in blue, while the others above the top cone point are the upper wall in red and the `max.fps` in blue. In box **2)**, the user can hide the fps or wall bounds as desired to have a cleaner view and position them better. In box **3)**, the user has four sliders according to the four values to modify the distance of these bounds. The `max.fps` and `upper wall` are values that go from 0 to 100 and can not go below the base cone point, while the other two values can be negative or positive since they can stay below or above the base cone point. There is one special relationship between the `min.fps` and `lower wall`, the latter is always at least 2 Å towards the base cone point. Therefore, the distance between the lower wall and the base point can not be greater than the distance between the `min.fps` and the base cone point. This is done on purpose (according to the technique) and is necessary in order to avoid the FM simulation crashing during runtime.

4.8 Definition of the CV

The third use case is the definition of the collective variables interactively within the main view on the molecular structure. In general, there are several CVs that can be defined[3] that are used in different methods and situations based on the goal of the simulation. For the FM protocol, we implemented three CVs, distance, angle, and torsion, but more can be added through new feature updates. The fourth configuration is the center of mass, which is not correctly a CV but is used as a reference for calculations. This is an important step because throughout the simulation the user wants to measure a specific CV based on the goal of the research and the molecular structure of the study.

[3] https://www.plumed.org/doc-v2.8/user-doc/html/_colvar.html

As we already mentioned, this step is done manually by entering the atoms' indices into the CV definition in the PLUMED file. Figure 4.14 shows the view of the aforementioned step, with which the user can define his own CV accessed by switching the panel tab in box **1)**.

A distance CV is composed of two atoms, an angle of three, a torsion of four, and a center of mass is a virtual atom positioned as a weighted sum of the Cartesian coordinates of two or more atoms. The selection procedure is done by picking the desired atoms where the user is able to click on atoms using `ctrl + left-click` or `right-click` combinations. On selection, the view shows the selected atom by wrapping it in a green transparent sphere, and when the definition is complete, the user has to pick the last atom again to confirm the selection. Moreover, if the user double-clicks while picking an atom, the selection procedure ends, and the tool understands which of the four CVs belongs based on the number of atoms selected. Until the user double-clicks, the definition of the CV will not be completed, and it is possible to correct the selection by removing atoms incorrectly picked by single-clicking again on already selected atoms. In this way, the CV selection should be flawless without any delays in choosing the wrong atom. In box **2)**, the user will see the atom indices selected for the CV.

FIGURE 4.14: CV panel view.

An example of selected CVs are shown in Figure 4.16. On the right sidebar in **box 1** are the defined four CVs by the user. These can still be deleted with the delete button right to the defined CV with the atom indices. Accordingly, the view is always updated on selecting and defining the CV, as **box 2**, **box 3**, **box 4** show we have defined one distance, one angle, and one torsion CV. Each CV has an associated acronym in string format which is composed of the type of CV and the index starting from zero. For instance, the first distance CV defined will have the acronym **D0** with **D** representing the distance CV and **0** the first distance CV element. For angle, is used **ANG** while for torsion **DIH** is used.

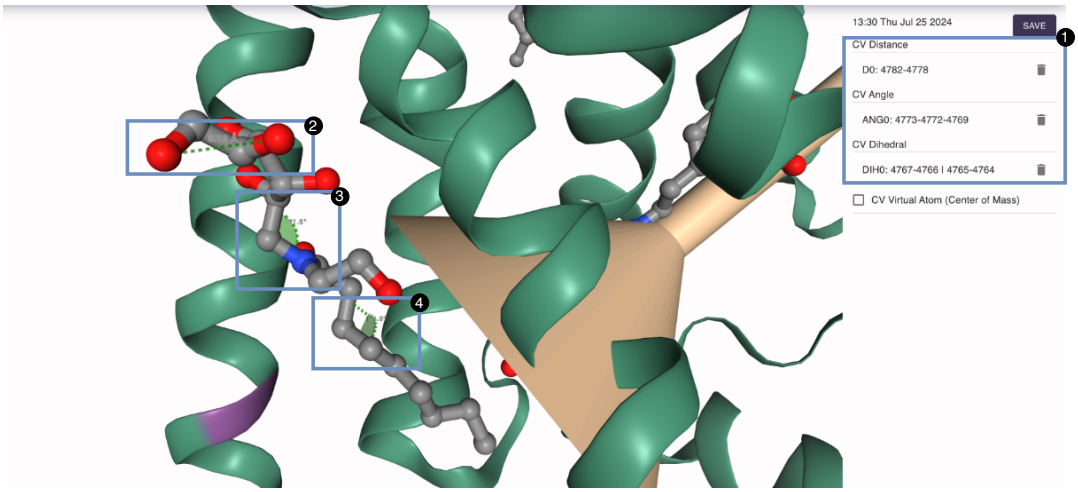


FIGURE 4.15: Distance, angle, dihedral CV selection example.

The definition of a center of mass CV is slightly different from that of previous CVs. Figure 4.16 shows two distinct methods to select a center of mass. The first one is to select atoms one by one in the same way as the other CV definition but to do that, the user has to click on the checkbox highlighted in **box 1** before starting to pick the atoms for the center of mass. On clicking atoms, the right sidebar updates the atom indices, and once the user has the desired set of selections can click on the adding button in the **box 2** or cancel it with the delete button on the left.

The first way of selecting a center of mass can be dull when the user needs to select all atoms of an entire portion of the protein or molecule. For instance, a user who needs to select an entire ligand might select tons of atoms, which is not very suitable for flawless and easy selection. For that purpose, we develop a method to select all atoms once within a selection in CLI. The selection is done with the selection language explained in 3 and the command `com` followed by the selection string presented in box **3)** and as follows:

- Center of mass selection command: `com [selection string]`

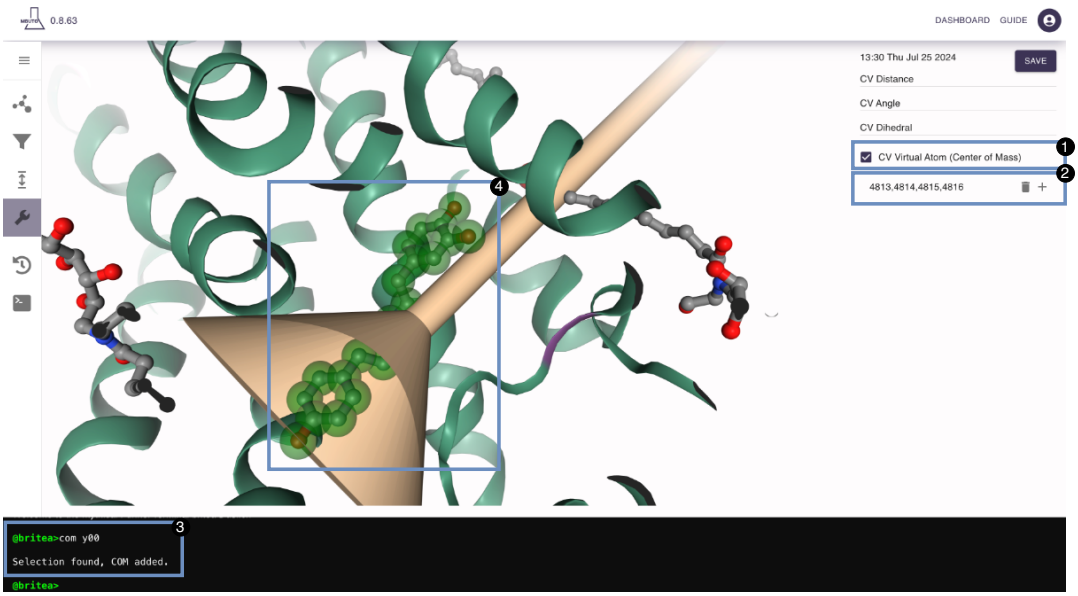


FIGURE 4.16: Center of mass selection view.

In box **4)**, we can see the selection of all atoms belonging to the `y00` compound. After confirming the center of mass selection, the view is updated with a sphere representing the center of mass as shown in box **1)** in Figure 4.17. To separate different centers of mass, each is displayed in a different color and a unique acronym composition composed by "COM" and the index representing the element in the list of atoms. Specifically, when we define a center of mass, we need to add it as an atom of the molecular structure. Thus, they are called virtual atoms. This is useful for adding a COM as an atom of other collective variables, allowing the user to select it to represent a distance, angle, or torsion CV. To recognize them in the view, the user can hover over a center of mass, and a tooltip with the name appears.

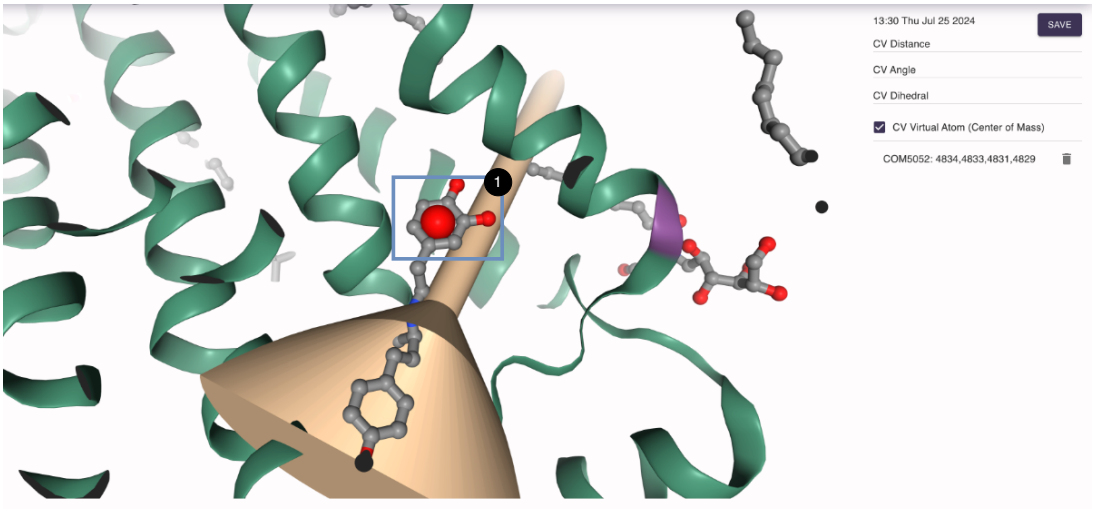


FIGURE 4.17: Center of mass selected example.

The library provides the implementation of the representation of the distance, angle, and torsion, but there is no such feature in the CV center of mass. First of all, the computation of the position of the center of mass is provided by the formula 4.4 which is the summation of the coordinates multiplied by the atom masses and divided by the summation of the atom masses for each type of coordinate.

$$(x_c, y_c, z_c) = \left(\frac{\sum_{n=1}^{N} x_n \cdot m_n}{\sum_{n=1}^{N} m_n}, \frac{\sum_{n=1}^{N} y_n \cdot m_n}{\sum_{n=1}^{N} m_n}, \frac{\sum_{n=1}^{N} z_n \cdot m_n}{\sum_{n=1}^{N} m_n} \right). \tag{4.4}$$

In Figure 4.17, there is one red sphere, but the user can define multiple centers of masses. We defined multiple colors such that each COM has a different color and can be easily distinguished from the others.

4.9 Definition of the simulation parameters

The definition of the simulation parameters is the fourth use case, and it is the step required after the definition of the CVs to complete the configuration and download the final PLUMED file. This feature, besides setting the FM simulation parameters, allows us to lay out standard metadynamics simulations by manually removing the unnecessary lines in the PLUMED file. A further extension beyond the scope of this work may integrate the automatic formatting based on the type of simulation the user is doing. In that case, the interface would be adapted to support all the tunable parameters pertaining to the selected simulation scenario. In Figure 4.18 the simulation parameter configuration view is presented and is accessible with the first tab panel highlighted with box **1)**. The user can then find the button **PLUMED file** in box **2)** to show the

configuration of the parameters in the box. In box **3)** the simulation parameters configuration component is presented, including the reference PDB file used during the simulation when applying the metadynamics bias to the system for the alignment of the target structure. To achieve that, we provide a selection input where the user can select a structure from the system to align and download a file containing it in PDB format.

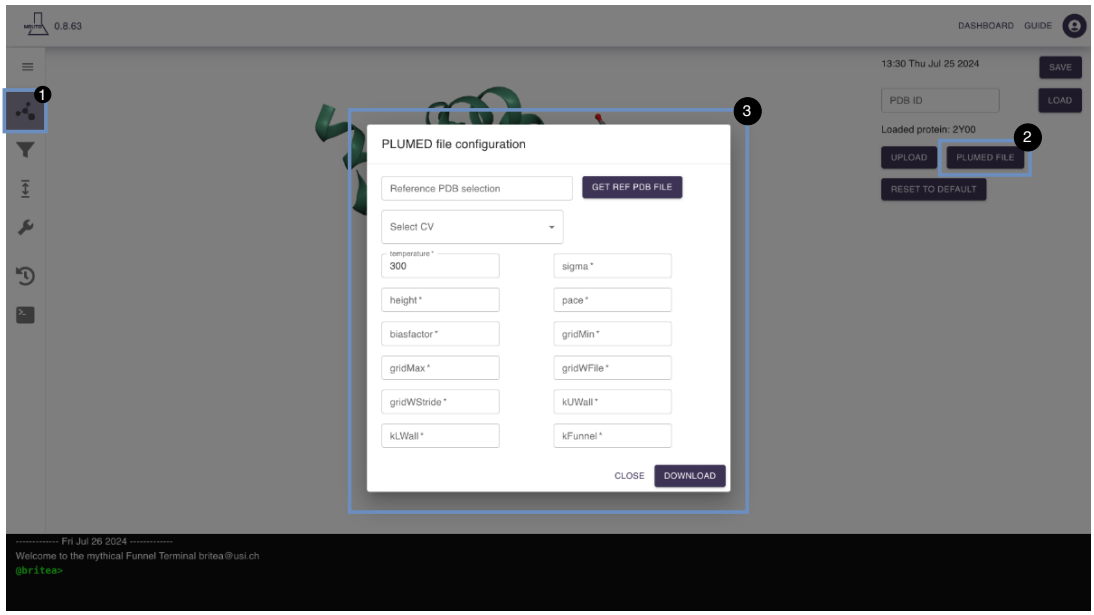


FIGURE 4.18: Simulation parameter configuration view.

The next step is to select the CVs to simulate, in which the user has a list built from all the CVs previously defined. Then, the user can set the other simulation parameters illustrated in Table 2.2. Once the parameters are configured, the download button downloads the PLUMED file. Last but not least, the user can freely close the configuration panel without losing the parameters since they are saved locally, making it possible to modify other values such as the funnel parameters, bounds, and CVs.

4.10 History state and action management

The final use case represents the tool's ability to assist the user throughout the configuration process. Moreover, since there are a large number of parameters to define across various sections, we implement a history management system, allowing users to go back to previous modifications. There are two kinds of historical states: persistent and local. The first one occurs when the user saves the configuration state, which saves the progress of the work. In this manner, we keep track of all changes saved by the user, and these are always loaded in the history component when a discovery is open. With this, the user is able to go back to any previously saved state at a later time. The persistent configuration state is the complete configuration with all parameters. The second one, instead, is a local history, where the user does not have to actively save anything, and the system automatically records every small change the user makes. The local history system tracks not only the entire state, such as the persistent one but also the changes that have been made, thus allowing for specific modifications to be reverted without affecting the entire configuration.

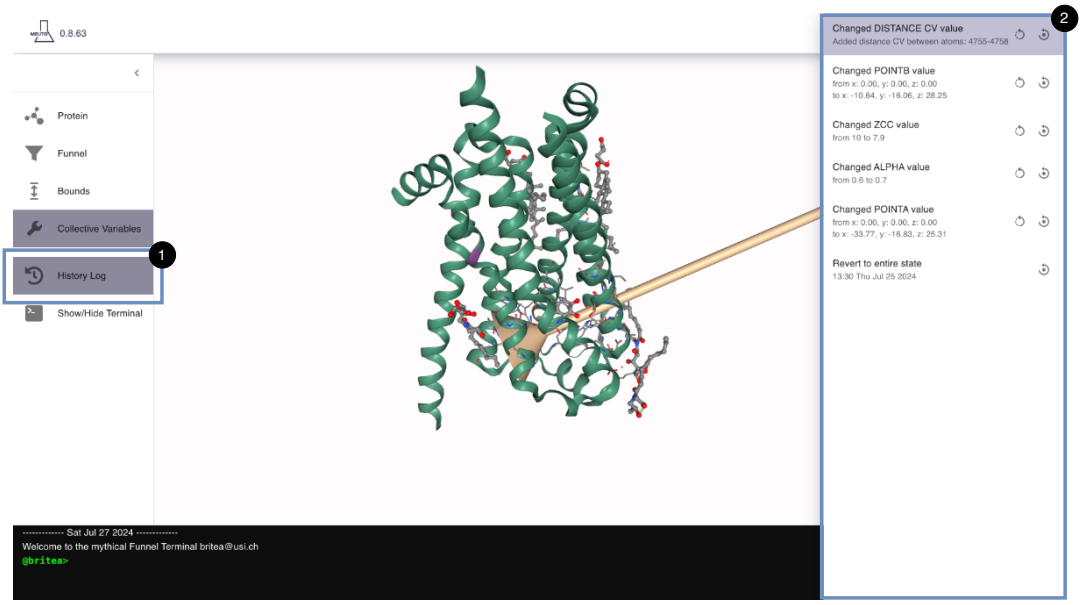


FIGURE 4.19: History management view.

A history state is the state of the entire configuration, including the funnel, the bounds, and the CV settings. Box **1)** in Figure 4.19 shows how to access the history management, and box **2)** contains how the history appears on the right sidebar, with a scrollable list of history states where each line represents a change. Every row has a title describing what is changed, and below it, the user has information on the changed values, providing the original and modified values. On the right are two buttons, the first one to revert the action and the second to reset the entire state. The elements stored in a state are presented in Table 4.4.

Field	Type
alpha	number
angleAtomsPair	pair: number[], removed: boolean, cv: number[][]
distanceAtomsPair	pair: number[], removed: boolean, cv: number[][]
dihedralAtoms	pair: number[], removed: boolean, cv: number[][]
pointA	Point3D
pointB	Point3D
pointC	Point3D
comAtoms	pair: MappedComAtoms, removed?: boolean, cv: MappedComAtoms[]
fps	number[]
walls	number[]
rcyl	number
radius	number
zcc	number
action	actionType
actionValue?	string
lastUpdateTimestamp?	string

TABLE 4.4: Funnel parameters example.

Through the **History Log** button, the user can access the history states and view a list of all changes made with two possible actions in each state. The first action is called **revert**, and can be used to revert

the change with respect to the actual configuration state. For instance, the user has the following funnel settings described in Table 4.5, and, as a consequence of having defined the bounds, the user needs to revert the **PointA** back to the original position.

Field	Value
PointA	-12.00, -22.00, 36.00
PointB	-4.00, -13.00, -3.00
Alpha	0.6
RCyl	1.0
Zcc	10

TABLE 4.5: Initial funnel parameters example.

Field	Value
PointA	-12.37, -22.93, 36.57
PointB	-28.24, -16.88, 12.59
Alpha	0.6
RCyl	1.0
Zcc	10

TABLE 4.6: Changed funnel parameters example 1).

An example is given with Figure 4.20 referring to the values reported in Table 4.5 and Figure 4.21 presenting the values in Table 4.6. By reverting the position of the **PointB** to the initial position, the funnel position changes the coordinates, maintaining all other parameters.

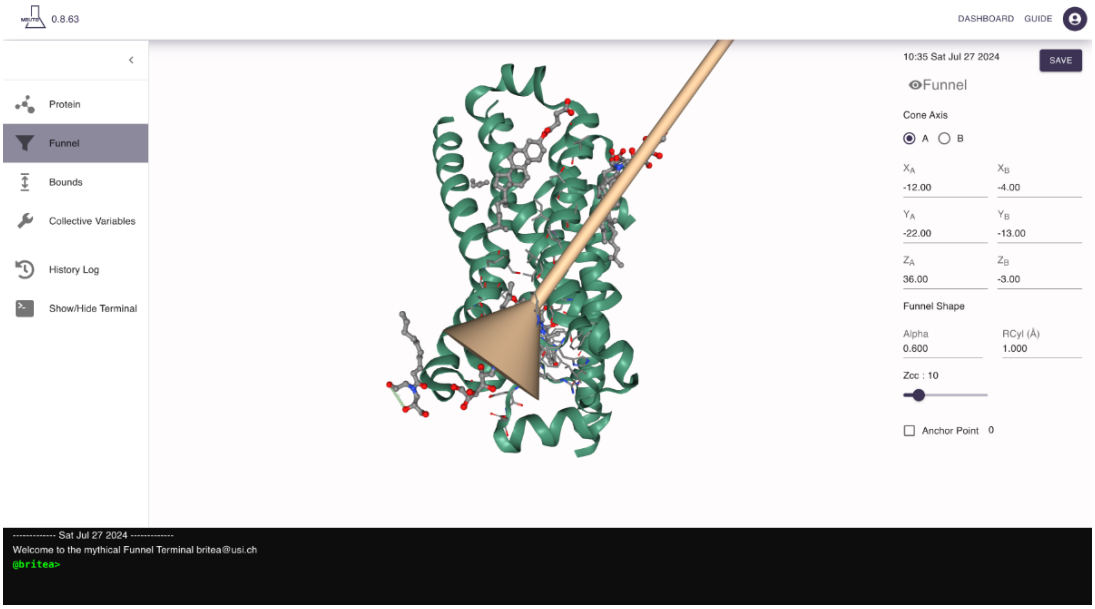


FIGURE 4.20: Example of funnel parameters (refers to Table 4.5).

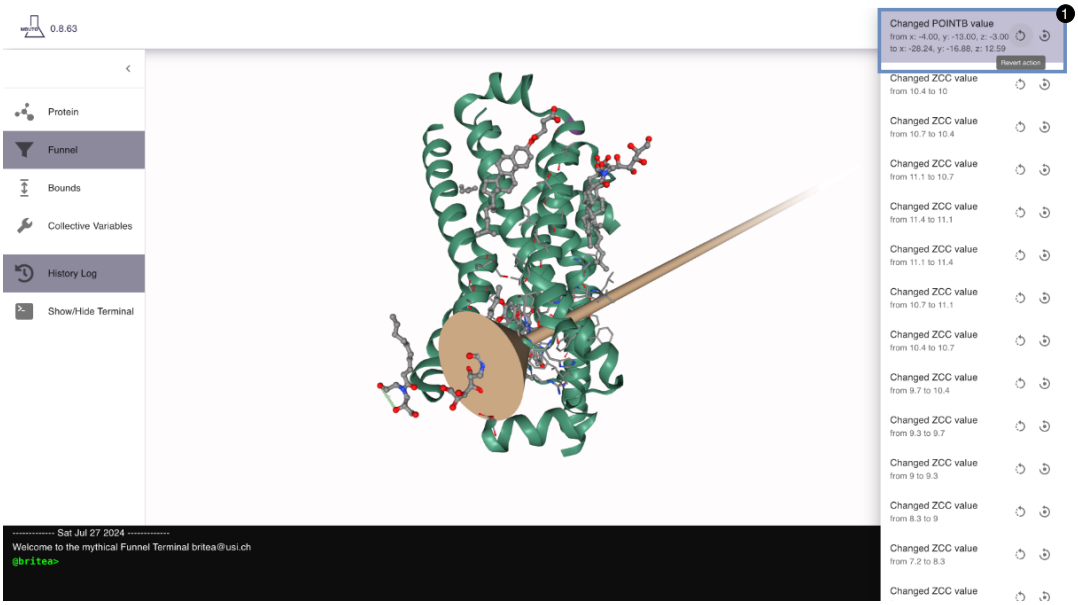


FIGURE 4.21: Example of funnel parameters (refers to Table 4.6). and history management view.

The second action is the reset state, a button next to the revert button shown in box **1)** in Figure 4.21, which completely resets the state representing the moment in history when the change was made. This action resets all parameters in the history state. For instance, following a state with the values presented in Table 4.7, even if the value changed is the alpha parameter, by using this feature, all other parameters will also be reset to this state.

Field	Value
PointA	-12.37, -22.93, 36.57
PointB	-28.24, -16.88, 12.59
Alpha	0.8
RCyl	1.3
Zcc	7

TABLE 4.7: Changed funnel parameters example 2).

The implementation of this feature uses the `CommandLineInterface` context component to send commands from and to React components overall. A command is an object containing a field **type** as a string to classify the command and the **value** field, a string containing all values inserted after the command keyword. As Figure 4.22 shows, when a command event is sent, the CLI Context component takes the command and updates the internal state. Once the command has been updated, the `StructureComponent` gets triggered by the command update and invokes a command dispatcher that performs the correct behavior for the specific command given, updating the molecular structure and applying the changes on the canvas.

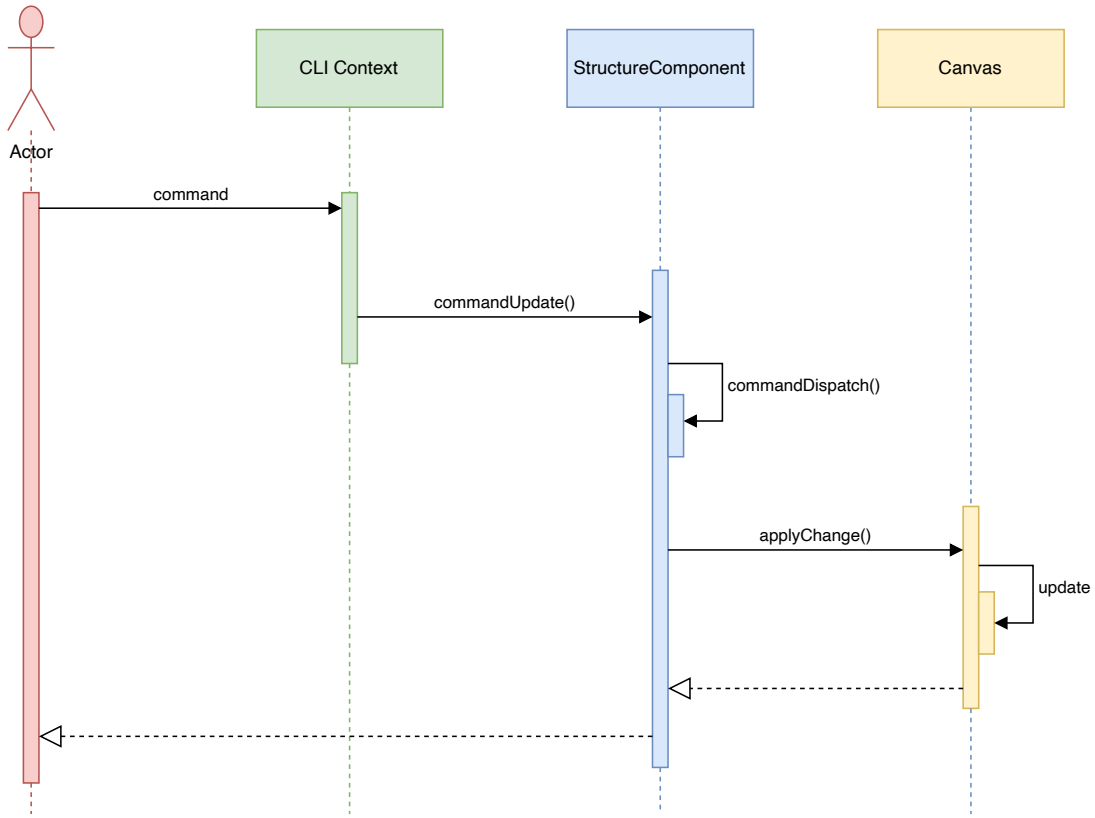


FIGURE 4.22: Command line interface sequence diagram.

With the aforementioned process, we can exploit it to have a history of the configuration states. We store within the CLI Context an array of configuration states described in Table 4.4, and every time the user submits an action to revert or to reset a state, we take the configuration state and we apply the related changes based on the type of action required. The action type is an enumeration within the different data in a configuration state, thus on adding a new state, we set the action field in Table 4.4 with the respective enumeration value, and then when a revert action is required we know which configuration value has to be updated.

Furthermore, the CLI context is also used to manage all commands sent from the terminal component view in the bottom sidebar as previously specified in Figure 4.4.

Chapter 5

User evaluation

In this chapter, we delve into the comprehensive user evaluation. This study involved gathering detailed feedback and insights from our target audience, providing a foundation for refining our software. Moreover, the purpose of this evaluation was to assess usability where the users perform with their own machines and applications of their choice, such that we implicitly study the portability of our tool through different browsers used. Moreover, identify functional issues where we aim to spot critical bugs such as API errors in specific situations, corrupted PLUMED files, or reporting erroneous information to the user. Finally, we collect feedback and we evaluate the results to improve our system. The user evaluation employs a questionnaire with an associated task that has to be done by the participants. Both the evaluation task and the questionnaire, implemented through the survey platform Qualtrics, have been delivered to the users via e-mail invitation. We did not rely on a specific user evaluation technique, such as the System Usability Scale (SUS), since we are in the preliminary phase of the system, and considering the domain specificity of the tool, we intend to achieve a more stable and complete version of the software. Due to the time and resource constraints, as well as the difficulty of finding users with the required experience for our user evaluation, we choose to use a simpler and quicker questionnaire to ensure that the user utilizes the main features and provides feedback. This approach allows us to gather a lot of feedback during the final open question. However, we design a similar questionnaire used in the standard system usability evaluations.

5.1 Target users

The evaluation task has been formulated for users with background knowledge of computational biology in research, can use relative tools in the field, such as VMD and PyMOL, and are proficient with the concepts of molecular structures. To diversify the results, we contacted several research groups from four different universities that study computational biology and employ molecular dynamics for their research activities.

5.2 Background questionnaire

The background questionnaire inserts the users in a position that allows us to compare them to each other. For this reason, the evaluation starts with questions directed to the user to better understand and discern its background. In this manner, we can assess the theoretical knowledge and expertise of the users by using tools of the users with molecular visualization and FM simulations. This distinction is important because users with different proficiency in the theory and practical aspects have different judgments on their evaluation. This part of the evaluation includes screening questions and preliminary questions.

Q1) Do you know any molecular tools such as VMD, PyMOL, etc.?
○ Yes ○ No

Q2) How would you evaluate your proficiency with VMD from 1 to 5?
No proficiency ○ ○ ○ ○ ○ High proficiency

Q3) Do you have knowledge about Funnel Metadynamics?
○ Yes ○ No

Q4) Do you know any problems related to the usage of VMD to prepare a Funnel Metadynamics simulation?
Text field to answer this open question.

Q5) How much confidence do you have in preparing a Funnel Metadynamics simulation?
No confidence ○ ○ ○ ○ ○ High confidence

The first question **Q1)** is a screening question to confirm that the user knows how molecular visualization tools such as VMD, PyMol, and many others work in general. If the user does not confirm the knowledge about molecular visualization, the questionnaire ends. Then, a set of preliminary questions starts with question **Q2)**, a self-assessment where the user declares his proficiency with the VMD, which is the software used in preparing FM simulations. Question **Q3)** verifies if the user knows the FM theoretical approach, and this is important since the tool developed in this thesis is a general molecular visualization software specialized in preparing FM simulation. Nevertheless, we take all feedback into account, but we expect to give more weight to the ones with FM knowledge. Question **Q4)** is an open question that provides feedback on the existing approach before starting the evaluation such that the user is guided to reason about the problem he encountered in his experience with VMD. Therefore, if the user has knowledge about preparing FM simulation through VMD, they are able to assess the usability of the product better, and we can compare at a later time the possible advantages or disadvantages of the new solution. Last but not least, question **Q5)** is a range question from 1 to 5 where the user states his confidence in preparing an FM simulation file. From this question, we can divide users who, for example, know only the theoretical part from the ones who also know the practical phase. For the participants with lower knowledge of the FM, we prepared an example to follow with all the parameters needed.

5.3 Evaluation task

Once the user completes the background questionnaire, a defined task must be completed to complete the questionnaire shown in Figure 5.1. Every participant has been provided with a link to the system and an account. The task requires **1)** the user to log in first, then create a new discovery in the dashboard and load the protein under study. After successfully loading the molecular structure, the participant is then asked to **2)** build a funnel and choose the anchor point. As soon as the user builds the funnel, it should appear on the view and can move to the next step, the definition of the bounds **3)**. When the view contains the funnel and the bounds, the user can move the **4)** step, choosing the collective variables. Once CVs are selected, the user can set the simulation parameters **5)**. Finally, after defining the fifth step, the user can export the PLUMED file **6)**.

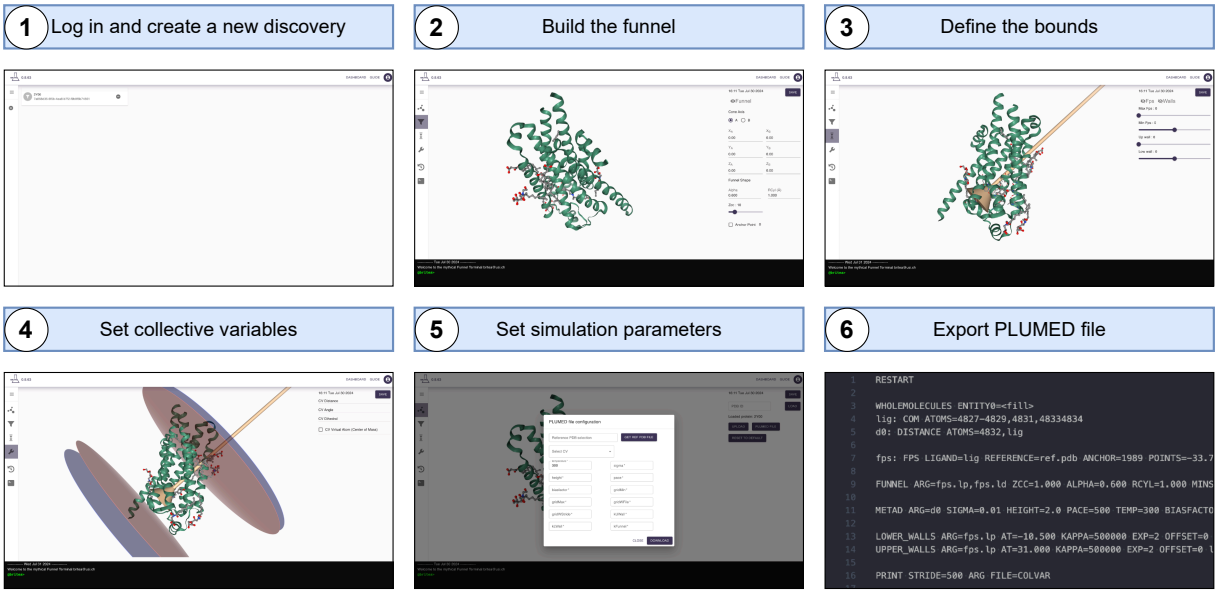


FIGURE 5.1: Overview of the phases of the user evaluation task.

5.4 Evaluation questionnaire

Once the user has finished the tasks, the set of follow-up questions starts.

Q7) On a scale of 1 (very hard) to 5 (very easy), how would you rate your overall experience with the tool?
very hard ○ ○ ○ ○ ○ very easy

Q8) On a scale of 1 (very hard) to 5 (very easy), how easy was to learn the tool?
very hard ○ ○ ○ ○ ○ very easy

Q9) On a scale of 1 (very hard) to 5 (very easy), how user-friendly was the tool?
very hard ○ ○ ○ ○ ○ very easy

Q10) On a scale of 1 (very hard) to 5 (very easy), how easy was it to find the features you need?
very hard ○ ○ ○ ○ ○ very easy

Q11) On a scale of 1 (very hard) to 5 (very easy), how usable was the tool?
very hard ○ ○ ○ ○ ○ very easy

Q12) On a scale of 1 (very hard) to 5 (very easy), how easy was to build the funnel and set the bounds?
very hard ○ ○ ○ ○ ○ very easy

Q13) On a scale of 1 (very hard) to 5 (very easy), how easy was to define the collective variables?
very hard ○ ○ ○ ○ ○ very easy

Q14) On a scale of 1 (very hard) to 5 (very easy), how easy was to set the simulation parameters and export the PLUMED file?
very hard ○ ○ ○ ○ ○ very easy

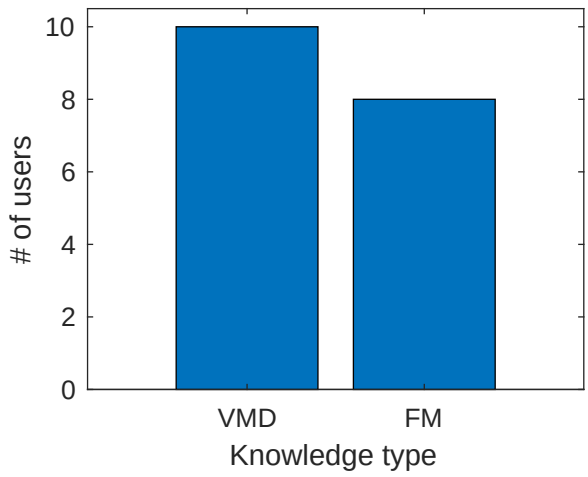
Q15) Do you have any suggestions for the tool or did you find any bug to report?
Text field to answer this open question.

The first question **Q7)** is to assess the overall experience and how the user was satisfied using the tool. The preparation of the FM simulation remains a complex process in itself, and the tool has to be learned in order to be used properly. For this reason, with question **Q8)** we verify how easy it is to learn. The following question **Q9)** describes how user-friendly the tool is and with question **Q10)** we can understand how easily users found the features they needed to complete the task. Question **Q11)** rates the usability of the system. After these general questions, three more focused questions are in the questionnaire: questions **Q12)**, **Q13)**, **Q14)**. They assess the easiness of the three main steps during the pre-processing phase: build the funnel and set the bounds, define the collective variable, set simulation parameters, and export the final file. Finally, the questionnaire asks the user to report any suggestions or bugs found during the task in question **Q15)**.

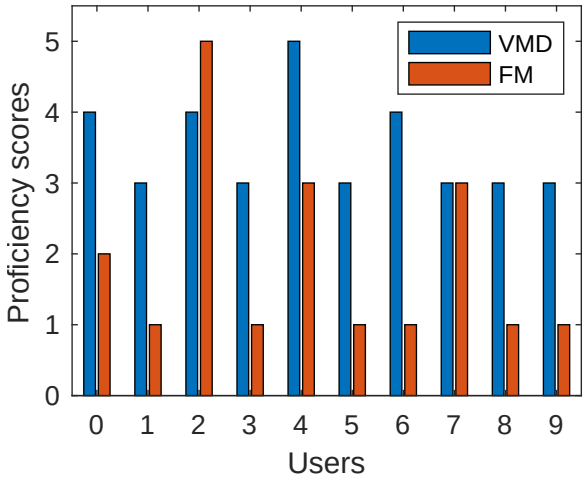
5.5 Results

In this section, we report the outcome of the questionnaire and the scores range from one to five. We contacted 15 participants, and 10 of them answered our questionnaire. The raw results are reported in Appendix E. However, our sample allows us to draw a representative and varied outline. From the first question, we gained insights into the user familiarity with molecular visualization (reported with the VMD acronym as our standard tool) and FM concepts. According to the results represented in Figure 5.2, we observe that, generally, most of the users enrolled in the evaluation task have knowledge about VMD rather than FM, even though 80% of the participants declare to be aware of both. On the other hand, the self-reported level of proficiency for VMD tools spans from average to high while, except for one user, the proficiency for FM simulations is generally average, if not lower.

The evaluation questionnaire results collected let us understand how the system was perceived by the users in terms of usability and learnability, as well as, specifically, how the three main features were perceived. Precisely, we divide the outcome into two different categories: the system usability metrics from **Q7** to **Q11** and the feature related metrics from **Q12** to **Q14**. Table 5.1 presents the descriptive statistics for the set of questions from **Q7** to **Q14**. We can notice that the median and the mean are generally high, indicating positive feedback, except for **Q13**, which shows lower scores. More intuitively, Figures 5.3 confirm that, in general, the tool was perceived very well, and the weakest feature is the definition of the collective variables.



(A) Comparison on the user knowledge

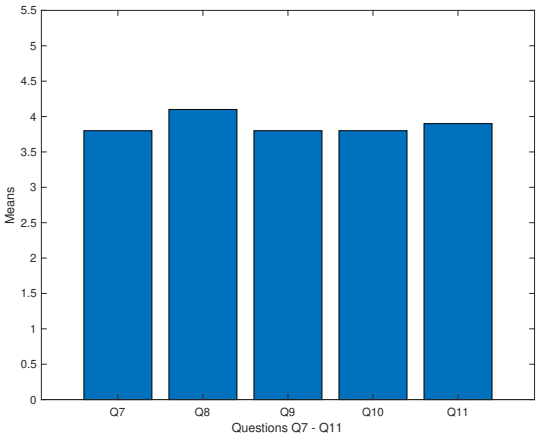


(B) Comparison on the user tool proficiency

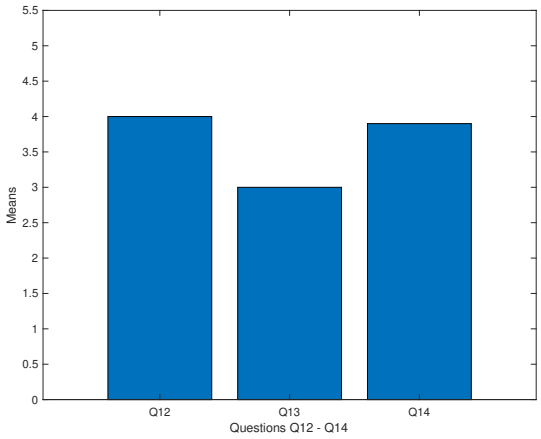
FIGURE 5.2: User metrics comparisons.

Question	Min	Max	Median	Mean	Mode
Q7	3.0	5.0	4.0	3.8	3.9
Q8	3.0	5.0	4.0	4.1	5.0
Q9	2.0	5.0	4.0	3.8	4.0
Q10	3.0	5.0	4.0	3.8	3.0
Q11	3.0	5.0	4.0	3.9	4.0
Q12	3.0	5.0	4.0	4.0	4.0
Q13	1.0	5.0	2.5	3.0	2.0
Q14	1.0	5.0	4.5	3.9	5.0

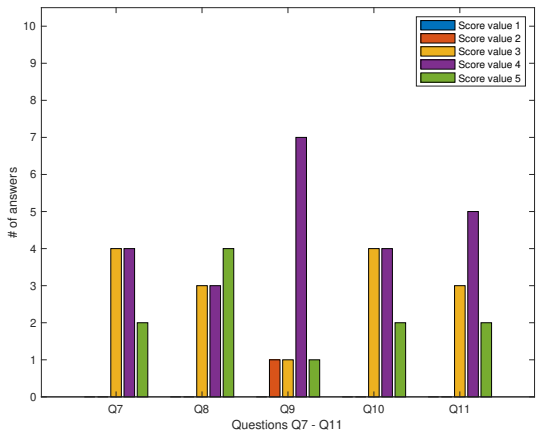
TABLE 5.1: Descriptive statistics of the questions with scores.



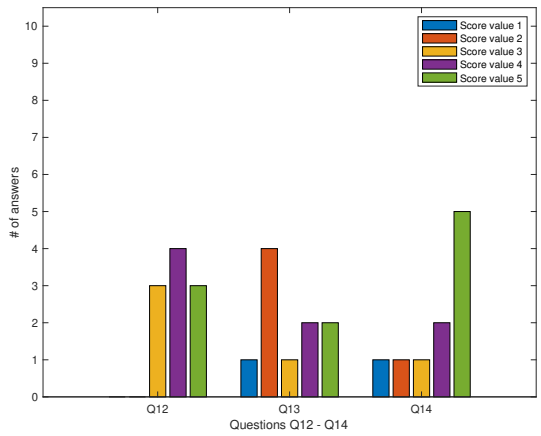
(A) System usability metrics means comparison.



(B) System feature metrics mean comparison.



(C) Number of answers by scores from Q7 to Q11.



(D) Number of answers by scores from Q12 to Q14.

FIGURE 5.3: System metrics mean comparison.

5.6 Considerations

In the previous section, we provided an overview of the results, and in this section, we discuss our considerations. While we retrieved many metrics, as described in the results, the evaluation questionnaire was supplied with a final open question about the experience where users have provided constructive feedback for improving the user interface, adding advanced features, and addressing bugs to enhance the overall experience. The considerations are based on the scores obtained from the questionnaire compared with the suggestions given by the users.

Overall, while the program is praised for its intuitive design and certain functional aspects, there are concerns regarding usability, precision, and bugs. Despite some bugs, all users were able to complete the task, which is an important outcome because it proves the validity and the efficacy of the system, considering that most of the users declared to be inexperienced in preparing an FM simulation with the current tool.

The weaker core functionality is the definition of the collective variables, which is comprehensible since it is the most interactive one. At first glance, the user gets the view with a list of CVs to define, but without properly reading the *Guide* section, there is no information on how to select the desired atoms, and this should be improved. On the other hand, setting the funnel position and the simulation parameters have been well perceived. An interesting outcome can be seen from **Q4**, where two users highlighted how the user interface of *FMAP GUI* is unintuitive and the absence of an import feature to quickly modify funnel parameters in the *FMAP GUI*. The feedback received through the final open question is very useful and effective, which is a good indication of the effectiveness of the questionnaire design. The latter was intended to guide the user, making it easy to reason about each step. Now we summarize the user suggestions, dividing them into two categories: suggestions and bugs.

5.6.1 Suggestions

In general, users appreciate the ease of retrieving and loading proteins using PDB IDs. Similarly, building the funnel shape and setting the bounds resulted in an intuitive and easy-to-understand approach, but some comments were made on this step. A user reported that it would be beneficial to add an input next to the "anchor point" to manually input the serial, as identifying and clicking on it within the interface can be challenging sometimes. Some users suggested implementing transparent visualization for the funnel so that the atoms inside it could be seen. The absence of the x, y, and z coordinate axes also hinders precise adjustments, making it difficult to change points A and B accurately without a clear reference. On that matter, adding a warning during the selection mode for points A, B, and the anchor point would help prevent unintended funnel movements and make it clearer what the user is selecting. As an improvement of the funnel positioning, a user proposed an additional easier way of placing it. Enabling mouse rotation and introducing drag-and-drop functionality for updating funnel coordinates would significantly enhance usability and would greatly assist in placing the funnel. Concerning the definition of the collective variables, users complained that selecting CVs was not very intuitive, though no specific reasons were reported. The only concern mentioned was the difficulty in defining the center of mass, particularly when toggling the tick to enable the selection. A few suggestions were made for the simulation parameters window. Instead of placing the button to open the simulation parameters window in the first tab where the user loads the molecular structure, it should be positioned at the end of the funnel setup process. Finally yet importantly, a user made a very interesting suggestion about adding an option to export to a ChimeraX[1] session that would be beneficial for generating publication figures since it has significant graphical performance. It is a program developed by Meng et al. [43] for the interactive molecular visualization and analysis of molecular structures and associated data. This would be a great addition to the tool since ChimeraX is widely used for virtual reality, light-sheet microscopy, and medical imaging data in the life sciences field. Finally,

[1] https://www.cgl.ucsf.edu/chimerax/

users made some suggestions on how the *Guide* page is displayed. The latter should not occupy the entire window, instead, it would be more useful to have it on the side or in a popover so that people can always follow instructions.

Chapter 6

Conclusion

In this thesis, we started looking at the drug development process and how it is expensive financially and time-wise, where each phase is crucial to have a successful final product. Researchers, during the *Drug discovery*, are always investigating new promising drugs, basing their studies on the possible interactions that ligands and targets might establish. In that regard, the key factor is the binding affinity, and in order to study it and speed up the process, we introduced two categorized methodologies: experimental and computational approaches. We focused on computational methods, in particular FM simulation, a well-known technique to compute the binding affinity and have insights on the ligand-protein interaction. In Chapter 2, we discussed the evolution of MD simulations and bioinformatics software. Initially developed in the 1950s, MD simulations have advanced significantly with computing power improvements, leading to software like CHARMM and GROMACS. However, while bioinformatics tools have evolved, challenges remain in software engineering practices. Issues include a general lack of software documentation, formal training for bioinformatics programmers, and user interfaces slowing the research activities. This highlights the importance of adopting better software development practices and improving user experience to enhance research efficiency in bioinformatics. Furthermore, we thoroughly pictured the three phases of the FM simulation: pre-processing, simulation, and post-processing. Then, we delved into FM pre-processing with the aim of improving this specific phase by designing a tool to improve its user-friendliness, designing a data model to store the configurations, and speeding up the creation of inputs for this computational technique.

In Chapter 3, we presented the design and architectural choices, and in Chapter 4, we devised an overview of the implementation through defined user cases. This was accomplished by using the NGLviewer library, enabling an interactive environment with molecular visualization. Throughout the development, continuous user feedback was received from experts in the field to minimize functionality errors, as depicted in Figure 3.3. Lastly, in Chapter 5, we described in detail the user evaluation that was done with targeted users who had knowledge of the field, using a defined questionnaire divided into *Background* and *Evaluation* to assess the validity of the system and its usability.

6.1 Future work & Considerations

This section discusses possible enhancements to our system to overcome current limitations and potential features to complete the entire FM protocol.

6.1.1 User evaluation feedback integration

We performed a user evaluation of the system to assess its usability and user-friendliness, and we received valuable feedback. Since we could not fully integrate all the feedback into the system, the first improvement is to incorporate all user evaluations completely.

Bugs

The website form for configuring the PLUMED file is generally perceived as intuitive and well-designed, though it may contain some bugs, which are acceptable for a preliminary release. However, several issues need to be addressed:

- Users have pointed out a general problem with the ability to change decimal places in the number inputs in the funnel building panel. This is due to a small oversight in setting the min, max and step values in the Material UI component text input field.
- A user reported that when preparing a distance CV between the two centers of masses of the protein and ligand, the index numbers were not printed in the PLUMED file, and both COMs were incorrectly labeled as *lig* instead of their actual respective and unique label. The correct format would have been to have *lig* for the ligand and *com1* for the protein. This problem was raised as the system was not prepared to manage two COMs from different structures, but currently, only one COM is used for the ligand when building the PLUMED file.
- The exported file lacks the correct syntax in the PRINT command (ARG=*), which is due to an error in the template format of the API responsible for building the output file.

6.1.2 NGLview library improvement

In Chapter 4, we have seen how the user can define a CV as a distance, angle, torsion, or center of mass. This is done by selecting atoms to build these geometric measurements. Theoretically, when a user defines a center of mass, it can be used as an atom to set up a distance, angle, or torsion. During the system's development, we represented the COM entity as a geometrical sphere, but the library is limited to instantiating a measurement between atom objects and non-atom objects such as a sphere. We have partially solved this problem by collecting the atom indices while they are picked and the COM's identifier object. This only stores the CV configuration but has no visual effect since the library class for representing measures is not possible with non-atom objects. For this particular reason, we submitted a GitHub issue [1] to their repository to confirm the lack of the library design. Thus, an improvement is to update the library, enhance it with the aforementioned feature, and submit a pull request. Last but not least, we have seen that having 3D coordinate axes is fundamental for a space reference when setting the funnel shape, the authors of the library already know about the absence of this feature [2].

6.1.3 Launch simulation feature

In our system, we export the PLUMED file once the user completes the pre-processing configuration. After that, the user shares or sends the file to the dedicated hardware and can launch the simulation. The idea is to add the possibility of connecting our system's backend to the machine performing the simulation, potentially using *gRPC*, to send and launch everything from our system and get real-time information about the progression.

6.1.4 Post-processing phase integration

In this thesis, we focused on the pre-processing phase of the FM protocol. Despite that, the post-processing phase could be integrated into our system to analyze the simulation output directly. Therefore, we must introduce a new frontend component in the GUI devoted to the simulation analyses and extend the existing API to load, store, and provide the data.

[1] https://github.com/nglviewer/ngl/issues/1036
[2] https://github.com/nglviewer/ngl/issues/998

6.1.5 Extended application

The platform will be tested for the FM protocol, but potentially, it has the structure to integrate the pre-processing of other enhanced sampling techniques based on CV usage.

6.2 Epilogue

We have thoroughly delved into the ligand-protein binding domain from a molecular biology perspective. Starting with the advantages during the drug development process and ending with how we can gain insight by performing molecular simulations, finally uncovering the FM protocol. In this thesis, we presented the design and implementation of a web platform allowing users to conduct the pre-processing phase of the FM protocol. Through this tool, we enhanced the user experience for preparing the PLUMED input file to run an FM simulation. We also reduced the setup process complexity by converging to a web-based approach instead of relying on several components. Finally, we added data persistence to every FM configuration such that, in case the user needs to rerun the simulation, he can just fine tune the previous one.

Appendix A

CQL Queries

This section illustrates the queries used in the backend to insert and retrieve data from the ScyllaDB instance required by the system. The queries are implemented using the Cassandra Query Language (CQL).

A.1 User queries

Create a new Account.

```
INSERT INTO accounts (email, password, creationtimestamp, status, discoveries)
VALUES (String, String, Time, Int, Map<UUID, Int>) IF NOT EXISTS
```

Result: Error or Account created successfully.

Retrieve Account information for internal use to check and validate account actions.

```
SELECT email, password, creationtimestamp, status
FROM accounts
WHERE email = [String]
```

Result: Error or Account information.

A.2 Discoveries queries

This is a batch that first adds the new discovery in the Account Table and then inserts the new discovery in the Discovery Table.

```
UPDATE accounts
SET discoveries = discoveries + [Map<UUID, Int>]
WHERE email = [String]
```

```
INSERT INTO discoveries (id, account_email, protein_name, discovery_name,
    creationtimestamp, lastupdatetimestamp, status, radius, b_point, d_point, t_point,
    alpha, r_cyl, zcc, walls, fps)
```

```
VALUES (UUID, String, String, String, Time, Time, Int, Int, Array<Double>, Array<Double>,
    Array<Double>, Double, Int, Double, Array<Double>, Array<Double>)
```

Result: Error or the created discovery with default values.

This is a batch where it first gets the discoveries ID from the account and then selects all user discoveries.

```
SELECT discoveries
FROM accounts
WHERE email = [String]
```

```
SELECT id, protein_name, discovery_name
FROM discoveries
WHERE account_email = [String] AND id = [UUID]
```

Result: Error or a list of user discoveries.

Retrieve the user discovery information associated with the given discovery UUID and user email.

```
SELECT id, account_email, protein_name, creationtimestamp, lastupdatetimestamp, radius,
    b_point, d_point, t_point, alpha, r_cyl, zcc, fps, walls, distance_atoms, angle_atoms,
    dihedral_atoms, com_atoms, anchor_point
FROM discoveries
WHERE account_email = [String] AND id = [UUID]
```

Result: Error or a discovery information.

Update discovery values associated with the given ID.

```
INSERT INTO discoveries (id, account_email, protein_name, discovery_name,
    creationtimestamp, lastupdatetimestamp, status, radius, b_point, d_point, t_point,
    alpha, r_cyl, zcc, walls, fps, distance_atoms, angle_atoms, dihedral_atoms, com_atoms,
    anchor_point)
VALUES (UUID, String, String, String, Time, Time, Int, Int, Array<Double>, Array<Double>,
    Array<Double>, Double, Int, Double, Array<Double>, Array<Double>, Array<Array<Int>>,
    Array<Array<Int>>, Array<Array<Int>>, Array<Array<Int>>, Int)"
```

Result: Error or discovery values updated.

Soft delete a discovery associated with the given discovery UUID by updating its status to one, the status value for representing a deleted item.

```
UPDATE accounts
SET discoveries[UUID] = 1
```

```
WHERE email = [String]
```

Result: Error or discovery values updated.

Appendix B

API

This section shows the API used by the system for the *Account* and *UserDiscovery* instances.

B.1 Account API

An `Account` instance is made of the following:

- `email` ∈ String: An Account email for the account.
- `creationtimestamp` ∈ Timestamp: A timestamp used to store the Account creation date.
- `password` ∈ String: The Account hashed password.
- `discoveries` ∈ Map<UUID, int>: A map with UUID of the Account discoveries and their statuses (active, disabled, deleted).

The system provides the following API for operating on Accounts:

- `POST /auth/register`
  - Register a new Account.
  - Response: A string value for the creation success status.
- `POST /auth/login`
  - Handle the Account login.
  - Request:
    - Body:
      - `email` ∈ String: The Account email.
      - `password` ∈ String: The Account password.
  - Response:
    - `access_token` ∈ String
    - `access_token_expires_at` ∈ String
    - `User`

B.2 User Discoveries API

An `UserDiscovery` instance is made of the following:

- `id` ∈ String: The UUID of the user discovery.

- `account_email` ∈ String: The Account email associated with the user discovery.
- `protein_name` ∈ String: The protein name to be loaded into the view.
- `discovery_name` ∈ String: The user discovery name defined on discovery creation.
- `creationtimestamp` ∈ Timestamp: A timestamp used to store the UserDiscovery creation date.
- `lastupdatetimestamp` ∈ Timestamp: A timestamp used to store the UserDiscovery last update date.
- `status` ∈ Integer: The UserDiscovery status value.
- `alpha` ∈ Double: The UserDiscovery alpha value.
- `anchor_point` ∈ Integer: The UserDiscovery anchor point index value.
- `radius` ∈ Integer: The UserDiscovery radius value.
- `zcc` ∈ Double: The UserDiscovery zcc value.
- `r_cyl` ∈ Double: The UserDiscovery radius of the cylinder value.
- `walls` ∈ List<Double>: The UserDiscovery walls list for the bottom and top walls.
- `fps` ∈ List<Double>: The UserDiscovery fps list for the min and max fps values.
- `distance_atoms` ∈ List<List<Integer»: The UserDiscovery distance atoms list containing a list of two atoms indices.
- `angle_atoms` ∈ List<List<Integer»: The UserDiscovery angle atoms list containing a list of three atoms indices.
- `dihedral_atoms` ∈ List<List<Integer»: The UserDiscovery dihedral atoms list containing a list of four atoms indices.
- `com_atoms` ∈ List<List<Integer»: The UserDiscovery center of mass atoms list containing a list of atoms indices representing the selected center of mass by the user.
- `b_point` ∈ List<Double>: The UserDiscovery bottom cone point selected by the user.
- `d_point` ∈ List<Double>: The UserDiscovery directional cone point selected by the user.
- `t_point` ∈ List<Double>: The UserDiscovery top cone point computed.

The system provides the following API for operating on UserDiscovery under `/api/v1` with authorization:

- `POST /user/discoveries`
  - Create a new UserDiscovery.
  - Request:
    - Headers: `Authorization`
    - Body:
      - `discoveryName` ∈ String: The discovery name used to denote the UserDiscovery.
  - Response: an `UserDiscovery`.
- `GET /user/discoveries`

  - Retrieve all UserDiscovery associated with the account.
  - Request:
    * Headers: `Authorization`
  - Response: a list of `UserDiscovery>`.

- `GET /user/discoveries/:id`
  - Retrieve the UserDiscovery associated with the provided ID.
  - Request:
    * Headers: `Authorization`
  - Response: an `UserDiscovery`.

- `PUT /user/discoveries/:id`
  - Create a new UserDiscovery.
  - Request:
    * Headers: `Authorization`
    * Body:
      · `userDiscovery` $\in$ UserDiscovery: The UserDiscovery with updated values.
  - Response: an `UserDiscovery`.

- `DELETE /user/discoveries/:id`
  - Create a new UserDiscovery.
  - Request:
    * Headers: `Authorization`
  - Response: an `UserDiscovery`.

- `GET /user/discoveries/:id/file`
  - Create a new UserDiscovery.
  - Request:
    * Headers: `Authorization`
  - Response: a Blob containing the PLUMED instructions.

Appendix C

Deployment and CI/CD files

In this section we collected all the configuration files we used for the deployment and CI/CD pipelines. In Listing C.1 we reported the Dockerfile for the server, while in Listing C.2 for the client side. Listing C.3 shows how each container is configured. Finally, Listing C.4 presents the GitHub action configuration.

LISTING C.1: Backend Docker file.

```
FROM golang:1.21 AS base

FROM base AS builder

WORKDIR /usr/src/app

COPY go.mod go.sum ./

RUN go mod download

COPY . /usr/src/app


# swagger docs
RUN go install github.com/swaggo/swag/cmd/swag@latest
RUN swag init

RUN CGO_ENABLED=0 GOOS=linux go build -o /mbuto-backend

FROM base

EXPOSE 8080
CMD ["/mbuto-backend"]

WORKDIR /usr/src/app

COPY --from=builder /mbuto-backend /mbuto-backend
COPY --from=builder /usr/src/app/docs .


RUN addgroup --system --gid 1001 mbuto
RUN adduser --system --uid 1001 mbuto-user
USER mbuto-user
```

LISTING C.2: Frontend Docker file

```
FROM node:lts-hydrogen AS base

FROM base AS builder
```

```
WORKDIR /usr/src/app

COPY . /usr/src/app

RUN npm install
RUN npm run build

FROM nginx:1.23.1-alpine

ENTRYPOINT ["/docker-entrypoint.sh"]
CMD ["nginx", "-g", "daemon off;"]

COPY nginx.conf /etc/nginx/conf.d/default.conf.template
COPY docker-entrypoint.sh /
COPY --from=builder /usr/src/app/dist/ /www/data
```

LISTING C.3: Docker-compose file.

```
version: "3.7"
services:
  # traefik
  traefik:
    container_name: "mbuto-traefik"
    image: "traefik:v2.10"
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.websecure.address=:443"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.web.http.redirections.entryPoint.to=websecure"
      - "--entrypoints.web.http.redirections.entryPoint.scheme=https"
      - "--certificatesresolvers.myresolver.acme.tlschallenge=true"
      - "--certificatesresolvers.myresolver.acme.email=andrea.brites.marto@usi.ch"
      - "--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json"
      - "--providers.file=true"
      - "--providers.file.filename=/traefik-conf.yaml"
      - "--accesslog"
    ports:
      - "443:443"
      - "80:80"
      - "5051:8080"
    volumes:
      - "./letsencrypt:/letsencrypt"
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
      - "/home/andrea/traefik-conf.yaml:/traefik-conf.yaml:ro"
  # DB
  mbuto-db:
    container_name: mbuto-db
    image: scylladb/scylla:5.1
    restart: always
    command:
      - "--smp"
      - "2"
    volumes:
```

```
      - ./volumes/db/scylla:/var/lib/scylla
    environment:
      - TZ=Europe/Zurich
  # frontend
  mbuto-frontend:
    container_name: mbuto-frontend
    image: britea/mbuto:frontend-latest
    restart: always
    environment:
      - VITE_APP_BACKEND_HOSTNAME=mbuto-backend
      - VITE_APP_BACKEND_PORT=8080
      - VITE_APP_PROD=true
      - TZ=Europe/Zurich
    depends_on:
      - mbuto-backend
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.mbuto-frontend.rule=Host(`mbuto.si.usi.ch`)"
      - "traefik.http.routers.mbuto-frontend.entrypoints=websecure"
      - traefik.http.services.mbuto-frontend.loadbalancer.server.port=3000
      - "traefik.http.routers.mbuto-frontend.tls.certresolver=myresolver"
  # backend
  mbuto-backend:
    container_name: mbuto-backend
    image: britea/mbuto:backend-latest
    restart: always
    volumes:
      - ./app.env:/usr/src/app/app.env:ro
    depends_on:
      - "mbuto-db"
    environment:
      - TZ=Europe/Zurich
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.mbuto-backend.rule=Host(`mbuto.si.usi.ch`) && (PathPrefix(`/api`)
        || PathPrefix(`/auth`) || PathPrefix(`/docs`))"
      - "traefik.http.routers.mbuto-backend.entrypoints=websecure"
      - traefik.http.services.mbuto-backend.loadbalancer.server.port=8080
      - "traefik.http.routers.mbuto-backend.tls.certresolver=myresolver"
  networks:
    default:
      name: mbuto
```

LISTING C.4: Github Action workflow file.

```
name: Docker Image CI

on:
  push:
    tags:
      - "*"

jobs:
  build-frontend:
    runs-on: ubuntu-latest
    steps:
      - name: Login to DockerHub
```

```
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_PSW }}
      - name: Checkout
        uses: actions/checkout@v4.1.1
      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: ./frontend
          push: true
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/mbuto:frontend-latest
  build-backend:
    runs-on: ubuntu-latest
    steps:
      - name: Login to DockerHub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_PSW }}
      - name: Checkout
        uses: actions/checkout@v4.1.1
      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: ./backend
          push: true
          tags: ${{ secrets.DOCKERHUB_USERNAME }}/mbuto:backend-latest
```

Appendix D

User Guide

This is the user guide provided by the tool in the section *Guide*.

D.1 General

shift + click: recenter the view on the protein.
crtl + drag: drag the view.
left click: if you left click on the view, it will display all atoms within 5 angstrom of the clicked atom.
mouse: with left mouse pressed you can rotate the view.
scroll: with scroll you can zoom in/out the view.
selection + range: You can click on a residue or atom and get the atoms within a range expressed in angstrom, the default range is 5, but by pressing a number between 1 and 7 you can change this value range and the selection automatically shows the view according to the selected range.
shift+scroll: clip near the view based on the scroll delta.
scroll: zoom in/out of the view.
w | a | s | d: respectively move the view down, up, left, and right.
q | e: rotate the view to the left and to the right.
hover: highlight the amino acid on the protein ribbon representation.

D.2 Funnel

set funnel: by choosing A or B with radio buttons you can then select an atom to attach the position. Then through the other parameters you can tune the shape of the funnel.

D.3 Bounds

set walls: with the slider input for both low and upper walls (by default they are hidden).
set fps: with the slider input for both low and upper fps (by default they are hidden).

D.4 Collective Variables

ctrl + click or **right-click**: select atom for distance, angle or dihedral collective variable.
double right click or **double ctrl + click**: select last atom for the desired distance, angle or dihedral collective variable, if 2 atoms are selected it creates a distance, if 3 angle and with 4 dihedral.
center of mass: click the checkbox for starting the center of mass, then by just clicking atoms it will display the center of mass of the chosen atoms. When done just re-click the checkbox.

D.5 History Management

reset action: this action allows to reset the entire state to that point of the history.
reset state: this action allow to revert the change in respect to the current state.

D.6 Funnel Terminal

help: display all commands and their option. With the '-selection' flag, you can consult the selection language.
clear: clear the terminal component.
exit: exit, clear and close the terminal component.
filter [string]: filter the view by adding 'ball+stick' representation to the given selection [string] by keeping the protein as ribbon.
com [string]: apply the center of mass collective variable to the given selection [string], if the selection is invalid or does not provide any result it throws an error.
show [funnel, fps, walls, protein]: show if hidden the given view component, the protein, the funnel shape, the walls or the fps bounds.
hide [funnel, fps, walls, protein]: hide if currently shown the given view component, the protein, the funnel shape, the walls or the fps bounds.
history [number]: keep track of all actions taken on the funnel and protein and, by providing a number between 1 and 50 representing the number of action you want to go back.
reset [last, funnel, fps, walls]: completely resets the view and all the parameters needed to build it. If last option is given it reset to the last configuration saved. If funnel option is given to reset only the funnel parameters. If fps option is given to reset only the fps parameters. If walls option is given to reset only the walls parameters.

Appendix E

User evaluation questionnaire raw results

In this section, we present the raw results from the questionnaire conducted during the user evaluation phase.

E.1 Results from Q1 to Q3 and from Q5 to Q14

Responses included in Table E.1 are represented by a number (score from 1 to 5) or by a binary value (Yes/No).

User	Q1	Q2	Q3	Q5	Task	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14
1	Yes	4	Yes	2	Yes	3	3	2	3	4	3	2	1
2	Yes	3	Yes	1	Yes	3	4	4	3	3	4	1	4
3	Yes	4	Yes	5	Yes	5	5	5	4	4	4	4	4
4	Yes	3	Yes	1	Yes	5	5	4	4	5	5	5	5
5	Yes	5	Yes	3	Yes	4	4	3	5	4	3	2	3
6	Yes	3	Yes	1	Yes	4	3	4	3	3	4	5	5
7	Yes	4	Yes	1	Yes	4	5	4	4	4	4	3	5
8	Yes	3	Yes	3	Yes	3	3	4	3	3	5	2	2
9	Yes	3	No	1	Yes	3	4	4	4	4	5	2	5
10	Yes	3	No	1	Yes	4	5	4	5	5	3	4	5

TABLE E.1: Raw results from questionnaire from Q1 to Q3 and from Q5 to Q14

E.2 Results from Q4

This question was to gather information about existing problems in preparing an FM simulation through VMD. The raw textual answers from **Q4** are in the same order as E.1.

1. yes
2. I didn't use it until now
3. no
4. N/A
5. it could be useful to have the possibility to import again the plumed.dat file within the FMAP GUI in order to eventually quickly modify the funnel parameters, whitout rewriting them manually. Currently, only the ""Export"" option is available. It is also useful to automatically select the unit of measure whereby writing the plumed.dat file

6. The procedure is cumbersome and the user interface is not intuitive
7. Nerver prepare a FM on VMD
8. No
9. I have never prepared a Funnel Metadynamics simulation on VMD.
10. -

E.3 Results from Q15

This was an optional open question where participants could provide feedback regarding the tool experience and suggest improvements. Here, we report the raw textual answers.

1. -decimal places can not be changed using keyboard in Funnel building panel ; - funnel transparency could be added; - ctrl+z to Undo action could be added; - drag & drop the funnel (updating coordinates) could be very useful; - alpha parameter in the Funnel building panel seems to consider two times the actual angle; - COM sphere is not shown; - CV inserted in the PLUMED file export window cannot be deleted; - an ""export to Chimera Session"" option could be nice to generate publication figures
2. Add transparent visualization of the funnel. Add the possibility to select a list of residues. Change the selection of different collective variables. Put the window for saving plumed input file at the end of the setting of the funnel. Add a warning when the user is in selection mode for point A, point B and anchor point to avoid unwanted funnel movements.
3. 1. the axis of the coordinate space x,y,z are missing or at least i did not see them. Showing them is very important because it allows us to change the point A and B of the funnel easier using the axis as reference. 2. The terminal window was always open on the bottom and i could not close it, this blocked from my view some tools. For example, while setting the funnel ZCC was behind the terminal window and I could not reach it. Of course, i tried minimising the terminal window but it did not go away. 3.There seems to be a default minimum space between the lower wall and min.fps. Like, at a certain point, I cannot get them any closer because the software prevents it. 4. The guide screen should not take the whole window. It would be ideal if it was a pop up on the side of something like that, so that we can read and click at the same time following the instructions on the guide. 5. On the plumed input file, I prepared a distance CV between center of mass of protein and ligand. The index numbers were not printed in the plumed file and on top of that both COMs are named lig. 6. The plumed input file seems to be in Angstroms? Should be in nm.. 7. The plumed input file is missing the correct syntax in the PRINT command ARG=*... 8. The selection for the ref pdb are not clear. I wanted to select only the CA of the protein but it gave me an empty file as output..
4. -In the ""Guide"" section, add a paragraph about the syntax to use to select a specific kind of atom (es. C-alpha) to print in the ref.pdb file;-Add unit of measure for all the parameters (temperature, bounds ecc.) and convert automatically Angstrom in nm in the output file;-tool to display/undisplay residues sidechains;-I would like to see the xyz axis in the interface; BUG: I tried to define the COM of my protein but i couldn't visualize the resultant COM as a virtual atom;
5. Suggestions: 1. I would add a writable small window on the right to the ""anchor point"" in order to manually write the serial of the anchor point. It was not so easy to identify and click on it from the interface. 2. I would add an option to make the funnel shape transparent and not only visible or not visible. 3. I would add a section in ""Protein"" in order to modify the visualization rendering (stick,

lines and ball and stick). 4. Personally I do not understand the difference between point A (funnel base point) and point B (Funnel directional point). I haven't seen any changes when I was building the funnel. 5. The window relative to the setting up of the other CVs could be improved. It is not so quickly comprehensible.

6. N/A

7. Positioning the funnel by being able to rotate it with the mouse would help in positioning

8. N/A

9. 0) Creating a new discovery is very intuitive from the point of view of the interface. However, you ask me to provide a custom name for the discovery and that name is not displayed anywhere. I just see a box (corresponding to my discovery) labelled with a long lowercase pseudo-random alphanumeric sequence. 1) I like the way the protein is retrieved and loaded through the PDB ID. 2) As far as concerns the funnel generation, first it is not intuitive how to locate the funnel with the mouse cursor. Second, I don't like the slider as the only way to set up a Zcc. I cannot set it to be 1.8, as a minimum step makes it shifting from 1.6 right to 1.9. A text field should be inserted like for Alpha and RCyl. Also, these latter two options cannot be properly set using the keyboard, and the slider can increase/decrease them just respectively by decimals and units. Selection of anchor point is not responsive. 3) Setting of wall bounds is a bit buggy, the sliders do not work properly, and I managed to set some of them with a bit of luck and direction buttons on the keyboard. Again, I would suggest against using sliders, as it is difficult to set decimals properly. Fps values increase only by 0.5, thus it is not possible to work with intermediate values, e.g. 3.7. 4) Selection of the center of mass is not intuitive with the tick to be toggled (the same as selection of the anchor point). Selection fo CV is not intuitive and a bit buggy. 5) PLUMED file configuration is intuitive and well designed, although it may be a bit buggy, but still suitable for a preliminary release. 6) The user interface is generally intuitive and well designed. Colors and shapes are well balanced. However, when exploring the discovery space, it is not clear what the buttons UPLOAD and PLUMED FILE stand for. I mean, you could rephrase those buttons with more explicit expressions, like UPLOAD THIS, GENERATE PLUMED FILE, or arrange them in a way so that buttons attaining to similar tasks are located closer. The project is generally well designed and cleverly implemented. Some bugs can be detected during the execution of its preliminary goals on some machines, but it is generally acceptable in the scope of a preliminary, yet complete, implementation.

10. I would like to have the ability to move the funnel with arrows and mouse (like you can move a protein). It would be nice to have a live feed about the last action execution, perhaps in a small pop-up (with the possibility to silence it as user setting).

Glossary

AMBER Assisted Model Building with Energy Refinement is a molecular dynamics software package that simulates the AMBER force fields. 10

BFES Binding Free Energy Surface is the manifold that shows the free energy associated with the binding state of a ligand to its target. 13

CV Collective Variables are any structural parameter that can be measured throughout the simulation such as combination of distances, angles, dihedrals . 3, 9

DESMOND DESMOND is a software package to perform high-speed molecular dynamics simulations of biological systems. 10

FM Funnel Metadynamics is a binding free energy method. 2

FMAP Funnel Metadynamics Advanced Protocol is a well defined ethodology to predict the drug binding between a ligand and a target. 3, 17

FMAP GUI Funnel Metadynamics Advanced Protocol Graphical User Interface is the support tool for the Funnel Metadynamics pre and post processing phases. 3, 13

GROMACS GROningen MAchine for Chemical Simulations is a versatile package to perform molecular dynamics. 10

gRPC gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment.. 60

MD Molecular Dynamics is a computational method to simulate the interaction an. 6

NAMD Nanoscale Molecular Dynamics, formerly Not Another Molecular Dynamics Program, is a software for molecular simulations. 10

PBC Periodic Boundary Conditions are a set of boundary conditions to approximate an infinite system by using a unit cell preserving thermodynamic properties. 15

PDB Protein Data Bank is a database for the 3D molecular structures such as proteins. 13

PLUMED PLUMED is an open-source, community-developed library that provides a wide range of different methods such as enhanced sampling and free energy method. 10

VMD Visual Molecular Dynamics is molecular visualization program for displaying, animating and analyzing biomolecular system. 17

Bibliography

[1] Natesh Singh, Philippe Vayer, Shivalika Tanwar, Jean-Luc Poyet, Katya Tsaioun, and Bruno O. Villoutreix. Drug discovery and development: introduction to the general public and patient groups. *Frontiers in Drug Discovery*, 3, 2023. ISSN 2674-0338. doi: 10.3389/fddsv.2023.1201419. URL https://www.frontiersin.org/journals/drug-discovery/articles/10.3389/fddsv.2023.1201419.

[2] Joseph A DiMasi. The value of improving the productivity of the drug development process: faster times and better decisions. *Pharmacoeconomics*, 20:1–10, 2002. doi: 10.2165/00019053-200220003-00001. URL https://doi.org/10.2165/00019053-200220003-00001.

[3] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Jr. Edward W. Lowe. Computational Methods in Drug Discovery. *Pharmacological Reviews*, 66(1):334–395, 2014. ISSN 0031-6997. doi: 10.1124/pr.112.007336. URL https://pharmrev.aspetjournals.org/content/66/1/334.

[4] Mattia Bernetti, Matteo Masetti, Walter Rocchia, and Andrea Cavalli. Kinetics of drug binding and residence time. *Annual review of physical chemistry*, 70(1):143–171, 2019. doi: 10.1146/annurev-physchem-042018-052340. URL https://doi.org/10.1146/annurev-physchem-042018-052340.

[5] Vittorio Limongelli, Massimiliano Bonomi, and Michele Parrinello. Funnel metadynamics as accurate binding free-energy method. *Proceedings of the National Academy of Sciences*, 110(16):6358–6363, 2013. doi: 10.1073/pnas.1303186110. URL https://doi.org/10.1073/pnas.1303186110.

[6] Hiroaki Hata, Tran Duy, Mohamed Sobeh, and Akio Kitao. Binding free energy of protein/ligand complexes calculated using dissociation Parallel Cascade Selection Molecular Dynamics and Markov state model. *Biophysics and Physicobiology*, 18, 12 2021. doi: 10.2142/biophysico.bppb-v18.037.

[7] Stefano Raniolo and Vittorio Limongelli. Ligand binding free-energy calculations with funnel metadynamics. *Nature Protocols*, 15(9):2837–2866, 2020. doi: 10.1038/s41596-020-0342-4. URL https://doi.org/10.1038/s41596-020-0342-4.

[8] Brendan Lawlor and Paul Walsh. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. *Bioengineered*, 6(4):193–203, 2015.

[9] Charles E. Robertson, J. Kirk Harris, Brandie D. Wagner, David Granger, Kathy Browne, Beth Tatem, Leah M. Feazel, Kristin Park, Norman R. Pace, and Daniel N. Frank. Explicet: graphical user interface software for metadata-driven management, analysis and visualization of microbiome data. *Bioinformatics*, 29(23):3100–3101, 09 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt526. URL https://doi.org/10.1093/bioinformatics/btt526.

[10] Dhawal Verma, Jon Gesell, Harvey Siy, and Mansour Zand. Lack of software engineering practices in the development of bioinformatics software. *ICCGI*, 2013:57–62, 2013.

[11] Byron C. Wallace, Marc J. Lajeunesse, George Dietz, Issa J. Dahabreh, Thomas A. Trikalinos, Christopher H. Schmid, and Jessica Gurevitch. OpenMEE: Intuitive, open-source software for meta-analysis in ecology and evolutionary biology. *Methods in Ecology and Evolution*, 8(8):941–947, 2017. doi:

https://doi.org/10.1111/2041-210X.12708. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.12708.

[12] Werner J. Geldenhuys, Kevin E. Gaasch, Mark Watson, David D. Allen, and Cornelis J. Van der Schyf. Optimizing the use of open-source software applications in drug discovery. *Drug Discovery Today*, 11 (3):127–132, 2006. ISSN 1359-6446. doi: https://doi.org/10.1016/S1359-6446(05)03692-5. URL https://www.sciencedirect.com/science/article/pii/S1359644605036925.

[13] Illia Horenko, Edoardo Vecchi, Juraj Kardoš, Andreas Wächter, Olaf Schenk, Terence J. O'Kane, Patrick Gagliardini, and Susanne Gerber. On cheap entropy-sparsified regression learning. *Proceedings of the National Academy of Sciences*, 120(1):e2214972120, 2023. doi: 10.1073/pnas.2214972120. URL https://www.pnas.org/doi/abs/10.1073/pnas.2214972120.

[14] A. V. Popov and Yu. N. Vorob'ev. GUI-BioPASED: A program for molecular dynamics simulations of biopolymers with a graphical user interface. *Molecular Biology*, 44(4):648–654, Aug 2010. ISSN 1608-3245. doi: 10.1134/S0026893310040217. URL https://doi.org/10.1134/S0026893310040217.

[15] Kevin J Bowers, Edmond Chow, Huafeng Xu, Ron O Dror, Michael P Eastwood, Brent A Gregersen, John L Klepeis, Istvan Kolossvary, Mark A Moraes, Federico D Sacerdoti, et al. Scalable algorithms for molecular dynamics simulations on commodity clusters. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, pages 84–es, 2006. doi: 10.1109/SC.2006.54. URL https://doi.org/10.1109/SC.2006.54.

[16] G A Ross, C Lu, G Scarabelli, S K Albanese, E Houang, R Abel, E D Harder, and L Wang. The maximal and current accuracy of rigorous protein-ligand binding free energy calculations. *Communications Chemistry*, 6(222), 2023. doi: 10.1038/s42004-023-01019-9. URL https://doi.org/10.1038/s42004-023-01019-9.

[17] Tomas Hansson, Chris Oostenbrink, and Wilfred F van Gunsteren. Molecular dynamics simulations. *Current opinion in structural biology*, 12(2):190–196, 2002. doi: 10.1016/s0959-440x(02)00308-1. URL https://doi.org/10.1016/s0959-440x(02)00308-1.

[18] Vittorio Limongelli. Ligand binding free energy and kinetics calculation in 2020. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 10(4):e1455, 2020.

[19] Alessandro Barducci, Massimiliano Bonomi, and Michele Parrinello. Metadynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(5):826–843, 2011. doi: https://doi.org/10.1002/wcms.31.

[20] Paolo Conflitti, Stefano Raniolo, and Vittorio Limongelli. Perspectives on Ligand/Protein Binding Kinetics Simulations: Force Fields, Machine Learning, Sampling, and User-Friendliness. *Journal of Chemical Theory and Computation*, 19(18):6047–6061, 2023.

[21] David E Shaw, Martin M Deneroff, Ron O Dror, Jeffrey S Kuskin, Richard H Larson, John K Salmon, Cliff Young, Brannon Batson, Kevin J Bowers, Jack C Chao, et al. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM*, 51(7):91–97, 2008.

[22] Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the national academy of sciences*, 99(20):12562–12566, 2002.

[23] Marcella Iannuzzi, Alessandro Laio, and Michele Parrinello. Efficient exploration of reactive potential energy surfaces using Car-Parrinello molecular dynamics. *Physical Review Letters*, 90(23):238302, 2003.

[24] Herman JC Berendsen, David van der Spoel, and Rudi van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer physics communications*, 91(1-3):43–56, 1995. doi: https://doi.org/10.1016/0010-4655(95)00042-E.

[25] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016. doi: 10.1038/sdata.2016.18. URL https://doi.org/10.1038/sdata.2016.18.

[26] David A Case, Thomas E Cheatham III, Tom Darden, Holger Gohlke, Ray Luo, Kenneth M Merz Jr, Alexey Onufriev, Carlos Simmerling, Bing Wang, and Robert J Woods. The Amber biomolecular simulation programs. *Journal of computational chemistry*, 26(16):1668–1688, 2005. doi: 10.1002/jcc.20290. URL https://doi.org/10.1002/jcc.20290.

[27] The PLUMED consortium. Promoting transparency and reproducibility in enhanced molecular simulations. *Nature methods*, 16(8):670–673, 2019. doi: 10.1038/s41592-019-0506-8. URL https://doi.org/10.1038/s41592-019-0506-8.

[28] Gareth A Tribello, Massimiliano Bonomi, Davide Branduardi, Carlo Camilloni, and Giovanni Bussi. PLUMED 2: New feathers for an old bird. *Computer physics communications*, 185(2):604–613, 2014. doi: 10.1016/j.cpc.2013.09.018. URL https://doi.org/10.1016/j.cpc.2013.09.018.

[29] Massimiliano Bonomi, Davide Branduardi, Giovanni Bussi, Carlo Camilloni, Davide Provasi, Paolo Raiteri, Davide Donadio, Fabrizio Marinelli, Fabio Pietrucci, Ricardo A Broglia, et al. PLUMED: A portable plugin for free-energy calculations with molecular dynamics. *Computer Physics Communications*, 180(10):1961–1972, 2009.

[30] Pieter BT Neerincx and Jack AM Leunissen. Evolution of web services in bioinformatics. *Briefings in bioinformatics*, 6(2):178–188, 2005.

[31] Hamish Mcwilliam, Franck Valentin, Mickael Goujon, Weizhong Li, Menaka Narayanasamy, Jenny Martin, Teresa Miyar, and Rodrigo Lopez. Web services at the european bioinformatics institute-2009. *Nucleic acids research*, 37(suppl_2):W6–W10, 2009.

[32] Medha Umarji, Carolyn Seaman, A. Gunes Koru, and Hongfang Liu. Software Engineering Education for Bioinformatics. In *2009 22nd Conference on Software Engineering Education and Training*, pages 216–223, 2009. doi: 10.1109/CSEET.2009.44. URL https://doi.org/10.1109/CSEET.2009.44.

[33] Kristian Rother, Wojciech Potrzebowski, Tomasz Puton, Magdalena Rother, Ewa Wywial, and Janusz M. Bujnicki. A toolbox for developing bioinformatics software. *Briefings in Bioinformatics*, 13(2):244–257, 07 2011. ISSN 1467-5463. doi: 10.1093/bib/bbr035. URL https://doi.org/10.1093/bib/bbr035.

[34] D. Kane. Introducing agile development into bioinformatics: an experience report. In *Proceedings of the Agile Development Conference, 2003. ADC 2003*, pages 132–139, 2003. doi: 10.1109/ADC.2003.1231463. URL https://doi.org/10.1109/ADC.2003.1231463.

[35] David W Kane, Moses M Hohman, Ethan G Cerami, Michael W McCormick, Karl F Kuhlmman, and Jeff A Byrd. Agile methods in biomedical software development: a multi-site experience report. *BMC bioinformatics*, 7:1–12, 2006. doi: 10.1186/1471-2105-7-273. URL https://doi.org/10.1186/1471-2105-7-273.

[36] Davide Bolchini, Anthony Finkelstein, Vito Perrone, and Sylvia Nagl. Better bioinformatics through usability analysis. *Bioinformatics*, 25(3):406–412, 12 2008. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn633. URL https://doi.org/10.1093/bioinformatics/btn633.

[37] Katrina Pavelin, Jennifer A Cham, Paula de Matos, Cath Brooksbank, Graham Cameron, and Christoph Steinbeck. Bioinformatics meets user-centred design: a perspective. *PLoS computational biology*, 8(7):e1002554, 2012.

[38] Homa Javahery, Ahmed Seffah, and Thiruvengadam Radhakrishnan. Beyond power: making bioinformatics tools user-centered. *Commun. ACM*, 47(11):58–63, nov 2004. ISSN 0001-0782. doi: 10.1145/1029496.1029527. URL https://doi.org/10.1145/1029496.1029527.

[39] Joan C. Bartlett and Elaine G. Toms. Developing a protocol for bioinformatics analysis: An integrated information behavior and task analysis approach. *Journal of the American Society for Information Science and Technology*, 56(5):469–482, 2005. doi: https://doi.org/10.1002/asi.20136. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20136.

[40] Parmit K. Chilana, Carole L. Palmer, and Amy J. Ko. Comparing bioinformatics software development by computer scientists and biologists: An exploratory study. In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, pages 72–79, 2009. doi: 10.1109/SECSE.2009.5069165. URL https://doi.org/10.1109/SECSE.2009.5069165.

[41] Helen Berman, Kim Henrick, and Haruki Nakamura. Announcing the worldwide protein data bank. *Nature structural & molecular biology*, 10(12):980–980, 2003.

[42] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996. doi: 10.1016/0263-7855(96)00018-5. URL https://doi.org/10.1016/0263-7855(96)00018-5.

[43] Elaine C Meng, Thomas D Goddard, Eric F Pettersen, Greg S Couch, Zach J Pearson, John H Morris, and Thomas E Ferrin. UCSF ChimeraX: Tools for structure building and analysis. *Protein Science*, 32 (11):e4792, 2023. doi: 10.1002/pro.4792. URL https://doi.org/10.1002/pro.4792.