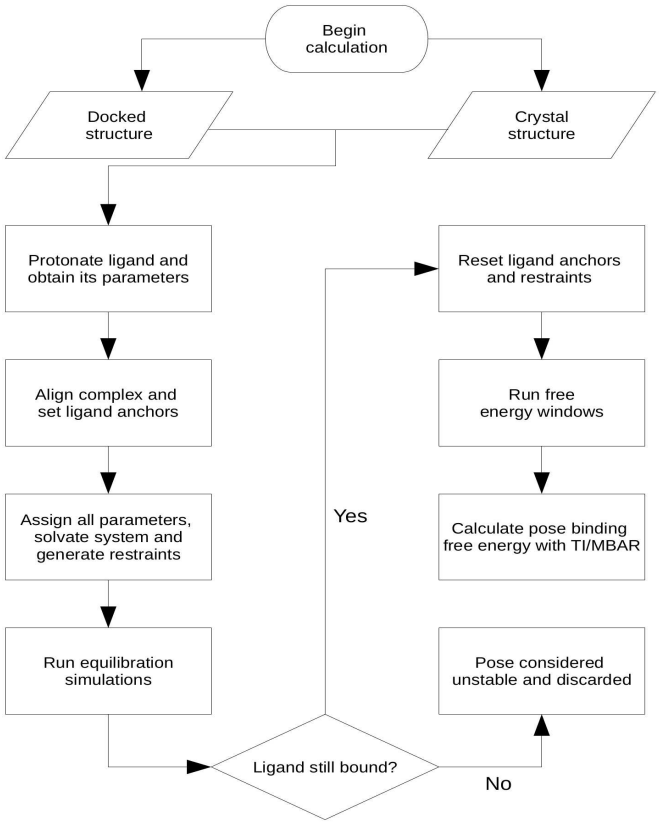
User guide for BAT.py – v2.2

1. Introduction

The Binding Affinity Tool (BAT.py) is designed to fully automate absolute binding free energy (ABFE) calculations, with a complete workflow starting only from a cocrystal structure or a docked complex. The building of the simulation boxes, generation of all the needed parameters, set up of the various simulation windows, running the simulations, and the final free energy analysis are all done without any manual interference. BAT.py can use either the *pmemd.cuda* sofware from AMBER20 [1], or the OpenMM 7.7.0 software [2], both showing very high performance on Graphics Processing Units (GPUs) at a reduced cost. The present implementation allows for free energy-based high-throughput screenings of ligands to a given receptor, and can also be applied for parameter testing and optimization. The program, along with a tutorial and input files for various protein systems, can be found on the link https://github.com/GHeinzelmann/BAT.py. For absolute binding free energy calculations on guest-host systems, check BAT's brother program, the Guest HOst Affinity Tool (GHOAT.py), on https://github.com/GHeinzelmann/GHOAT.py.

In this user guide we will first describe the workflow of the program, then the various components of the free energy calculation, and how the simulations are analyzed in order to obtain the quantities of interest. All the parameters needed for the program input file, and how they apply to the various calculation steps, will also be described in detail. Finally, we will explain how to add a new protein system to our automated protocol.

2. Workflow



Flowchart: Workflow of the BAT.py procedure, starting either from a docked pose or a complex cocrystal structure.

The flowchart above shows a sequence of equilibration and free energy calculation steps for a given ligand pose. It begins by first building the initial complex, either starting from the docked receptor and ligand files or directly from the receptor-ligand cocrystal structure. The necessary ligand parameters are obtained using Antechamber [3], with the General Amber Force-Field (GAFF or GAFF2) for the bonded and LJ parameters [4], and the AM1-BCC model for the partial atomic charges [5,6]. BAT.py can also use existing ligand parameters rather than generating them, as explained in section 6.3. The system is then aligned to a reference structure of a similar protein using the program lovoalign [7], and the ligand anchors are automatically assigned (see section 4.3).

With the aligned system coordinates and the anchors determined, the complex is then placed in a water box with a given ion concentration, and the necessary restraints are applied. An initial equilibration is then performed, in which the translational/rotational restraints of the ligand relative to the receptor are gradually released in order to find a nearby energy minimum. At the end of this last step, the ligand might still be bound or it might have left the binding site in the case of unstable binding mode. If the latter happens, this pose is considered unstable and no further simulations are performed for this system.

If the ligand is still bound at the final state of the equilibration process, this state will be the starting point of the subsequent binding free energy calculations. Here the equilibrated system is realigned, new ligand anchors are defined, as well as a new set of reference values for the restraints. The calculated binding free energy will be the sum of components related to either the application/removal of restraints, or the free energy of transferring the restrained ligand from the binding site to the bulk solvent. Once the free energy simulations are concluded, they can be analyzed using the Multistate Bennett Acceptance Ratio (MBAR) [8] method, Thermodynamic Integration with Gaussian Quadrature (TI), or analytically, depending on the free energy component and additional choices in the BAT.py input file.

3. Free Energy Components

The BAT.py expression for the calculated binding free energy is defined as follows:

$$-\Delta G^{o}_{bind} = \Delta G_{p,att} + \Delta G_{l,conf,att} + \Delta G_{l,TR,att} + \Delta G_{trans} + \Delta G_{l,TR,rel} + \Delta G_{l,conf,rel} + \Delta G_{p,rel} \quad (1)$$

In the equation above, the *att* index denotes attachment of restraints in the bound state, and *rel* indicates release of restraints with the ligand in bulk. The *l* and *p* indexes are for ligand and protein (receptor), respectively, *conf* is for conformational restraints and *TR* is for translational/rotational restraints. The ΔG_{trans} term is the free energy of transferring the ligand from the receptor binding site to bulk with all restraints applied, using either the double decoupling method (DD) of the simultaneous decoupling and recoupling (SDR) procedure.

For the double decoupling procedure, the transfer free energy $\Delta G_{trans\text{-}DD}$ is equal to the sum of four terms, as shown in the equation below. The index *dcpl* stands for decoupling, *elec* for electrostatic interactions and *LJ* for Lennard-Jones interactions, with these calculations being performed with the ligand in the binding site (*bound)* and in bulk (*unbound*). Each term is computed separately, with the bulk calculations being performed with the ligand in a separate box.

$$\Delta G_{trans-DD} = \Delta G_{dcpl,elec,bound} + \Delta G_{dcpl,LJ,bound} - \Delta G_{dcpl,elec,unbound} - \Delta G_{dcpl,LJ,unbound} \quad (2)$$

The SDR method uses essentially the same transformations as the DD method, but here the ligand is decoupled from the receptor binding site and recoupled back in bulk solvent (far from the

receptor) simultaneously and in the same system [9]. This approach overcomes one of the greatest limitations of the double decoupling method, which is dealing with ligands that have a net charge. The $\Delta G_{trans\text{-}SDR}$ transfer free energy is then the sum of two terms:

$$\Delta G_{trans-SDR} = \Delta G_{elec} + \Delta G_{LJ} \quad , \tag{3}$$

with:

$$\Delta G_{elec} = \Delta G_{dcpl,elec,bound} + G_{rcpl,elec,unbound} \tag{4}$$

$$\Delta G_{LJ} = \Delta G_{dcpl,LJ,bound} + \Delta G_{rcpl,LJ,unbound} \quad , \tag{5}$$

where *rcpl* stands for recoupling, which is just the inverse transformation of the decoupling process.

Table I summarizes all the free energy components from our simulations, with each identified by a letter. The **m** and **n** components are called the merged restraint components, with the former applying all restraints using a single set of windows, and the latter doing the same for the release of the conformational restraints, since the TR restraints are always released analytically. The **n** component uses a system that has the ligand and the empty receptor separated and non-interacting, the same way as the SDR box.

When the free energy calculations are set up, the windows from each free energy component will be in folders named according to their corresponding letter followed by the window number, starting at 0. The number of windows and their properties can be defined in the input file. The letters also identify the free energy output files, which are stored in the ./data folder of each component, after the analysis is performed.

4. Restraint setup

As shown in Equation 1, the applied restraints can either be conformational (*conf*), meaning that they are applied to atoms belonging to the same molecule, or translational/rotational (*TR*), which are restraints on the ligand relative to the protein. The restraint setup here follows essentially the same definitions from Ref. [9], with the main difference being that the ligand TR restraints in the BAT 2.x version are not relative to the three dummy atoms, but to the three protein anchor atoms. Also, there are no restraints on the protein relative to the dummy atoms and, finally, the distance restraints between the ligand anchors are no longer applied. The calculation of the attaching/releasing free energy contributions also follows the reference above, and are listed in Table I.

Table I: Binding free energy components, with the associated system, free energy contribution and calculation method.

Description	Letter		System	Free energy terms	Free Energy Method
Attachment of receptor conformational restraints	**a**	**m**	Complex	$\Delta G_{p,att}$	MBAR
Attachment of ligand conformational restraints	**l**			$\Delta G_{l,conf,att}$	
Attachment of ligand TR restraints	**t**			$\Delta G_{l,TR,att}$	
Decoupling of ligand charge interactions (binding site)	**e**		Complex	$\Delta G_{dcpl,elec,bound}$	MBAR/TI
Simultaneous dec/recoupling of ligand charge interactions			Complex + bulk ligand	ΔG_{elec}	
Decoupling of ligand LJ interactions (binding site)	**v**		Complex	$\Delta G_{dcpl,LJ,bound}$	
Simultaneous dec/recoupling of ligand LJ interactions			Complex + bulk ligand	ΔG_{LJ}	
Decoupling of ligand charge interactions (bulk)	**f**		Ligand only	$\Delta G_{dcpl,elec,unbound}$	
Decoupling of ligand LJ interactions (bulk)	**w**			$\Delta G_{dcpl,LJ,unbound}$	
Release of ligand TR restraints	**b**		Ligand only	$\Delta G_{l,TR,rel}$	Analytical
Release of ligand conformational restraints	**c**	**n**	Ligand only†	$\Delta G_{l,conf,rel}$	MBAR
Release of receptor conformational restraints	**r**		Receptor only†	$\Delta G_{p,rel}$	

† For the **n** component, the receptor and ligand are in the same box, but separated and not interacting.

4.1 New ligand TR restraints

BAT 2.x uses rigid-body TR restraints on the ligand relative to the protein, rather than restraining both of them relative to fixed dummy atoms, as done in the previous 1.0 version. Thus, dummy atoms are no longer necessary, only three anchor atoms in the ligand (L1, L2 and L3) and three in the protein (P1, P2 and P3). These rigid-body restraints comprise one distance, two angles and three dihedrals formed between the anchors, which restrain the six degrees of freedom of the ligand relative to the receptor.

The analytical expression for their release in bulk is written in the equation below, with the distance, angle and dihedral variables identified in the left of Figure 1.

$$\Delta G_{l,TR,rel} = k_B T \ln\left(\frac{C^o}{8\pi^2}\right) + k_B T \ln \int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} \exp[-\beta(u_r + u_\theta + u_\phi)] r^2 dr \sin\theta \, d\theta \, d\phi$$

$$+ k_B T \ln \int_0^{2\pi}\int_0^{2\pi}\int_0^{\pi} \exp[-\beta(u_\Theta + u_\Phi + u_\Psi)] \sin\Theta \, d\Theta \, d\Phi \, d\Psi \qquad (6)$$

Here C° is the standard concentration, 1 M = 1/1661Å^3. The u terms are the potential energies from the harmonic restraints, defined as $u = k\,(x - x_0)^2$, with x being a given coordinate with its reference value x_0, and k the spring constant. The ½ term is omitted following the AMBER definition of harmonic restraints between single atoms.

4.2 Center of mass restraints

In order to avoid problems related to the finite size and periodicity of the box, center of mass (COM) restraints are applied to all protein backbone atoms when the ligand is bound to the protein. This includes the equilibrium simulations and the free energy components **a**, **l**, **t**, **e** and **v**. In the case of the **e** and **v** components when SDR is being used, center of mass restraints are also applied to the copy of the ligand located in bulk solvent (right of Figure 1). This keeps the complex and the bulk ligand separated in the box, so that they do not interact, or interact negligibly, and the SDR method remains valid. Center of mass restraints have the advantage of not affecting the internal degrees of freedom of the two molecules during equilibration and free energy calculations. Note that the COM restraints still allow rotation of the protein and bulk ligand around their centers of mass, which does not bring problems as long as they remain separated during the SDR windows.

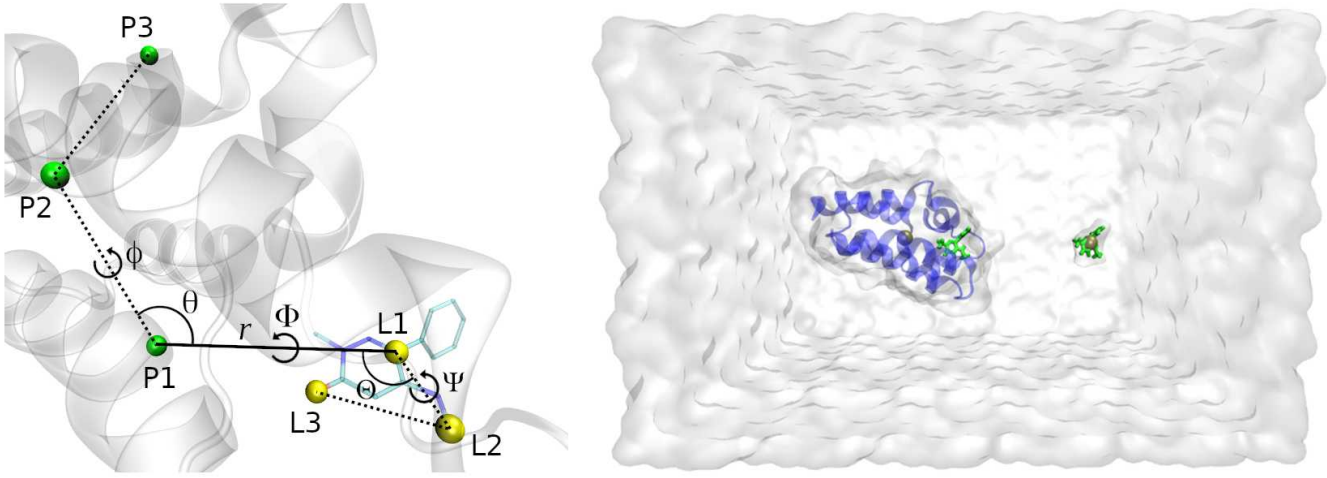


Figure 1: (*left*) The TR restraints on the ligand relative to the protein, showing the protein anchors in green, the ligand anchors in yellow, and the six restrained degrees of freedom: the distance r, the θ and Θ angles, and the ϕ, Φ and Ψ dihedrals. (*right*) The SDR box, with the protein in blue, the ligand in green, and the centers of mass of the protein backbone and bulk ligand as the golden spheres.

4.3 Protein alignment and ligand anchor atom assignment

The protein alignment and the determination of the ligand anchors are performed in the beginning of the equilibration and free energy steps, as explained in section 2. The alignment is necessary since the location of the binding site is defined relative to the P1 anchor, which is one of the input parameters of BAT.py for a new system, along with the P2 and P3 protein anchors. Thus, they must be chosen before running BAT.py, as explained further in section 7.2. Also defined beforehand are the three Cartesian components that define a distance vector **r**, between P1 and a point at the center of a spherical search range for the first ligand anchor L1, as well as a search radius. Figure 2 shows the BRD4(2) bromodomain bound to the 89J ligand, after alignment, displaying all the anchors and the L1 search range.

The determination of the ligand anchors L1, L2 and L3 starts after the protein alignment, with the L1 anchor being the ligand atom closest to the center of the search range. It is possible that during the equilibration simulations the ligand has left the binding site, so that, at the start of the free energy step, no ligand atoms are encountered inside the search region. If that is the case, the ligand is considered to have left the binding site, the starting docked pose is considered to be unstable, and no free energy calculation is performed (see flowchart). If L1 is found, the L2 anchor will be selected as the atom in which the P1-L1-L2 angle is closest to 90 degrees, and the L1-L2 distance is within a specified range, defined in the BAT.py input file. The choice of L3 follows the same procedure, but now using the L2-L3 distance and the L1-L2-L3 angle. Keeping the angles close to 90 degrees, along with a minimum distance, is done in order to minimize the forces applied to the restrained Θ, Φ and Ψ angles/dihedrals from Fig. 1. Large forces can create instabilities on the MD simulations, even more so when using a 4.0 fs time step, so it is a good practice to avoid them when possible.

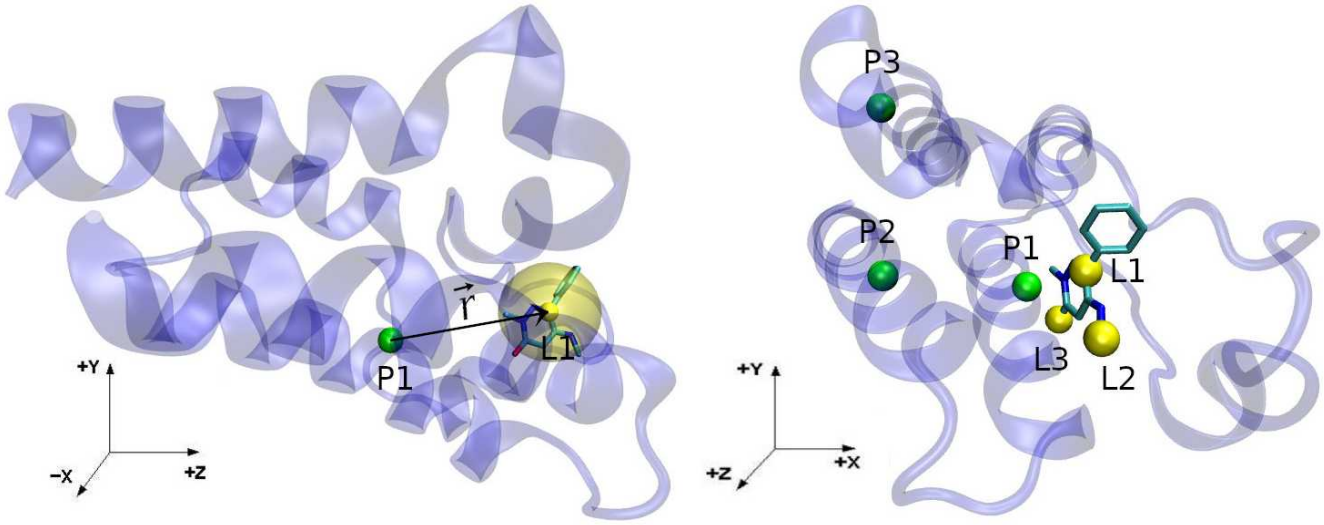


Figure 2: (*left*) The P1 and L1 anchors, as the green and yellow solid spheres, and the **r** vector connecting the two. The transparent yellow region shows the L1 anchor spherical search region. (*right*) All the protein and ligand anchors, as in the left of Figure 1, but from a different perspective.

5. Input file

The various options concerning the creation of the systems, simulations and analysis, can be chosen in the input file using the variables listed below. All variables that involve distances are defined in angstroms (Å).

5.1 Variables for use with AMBER20 *pmemd.cuda*

`calc_type:` Accepts the options “dock” or “crystal”, for a receptor ligand pair of pdb files, or a complex co-crystal structure, respectively. The system initial pdb files should be located in the ./all-poses folder.

`celpp_receptor:` Sets the name of the docked receptor, which should be followed by the “_docked.pdb” string. For example, choose “LMCSS-5uf0_5uez” for a receptor file called LMCSS-5uf0_5uez_docked.pdb. The naming is based on the CELPP challenge procedure. For a crystal structure, put the four letter identifier for the structure, for example “5uf0” for the 5uf0.pdb crystal structure.

`poses_list:` The list of poses that will be used for the calculations. The list should be placed in brackets, separated by commas and without spaces. Ex: “[0,1,2,3,4]”. The docked poses files in the ./all-poses folder must be listed accordingly as pose0.pdb, pose1.pdb, pose2.pdb, etc. This parameter is not used for crystal structure calculations.

`ligand_name:` The ligand residue name in the docked poses above, if `calc_type` is set to “dock” (Ex: “LIG”), or in the initial crystal structure, if `calc_type` is set to “crystal” (Ex: “89J” for the 5uf0 ligand.).

`P1, P2` and `P3:` These define the anchor atoms of the receptor, which have to be determined beforehand. They use AMBER masks to define each atom. Ex: “:425@CA” for the CA atom of residue 425. The original protein sequence numbering can be used for a single protein chain, and for multiple chains see section 7.3.

`fe_type`: Type of binding free energy calculation. If double decoupling with restraints will be performed, choose “dd-rest”. For the merged restraint components with SDR, choose “express”. For only the DD components without computing the free energy of attaching/releasing restraints, choose “dd”. The same goes for “sdr-rest” or “sdr”, when using the SDR method. One can also choose the option “custom”, for a chosen set of components.

`components`: If the option “custom” is set above, choose the components you want to calculate, using a list of letters separated by spaces inside a bracket. Ex: “[c l e v]”. When using the custom option with the **e** and **v** components, set the `dec_method` variable to either “dd” or “sdr”. This is necessary to either build the system with only the complex (DD), or with an additional ligand in bulk (SDR), as shown in Table 1.

`dec_method:` Choice between the DD and SDR methods to build the **e** and **v** components, choosing “dd” or “sdr” for this variable. Only needed when the “custom” option is chosen for the `components` variable, in other cases it will follow the `fe_type` choice.

`sdr_dist:`Distance (in Å) between the bound ligand and the copy of the ligand located in bulk solvent (measured along the *z* axis), as required for the SDR method and the **n** component from Table I. The value of this variable should be large enough that the interactions of the receptor with the bulk copy of the ligand are negligible.

`release_eq`: The weights for the gradual release of the restraints in the equilibrium stage, going

from 100 (fully restrained) to 0 (unrestrained). Each option will be a new simulation, and they are performed in sequence. Use a list of letters separated by spaces inside a bracket to define these weights. Ex: "[5.00 2.50 1.00 0.00]". A single 0.00 inside the brackets (Ex: "[0.00]") will run just one equilibrium simulation without any ligand restraints.

`attach_rest`: List of weights for the spring constant of each window during the attaching/releasing of restraints using MBAR (components **a**, **l**, **t**, **c** and **r**). The total number of windows for each of these components will be the size of the array. Ex: "[0.00 2.00 4.00 16.00 64.00 100.00]" for a total of 6 windows.

`lambdas`: Lambda values for the decoupling procedure in DD and SDR, going from 0.00 to 1.00. Ex: For a 12-point Gaussian quadrature, choose "[0.00922 0.04794 0.11505 0.20634 0.31608 0.43738 0.56262 0.68392 0.79366 0.88495 0.95206 0.99078]" for the lambda array values.

`rec_dihcf_force`: Final spring constant for the protein conformational dihedral restraints, if this option is activated (see `rec_bb` variable). The nature of these restraints, and how they are implemented, are explained in Ref. [9]. Use units of kcal/mol.rad².

`rec_discf_force`: Final spring constant for the protein conformational distance restraints between its anchor atoms. Use units of kcal/mol.Å².

`lig_distance_force`: Force constant for the *r* distance (P1-L1) of the *TR* restraints on the ligand relative to the protein, as explained in section 4.1 and shown in Figure 1. Use units of kcal/mol.Å².

`lig_angle_force`: Angle and torsion angle spring constants for the ligand rigid body restraints, identified as θ, ϕ, Θ, Φ and Ψ in Figure 1. Use units of kcal/mol.rad².

`lig_dihcf_force`: Final spring constants for the ligand conformational dihedral restraints [9]. Use units of kcal/mol.rad².

`rec_com_force`: Force constant for the center of mass restraints on the protein, as explained in section 4.2. Use units of kcal/mol.Å². COM forces include the ½ term, omitted when using single atom restraints (section 4.1).

`lig_com_force`: Force constant for the center of mass restraints on the bulk ligand during the SDR procedure, as explained in section 4.2. Use units of kcal/mol.Å².

`water_model`: The water model used in the calculations. Supported options are "TIP3P", "TIP4PEW", "SPCE", "TIP3PF" and "OPC".

`num_waters`: Number of waters used in the simulations of the complex, the SDR box and the *apo* protein.

`buffer_x`, `buffer_y` and `buffer_z`: Options for the water padding in the three Cartesian axes of the system. The `buffer_z` option is mutually exclusive with the `num_waters` option above, since

for a fixed number of waters the *z* padding is a dependent variable, and vice-versa.

`lig_buffer`: Water padding in the three Cartesian axes for the box with only the ligand in it.

`neutralize_only`: Option to add ions only to neutralize the system, or to also include an additional number of ions. Accepts options “yes” or “no”.

`cation` and `anion`: Cation and anion species to be used, accepts all ions supported by the Joung and Cheatham monovalent ion parameters [10]. Ex: “Na+” and “Cl-”.

`ion_conc`: Salt concentration of the chosen ions for all simulation boxes, when additional ions are included after the system neutralization. Use units of mol/L. (Ex. 0.15).

`hmr`: Use hydrogen mass repartitioning [11] or not. Accepts options “yes” and “no”.

`temperature`: Temperature of the simulated systems, in Kelvin (K).

`eq_steps1`: Number of steps for each simulation of the gradual release of restraints, during the equilibration procedure.

`eq_steps2`: Number of steps for the last simulation of the equilibration procedure, in which the ligand is usually chosen to be unrestrained.

`[component]_steps1`: Number of steps of equilibration, for each window of the various components of the free energy calculation, with the component letters shown in Table I. No data is collected during this simulation.

`[component]_steps2:` Number of steps for the production stage of each window of the various components of the free energy calculation, in which data is collected.

`rec_bb`: Choice to use or not protein (receptor) backbone dihedral restraints, accepting “yes” or “no”.

`bb_start` and `bb_end`: If the option above is activated, these variables define the residue range of the protein backbone dihedral restraints. The original protein sequence numbering can be used for a single protein chain, and for multiple chains see section 7.3.

`bb_equil:` Choice to keep the protein backbone restraints fully attached during the ligand equilibration, or leave it unrestrained so it can adapt to the docked ligand. If the latter is chosen, the reference coordinates for the backbone restraints come from the final state of equilibration. Accepts “yes” or “no”.

`l1_x, l1_y` and `l1_z`: distances in the three Cartesian axes between the first protein anchor atom P1 and the center of the search range for the L1 anchor, denoted by the **r** vector (see section 4.3).

`l1_range`: Radius of the L1 search range.

`min_adis` and `max_adis`: Minimum and maximum distance between the ligand anchors.

`dec_int`: Type of integration method for the decoupling/recoupling components of the binding free energy calculation (**e**, **v**, **f** and **w**). If "TI" is chosen, Gaussian quadrature is applied, if "MBAR" is chosen, the latter is used to calculate these components. Remember that the lambda values have to be suitable for either type of integration method.

`weights`: Weights for Gaussian quadrature calculations, in case the TI option is chosen above. These values must correspond to the values in the `lambdas` array, for the procedure to be applied correctly. In the case of a 12-point Gaussian quadrature, write "[0.02359 0.05347 0.08004 0.10158 0.11675 0.12457 0.12457 0.11675 0.10158 0.08004 0.05347 0.02359]" for this variable.

`blocks`: Number of blocks for block data analysis. This separates the simulation data in blocks and provides the results for each, so the temporal variation and convergence of the results can be assessed. This option is also used for the calculation of the uncertainties of each free energy component [9].

`ntpr`, `ntwr`, `ntwe`, `ntwx`, `cut`, `gamma_ln`, `barostat` and `dt`: Options for running the various simulations, such as output frequency, non-bonded cutoff, barostat type, time step, and others. These use the same variables as the ones from the *pmemd.cuda* simulation input file, and their definitions can be found in the AMBER user guide.

`receptor_ff`: Choice of force field for the receptor atoms, such as "protein.ff14SB" or "protein.fb15". Also supports parameters for DNA and RNA, such as "DNA.OL15" and "RNA.OL3".

`ligand_ff`: Choice of force field for the ligand Lennard-Jones and bonded parameters. Accepts either "gaff" or "gaff2". The ligand partial charges are always determined by the AM1-BCC model.

`ligand_ph`: Reference pH for the protonation of the ligand, before generating parameters. Default is 7.0.

`retain_lig_prot`: Choose to keep the ligand protonation, if starting with a docked pose that already has all the ligand hydrogens included. Accepts "yes" or "no", the latter being the default choice. If this option is set to "yes", BAT will try to read the total net charge of the ligand using the element column of the docked pdb file.

`ligand_charge`: Ligand net charge, in case the user wants to manually choose this value when running Antechamber to get the AM1-BCC partial charges. If this variable is not chosen, BAT will use the net charge estimated from the babel protonation if `retain_lig_prot` is set to "no", or read it from the docked pdb file if the `retain_lig_prot` variable is set to "yes" (as explained above).

`other_mol`: List of cobinder molecules that will be used for equilibration and free energy calculations, using their three letter residue code in upper case. The list should be placed in brackets, separated by commas and without spaces. Ex: "[SO4,PO4]" for residues SO4 and PO4.

`solv_shell`: Radius around all protein atoms centers in which the cobinders and water molecules (also measured from atom centers) from the initial model will be included in the system, both in the

equilibration stage as well as in between equilibration and free energy calculation.

5.2 Specific variables for use with OpenMM

When using BAT with the OpenMM software, a few of the AMBER20 BAT variables above are not needed, as shown in the example input files provided in the tutorial. In addition, the OpenMM simulations/calculations require a few specific variables, which are listed below.

`[component]_itera1`: Number of replica exchange equilibration iterations for each window from a given component of the binding free energy calculation. Replaces the `[component]_steps1` variable from the AMBER20 version.

`[component]_itera2`: Number of replica exchange production iterations for each window from a given component of the binding free energy calculation. Replaces the `[component]_steps2` variable from the AMBER20 version.

`itera_steps`: Number of simulation steps per iteration, with the total number of steps per window being this value multiplied by the number of iterations per window above.

`itcheck`: write OpenMM checkpoint file at every `itcheck` iterations.

`software`: Choose “openmm” for this option if using this simulations software.

6. Adding new ligands to a given protein

In the example provided in the ./BAT folder, binding free energy calculations are performed for the 5uf0 crystal structure, as well as 5 docked poses of the same ligand docked to the receptor with the 5uez structure. The protein receptor is the second bromodomain of the BRD4 protein – BRD4(2). The ./all-poses folder in this example contains the original 5uf0.pdb file, as well as a set of pdb files for the docked poses and receptor. The same procedure can be applied to any ligand that binds to BRD4(2), as explained below:

6.1 Docked complexes

In order to generate a set of pdb files for the docked poses and receptor, there are two options, either do a manual docking with chosen input files, or using the CELPP challenge workflow. Both options use AutoDockTools [12], Chimera [13], and AutoDock Vina [14] to prepare the files and run the docking, so you must have them in your path in order to perform this stage.

For the first option, the ./docking-files/Vina-example folder has a simple docking workflow using the *dock.bash* script and the input files for the 5uf0 system, which already outputs the files in the right format for use with BAT. This script can be easily modified for other protein-ligand systems to be used with BAT.

The docking can also be integrated into the CELPP challenge, using the procedure from the CELPPade tutorial, which can be performed following the instructions in the link https://doc.google.com/document/d/1iJcPUktbdrRftAA8cuVa32Ri1TPr2XvZVqTccDja2OMh. The same scripts from the “internal_autodockvina_contestant” model from this tutorial can be used for docking, except for *internal_autodockvina_contestant_dock.py*. To output the docked structures suitable for use with BAT, you should use instead the *BAT_dock.py* script, which is included inside

the ./docking-files/CELPP-docking folder. Once you run the CELPP docking, the necessary pdb structures for various different receptors will be inside the ./4-docking folder that was just created with the docking results.

6.2 Co-crystal structure

In the case of a co-crystal structure, the `calc_type`, `ligand_name` and `celpp_receptor` options in the input file have to be adjusted properly, as explained in section 5. The script *prep-crystal.tcl*, inside the ./BAT/build_files folder, is used by VMD [15] to "clean" the original file and leave only a single chain of the receptor-ligand complex. The editing of this tcl script is usually not needed, but is necessary if the original pdb file has more than one chain. Keep in mind that the "MMM" identifier will be replaced by the ligand name, so it does not have to be changed beforehand.

6.3 Including your own ligand parameters

BAT.py also accepts existing ligand parameters, rather than generating them using AM1-BCC/GAFF, as long as they are compatible with AMBER/tleap. In that case, before running the equilibration step, an ./equil/ff/ folder should be created inside the program main folder (./BAT), and the necessary parameter files should be included in it. They can be either a .mol2 file with the partial charges, an .frcmod file with the bonded/LJ parameters, or both, and should be named according to the ligand three letter residue code in the pdb structure of the complex. For example, for the ligand called 89J from the 5uf0 crystal structure, the parameter names should be 89j.mol2 and 89j.frcmod, and the residue name in them should be 89J. That way, BAT will skip the parameter generation step and use the provided files instead.

7. Adding a new protein system

BAT.py can be extended to several protein systems, by including a reference alignment file and adjusting a few parameters in the input.in file. The ./systems-library folder included in the distribution already contains the necessary data for a few proteins, and below we show how add a new system to the BAT workflow.

7.1 Creating protein reference structure for alignment

The first step is to create the reference.pdb file, so that the complex can be aligned relative to it using lovoalign, before the BAT.py simulations are performed. Since the bound ligand does not need clear access to the solvent, as in the APR method, there are no restrictions on how the reference protein file should be oriented in space. However, for more elongated proteins, it is advisable to keep its longer side aligned to the *z* direction, since the separation between the bound ligand and the bulk ligand in the SDR process is done along this coordinate (Figures 1 and 2). That way, the rotation of the bound complex during the SDR calculations will not increase its proximity to the ligand located in bulk.

The reference file is created by rotating a structure of the desired protein with a ligand bound, in this example the 5uf0 crystal structure of BRD4(2), and then saving it in the ./BAT/build_files folder as reference.pdb. One way of doing this with VMD is applying a few useful tcl commands (http://www.ks.uiuc.edu/Training/Tutorials/vmd/vmd-tutorial.pdf), but other programs such as Chimera can also be employed for that purpose. The reference.pdb file does not need to have the same sequence as the receptor input file, so the same reference created here for for BRD4(2) can be extended to other

bromodomain homologs that share the same fold, such as BRD4(1), CREBBP and BAZ2B.

7.2 Determining the BAT.py input variables

Starting from the crystal structure above, already in the orientation chosen for the reference.pdb file, the remaining BAT input variables can be set. We will start with the distance between the bound and bulk ligand during the SDR calculations. Using a program such as VMD, move the ligand along the *z* coordinate towards +*z* until the latter is far enough from the protein that their interactions are negligible (right of Figure 1). Car-Parrinello Molecular Dynamics (CPMD) and classical MD calculations on ion pairs show that, at room temperature, this is generally the case for distances over 10 Å of pure solvent [16]. Thus, this value can be used as the minimum width of the solvent layer between the protein and the bulk ligand. Once the position of the bulk ligand is chosen, the *z* distance between the latter and the initial position of the bound ligand (in Å) will be the value used for the `sdr_dist` input parameter.

For the choice of the protein anchors P1, P2 and P3, an estimate of the location of the L1 anchor atom is necessary, which can be done using the aligned 5uf0 complex file above. Choose for the tentative L1 an atom close to the center of the binding site, since this position will be the center of the L1 search range, as shown in Figure 2. Once that is done, the P1 protein anchor can be determined following a few guidelines:

1 – It should be a backbone atom (CA, C or N) and part of stable secondary structure such as an alpha-helix or a beta sheet, always avoiding loop regions due to their increased flexibility.

2 – The distance vector ***r*** *between P1 and L1 (left of Fig. 2) should have an absolute value between 7 Å and 12 Å. Smaller values can create a small lever arm depending on the ligand; larger values might result in weak TR restraints, due to the* r^2 *dependence of the spherical surface sampled by the restrained ligand.*

3 - The distance vector ***r*** *does not have to be in a particular direction, even though it is (personally) preferable that it is nearly aligned with the +z direction.*

Once P1 has been selected, the P2 and P3 anchors can be chosen, also following a few rules:

1 – Should also be backbone atoms and part of stable secondary structures such as alpha-helix or beta sheet, again always avoiding loop regions.

2 – The P2-P1-L1 and P1-P2-P3 angles should not be close to 0 or 180 degrees, and the closer they are to 90^o *the better. The P1-P2 and P2-P3 distances should be at least 8 Å, to avoid large forces during the simulations due to a small lever arm on the torsional restraints.*

Once the protein anchors have been selected, the BAT variables `P1, P2` and `P3` are used to define them in the BAT input file using the AMBER mask format, as explained in section 5.

The x, y and z components of the **r** distance between P1 and L1 above will provide the values of the `l1_x, l1_y` and `l1_z` variables, which locate the center of the L1 search region relative to P1. The search radius is chosen using the `l1_range` parameter, in a way that the search sphere covers most of the binding site. Finally, the `min_adis` and `max_adis` variables can be set, which are the minimum and maximum distances between L1 and L2, and between L2 and L3. Here, safe values of 3.00 Å and 7.00 Å for the respective minimum and maximum distances can be used, but these can

decreased for molecules that are too small, such as a single ring, or increased for larger ligands, such as molecules with more than 100 atoms.

7.3 Proteins with multiple chains

From the 2.2 version, BAT supports the use of protein receptors that have multiple chains, identical or not, as well as the inclusion of molecules from the initial model besides the receptor and the ligand, the latter explained in the next section.

For BAT to automatically recognize the different protein chains and correctly assign the protein anchor atoms, there are a couple of rules for the initial docked/crystal pdb file, as well as for the anchor atom identification. Regarding the initial receptor file, all atoms from a given chain should be identified by its chain one letter code, such as A, B, C and so on, which comes right next to the residue name in the pdb file. For example in the lines below:

```
ATOM   1748  HA3 GLY A 114      34.212  27.603  25.500  1.00 11.25           H
TER    1749      GLY A 114
ATOM   1750  N   ALA B 115      20.463  47.395  35.918  1.00 13.88           N
```

, the A and B chain identifiers are highlighted in red. Since this is the standard pdb format, most of the times no changes are needed from the initial structure. The TER line in the middle is not needed, so it is optional.

The protein anchors chosen according to section 7.2 can belong to different protein chains, but they must be numbered according to the sequence in which the residues appear in the initial pdb file. For example, if the first chain (A) has a total of 150 residues and one wants to choose for the first anchor the CA atom from the 20th residue from the second chain (B), the variable `P1` should be defined as 170@CA, and the same rule goes for `P2` and `P3`. This numbering rule also applies to the protein backbone restraints in a given residue range, defined by the `rec_bb`, `bb_start` and `bb_end` variables. It is common for pdb files not to start the residue numbering at 1, or to restart it when a new protein chain begins, or both. This can be addressed by running the program pdb4amber (which should be in your path when running BAT.py) in your initial receptor file:

```
pdb4amber -i receptor_init.pdb -o receptor_renum_docked.pdb
```

Running the command above will renumber all protein residues from the initial receptor file (receptor_init.pdb), starting from 1 and in sequence regardless of the chain, so that the output file (receptor_renum_docked.pdb) can be used when choosing the protein anchors. This output receptor file should then also be used as the BAT.py input (`celpp_receptor` variable), so that the assignment of the protein anchors and backbone restraints is consistent.

7.4 Water molecules and cobinders

It is possible to include in the calculation molecules present in the initial model besides the receptor and the ligand, such as water molecules and cobinders such as ions. These molecules should be present in your receptor file when using the “dock” option for `calc_type`, or in the cocrystal structure if using the “crystal” option.

The variable `other_mol` will identify the chosen cobinders (not including water) as an array of residues names, as explained in section 5.1. Simulation parameters for these molecules should be included in the BAT/build_files/ folder in .mol2 and .frcmod formats, named in lower case according to

the residue name. For example, for a cobinder molecule with the residue name SO4, the respective so4.mol2 and so4.frcmod files should be added to this folder. Instructions on how to generate these two files can be found here https://ambermd.org/tutorials/basic/tutorial4b/index.php. The three letter residue codes inside the `other_mol` array should *never* be the same as the one for the ligand that will be decoupled, chosen with the `ligand_name` variable. If the user wants to keep in the system one or more copies of the ligand that are not included in the binding free energy calculation, they should give them a different residue name to avoid confusion.

Only cobinders within the `solv_shell` distance from the protein (see section 5.1) will be included in the system, together with all water molecules that are located within this range. This is true both for the equilibration stage, which starts from the initial docked structure, as well as the free energy stage, which starts from the equilibrated system. Thus, even if the initial docked model does not have water molecules in it, the `solv_shell` variable allows the user to keep the water molecules close to the protein when the system is rebuilt between the equilibration and free energy stages.

8. References

[1] D.A. Case, K. Belfon, I.Y. Ben-Shalom, S.R. Brozell, D.S. Cerutti, T.E. Cheatham, III, V.W.D. Cruzeiro, T.A. Darden, R.E. Duke, G. Giambasu, M.K. Gilson, H. Gohlke, A.W. Goetz, R. Harris, S. Izadi, S.A. Izmailov, K. Kasavajhala, A. Kovalenko, R. Krasny, T. Kurtzman, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, V. Man, K.M. Merz, Y. Miao, O. Mikhailovskii, G. Monard, H. Nguyen, A. Onufriev, F.Pan, S. Pantano, R. Qi, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C. Simmerling, N.R.Skrynnikov, J. Smith, J. Swails, R.C. Walker, J. Wang, L. Wilson, R.M. Wolf, X. Wu, Y. Xiong, Y. Xue, D.M. York and P.A. Kollman (2020), AMBER 2020, University of California, San Francisco.

[2] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande (2017). “OpenMM 7: Rapid development of high performance algorithms for molecular dynamics.” *PLOS Computational Biology*, **13**, e1005659.

[3] J. Wang, W. Wang, P.A. Kollman, and D.A. Case.. (2006) "Automatic atom type and bond type perception in molecular mechanical calculations".*Journal of Molecular Graphics and Modelling*, **25**, 247-260.

[4] J. Wang, R.M. Wolf, J.W. Caldwell, and P. A. Kollman, D. A. Case (2004) "Development and testing of a general AMBER force field". *Journal of Computational Chemistry*, **25**, 1157-1174.

[5] A. Jakalian, B. L. Bush, D. B. Jack, and C.I. Bayly (2000) "Fast, efficient generation of high-quality atomic charges. AM1-BCC model: I. Method". *Journal of Computational Chemistry*, **21**, 132-146.

[6] A. Jakalian, D. B. Jack, and C.I. Bayly (2002) "Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. Parameterization and validation". *Journal of Computational Chemistry*, **16**, 1623-1641.

[7] L. Martínez, R. Andreani, J. M. Martínez (2007) “Convergent algorithms for protein structural alignment.” *BMC Bioinformatics* **8**, 306.

[8] M. R. Shirts and J. Chodera (2008) “Statistically optimal analysis of samples from multiple equilibrium states.” *Journal of Chemical Physics,* **129**, 129105.

[9] G. Heinzelmann and M. K. Gilson (2021). “Automation of absolute protein-ligand binding free energy calculations for docking refinement and compound evaluation”. *Scientific Reports,* **11**, 1116.

[10] I. S. Joung and T. E. Cheatham III (2008). “Determination of Alkali and Halide Monovalent Ion Parameters for Use in Explicitly Solvated Biomolecular Simulations”. *The Journal of Physical Chemistry B,* **112**, 9020-9041.

[11] C. W. Hopkins, S. Le Grand, R. C. Walker, and A. E. Roitberg. (2015) “Long-Time-Step Molecular Dynamics through Hydrogen Mass Repartitioning”. *Journal of Chemical Theory and Computation*, **11**, 1864-1874.

[12] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson. (2009) “AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility” *Journal of Computational Chemistry*, **30**, 2785-2791.

[13] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin. (2004). "UCSF Chimera - A Visualization System for Exploratory Research and Analysis." *Journal of Computational Chemistry*, **25,** 1605-1612.

[14] O. Trott and A. J. Olson. (2010) "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading," *Journal of Computational Chemistry*, **31**, 455-461.

[15] W. Humphrey, A. Dalke and K. Schulten. (1996) "VMD - Visual Molecular Dynamics", *Journal of Molecular Graphics*, **14**, 33-38.

[16] J. Timko, D. Bucher and S. Kuyucak. (2010) “Dissociation of NaCl in water from *ab initio* molecular dynamics simulations”. *Journal of Chemical Physics* **132**, 114510.