Received: 7 November 2021 | Revised: 17 February 2022 | Accepted: 11 April 2022

DOI: 10.1002/jcc.26877

RESEARCH ARTICLE

Journal of COMPUTATIONAL CHEMISTRY WILEY

Alchemical free energy simulations without speed limits. A generic framework to calculate free energy differences independent of the underlying molecular dynamics program

Marcus Wieder[1] | Markus Fleck[2] | Benedict Braunsfeld[2] | Stefan Boresch[2]

[1]Department of Pharmaceutical Sciences, Faculty of Life Sciences, University of Vienna, Vienna, Austria

[2]Department of Computational Biological Chemistry, Faculty of Chemistry, University of Vienna, Vienna, Austria

Correspondence
Marcus Wieder, Department of Pharmaceutical Sciences, Faculty of Life Sciences, University of Vienna, Josef-Holaubek-Platz 2 (UZA II), A-1090, Vienna, Austria.
Email: marcus.wieder@univie.ac.at

Stefan Boresch, Department of Computational Biological Chemistry, Faculty of Chemistry, University of Vienna, Währingerstraße 17, A-1090, Vienna, Austria.
Email: stefan@mdy.univie.ac.at

Funding information
Austrian Science Foundation, Grant/Award Number: P-31024; FWF Erwin Schrödinger Postdoctoral Fellowship, Grant/Award Number: J 4245-N28

Abstract

We describe the theory of the so-called common-core/serial-atom-insertion (CC/SAI) approach to compute alchemical free energy differences and its practical implementation in a Python package called Transformato. CC/SAI is not tied to a specific biomolecular simulation program and does not rely on special purpose code for alchemical transformations. To calculate the alchemical free energy difference between several small molecules, the physical end-states are mutated into a suitable common core. Since this only requires turning off interactions, the setup of intermediate states is straightforward to automate. Transformato currently supports CHARMM and OpenMM as back ends to carry out the necessary molecular dynamics simulations, as well as post-processing calculations. We validate the method by computing a series of relative solvation free energy differences.

KEYWORDS
free energy calculations, solvation free energy, toolkit

1 | INTRODUCTION

The free energy difference between two states determines their relative stability; applied to a (bio)chemical reaction it determines the direction in which the reaction will take place voluntarily. While the above ignores complications from kinetic effects, such as reaction barriers, the capability to compute free energy difference between reactants and products permits the prediction of equilibria of chemically and biochemically relevant processes. Great efforts have been and are being exerted to compute free energy differences of, for example, binding, solvation, and partitioning reliably and reproducibly. The tool of choice to compute these quantities are so-called free energy simulations (FES), which are rapidly becoming a standard tool in computational chemistry.[1–5]

The ever-increasing speed of hardware, in particular the raw computational power of consumer graphics cards, combined with algorithmic progress today make it possible to carry out molecular dynamics (MD) simulations of biomolecular systems, which 15 years ago would have been possible only on the world's most powerful computers. This development has increased the usage of MD simulations as a standard tool even by non-experts.[6] FES, even though in most cases MD based, have until recently not profited to the same degree from the increase in computational speed. First, FES incur some principal overhead; for example, in many programs the reciprocal space energies and forces of the (particle-mesh) Ewald sum need to be computed twice, once for the initial, once for the final state at each step of a FES. Further, to compute free energy differences between states, various tricks are needed which require specialized computer code/routines. These

capabilities were or are often not available in the fastest code paths of many widely used MD programs. Therefore, MD simulations used to compute (alchemical) free energy differences often are slower than "plain" simulations of the same system. An overview concerning the computational overhead of FES can be found in Ref. 7 Clearly, it would be highly desirable to run the sampling phase of FES at a similar speed as regular MD simulations. Progress is being made and GPU support for alchemical FES is becoming available.[4,8]

Another obstacle to the more widespread use of FES is setting up the transformation between two states, that is, how to change state A into state B. Tools have been developed to aid with this step; an early example is the (now defunct) FESetup web server,[9] which handled the details to set up alchemical transformations for several simulation programs, such as AMBER,[10] CHARMM,[11] GROMACS,[12] or NAMD.[13] A principal problem, however, remains: each program that supports alchemical FES has its own internal approach to how transformations are set up; for example, is the transformation accomplished by a single vs. dual topology approach,[14] or—in case of single topology—is the mixing done on the level of parameters or energies/forces.[7] Each of these approaches has different strengths and weaknesses. Thus, a particular transformation may be easy to set up in one program, but difficult to accomplish in another program. In other cases, the opposite may be true. The practical difficulties resulting from this are illustrated in two very recent publications.[15,16] Loeffler et al. compared results for several relative free energy differences of solvation computed with AMBER, CHARMM, GROMACS, and SOMD.[17] While the final, overall agreement was good, the authors stressed that considerable effort was needed to achieve it, and they pointed out several "quirks" of each of the programs considered. Rizzi et al.[16] focused on convergence and reproducibility of binding free energy methodology of multiple programs starting from a single set of parameter files, partial charges, and initial geometries of host-guest systems in the course of the SAMPL6 SAMPLing challenge. They observed differences between converged binding free energy estimates ranging from 0.3 to 1.0 kcal/mol, highlighting the challenges that the field faces when it comes to the transferability of results between different free energy codes.

In this study, we show how the issues just outlined (computational overhead, reproducibility, and a sometimes inflexible corset to set up alchemical transformations) can be circumvented by avoiding dedicated codes to compute free energy differences. When the first GPU accelerated MD codes became available, Boresch and Bruckner presented and tested such an approach, which enabled them to compute alchemical free energy differences using programs without the functionality for this task.[18] The approach was limited to the calculation of absolute solvation free energy differences, and the necessary steps to set up such calculations included manual modifications of parameter and topology files, requiring expert knowledge both of free energy calculations and the inner workings of CHARMM. Despite that, we have continued to use it on occasion[19]; a recent study by Giese and York[7] utilized related ideas. Here we extend the approach of Ref. 18 to the case of, in principle, arbitrary alchemical transformations. We have developed a Python package (Transformato) to set up the intermediate steps leading from a state A to B. The tool generates force field parameters and systems information as needed (for how to obtain the code and data we point to the data availability statement). The underlying MD program is not carrying out any "alchemical FES" specific tasks; that is, it is carrying out a straightforward MD. All quantities needed to compute the free energy differences of interest are obtained using Transformato in post-processing steps from trajectories saved during the MD simulations.

In order to facilitate the task of setting up intermediate states, we adopt the following approach to the computation of a free energy difference between two states A and B. Rather than alchemically transmuting A into B, we search for a suitable common substructure, which is (mostly) identical in the two systems (molecules). We refer to this as the common core (CC); we stress that CC of A (CC_A) and B (CC_B) need not be described by identical force field parameters as long as there is correspondence between the atoms of the CCs (see Section 2 for details). Assuming for the moment the simplest case, that is, $CC_A \equiv CC_B \doteq CC$, this means that the free energy difference $\Delta G(A \rightarrow B)$ is carried out in two steps, $\Delta G(A \rightarrow CC) + \Delta G(CC \rightarrow B)$, where the second step is in practice computed as $-\Delta G(B \rightarrow CC)$. The use of a CC facilitates the setup of the alchemical transformation considerably because this allows us to define the physical end-states without dummy atoms and dummy parameters. Additionally, it is a good match for the serial atom insertion (SAI) approach of Boresch and Bruckner,[18] which is employed here as well. Further, if free energy differences between more than two states are needed, for example, solutes or ligands L_1, L_2, L_3,…L_n and provided a suitable CC exists, then one needs exactly n FES to compute all possible relative free energy differences between the n states.

We test the approach by recomputing all relative solvation free energy differences reported in Ref. 15 To highlight the generality of the approach, we report results carried out with CHARMM and OpenMM as the underlying MD program. The use of OpenMM, a program, or rather a library for MD simulations, illustrates the versatility since the base OpenMM suite has practically no provisions for alchemical FES, which is shipped separately in the OpenMMTools[1] or Perses package[2].

The remainder of the manuscript is organized as follows. In Section 2, we first provide full details of the CC approach. In particular, we demonstrate that contributions from so-called dummy atoms, which typically are present in CC_A and CC_B, and which constitute one difference between the two "common" cores, cancel from parallel legs of the thermodynamic cycles usually employed in applications of alchemical FES. Section 2 concludes with a review of SAI. While the CC concept is crucial to our approach, we use SAI out of practical necessity—SAI could be replaced by soft-core potentials if these are available without impeding computational speed. In Section 3, we first present the benchmark systems of Ref. 15, followed by a detailed description of the simulation details. The presentation of Results (Section 4) is followed by a Concluding Discussion (Section 5).

2 | THEORY

Common cores are used to connect the physical end-states of different molecular typologies. The central and technically challenging step of a

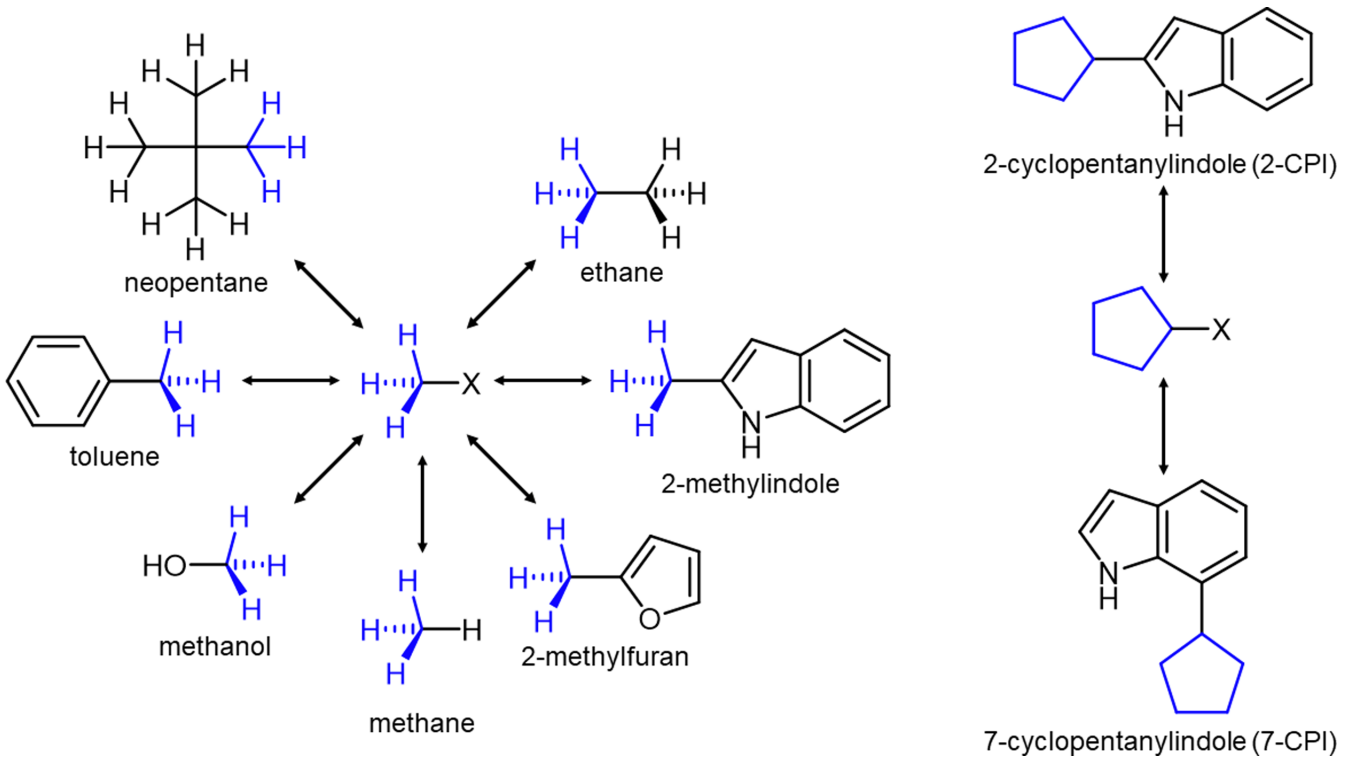

FIGURE 1 Model systems used in relative solvation free energy calculations. Two different common cores were used, a methane-like molecule (CH_3X) for the seven solutes shown on the left, and a modified cyclopentane for the transformation of 2-cyclopentylindole (2-CPI) to 7-cyclopentylindole (7-CPI).

FES is setting up the alchemical transformation between two or more molecules or chemical moieties, for example, the transformation of ethane to methanol. Often, if free energy differences for a series of solutes or ligands need to be computed, there is some common substructure. For example, in one of our model applications, shown on the left of Figure 1, a methyl (CH_3–) moiety is present in all compounds. Let us denote this situation as CC-R_i, i = 1,...N, where CC indicates the common substructure, and R_i the parts in which the molecules differ. To compute the free energy differences between *N* molecules, one has to carry out at least *N* - 1 alchemical transformations,

$$\text{CC-R}_1 \rightarrow \text{CC-R}_i \; (i = 2,\ldots,N). \tag{1}$$

If for example, the CC-R_i corresponds to ethane, neopentane, toluene, and so forth. from our example, then each of the transformations in Equation 1 must be set up individually. This is prone to error if done by hand and challenging to automate. Of course, in such a situation practitioners will choose the alchemical paths/transformations that are easiest to set up; for example, in the study which inspired this model application, Loeffler et al. mutated each of the larger molecules to methane, the smallest compound of the set.

The strategy to always mutate towards the smallest common denominator, that is, going to the smallest compound containing the common substructure (CH_3– in our example) can be generalized and forms the basis of what we refer to as the CC approach. Rather than setting up a transformation CC-$R_A \rightarrow$ CC-R_B in a single step, we break it up into two mutations:

$$\begin{aligned} &\text{CC-R}_A \rightarrow \text{CC-}\mathbf{DR}_A \quad (\Delta G_A) \\ &\text{CC-R}_B \rightarrow \text{CC-}\mathbf{DR}_B \quad (\Delta G_B) \end{aligned} \tag{2}$$

In each of the steps, the functional group R_i is mutated to non-interacting atoms DR_i, commonly referred to as dummy atoms. Provided they are treated correctly, dummy atoms have no influence on the result of double free energy differences, as calculated in the usual thermodynamic cycles.[20] While the partition function and, hence, the free energy of "CC" and "CC-DR_i" are different, the dummy atoms give an additive contribution to the partition function, which cancels from the double free energy differences of interest. From this, it follows that any double free energy difference $\Delta\Delta G(\text{CC-R}_A \rightarrow \text{CC-R}_B)$ can be split into the two steps of Equation 2. Both have to be computed for each leg of the thermodynamic cycle of interest. Specifically,

$$\begin{aligned} \Delta\Delta G(\text{CC-R}_A \rightarrow \text{CC-R}_B) = {} & \Delta\Delta G(\text{CC-R}_B \rightarrow \text{CC-DR}_B) \\ & - \Delta\Delta G(\text{CC-R}_A \rightarrow \text{CC-DR}_A) \end{aligned} \tag{3}$$

We stress that Equation 3 holds even when the number of dummy atoms is different for A and B.

From a practical point of view, each of the alchemical transformations arising in the CC approach consists of mutating one or more atoms to dummy atoms. In contrast to a general alchemical transformation, this is straightforward to set up and, most importantly, easy to automate.

There are cases in which Equation 3 is not sufficient. As mentioned in the Introduction, the CC needs not to be exactly the same in a series of transformations CC-$R_i \rightarrow$ CC-DR_i. Consider the following scenario: element identity (i.e., two nodes of the molecular graph match if they have the same element regardless of atom type or hybridization) is used as the matching criterion for the CC. When starting with several physical molecules CC-R_i, one or more atoms of the CC may be represented by different atom types of the force field, and/or their partial charges may be different. In other words, having transformed the -R_i to the respective dummy groups -DR_i, there may be small differences in the CCs, that is, the endpoints must be written as CC_i-R_i. These CC_i, however, must have the same number of atoms, as well as a one-to-one correspondence between each of the atoms. Permitting such flexibility makes it easier to find/define CCs.

Assuming that we have two such endpoints, CC_A and CC_B, then obviously the free energy difference between them must be accounted for. In principle, this can be done in a separate step/calculation by computing $\Delta\Delta G(CC_A \rightarrow CC_B)$ and adding it to Equation (3). Alternatively, one can add additional alchemical mutation steps after each transformation to CC_i, coercing them into a single CC. While the CC_i need not be identical, they are likely very similar, so the additional transformations do not require many steps. This is the procedure we used in all examples considered in this study; that is, our transformations always follow the pattern CC_i-$R_i \rightarrow CC_i$-$DR_i \rightarrow$ CC-DR_i for $i = A, B$.

If we applied this approach naively to our model application of Figure 1, we would choose the CH_3- moiety as the CC. While theoretically correct, this results in technical difficulties maintaining the dummy group DR in a meaningful position and orientation relative to the CC. In our recent analysis,[20] we classified a dummy atom configuration as in CH_3-DR as a "triple junction." This is the one case where the required decoupling between degrees of freedom of the dummy atom and of the physical atom is difficult to accomplish. By contrast, the easiest to handle case in this respect is the "terminal junction." The triple junction configuration can be avoided by maintaining one atom of the -DR_i group as interacting. The "better" CC, therefore, is CH_3X- as shown in Figure 1. Specifically, the alchemical transformation to mutate, for example, ethane into this CC becomes $CH_3CH_3 \rightarrow CH_3XD_3$. In our example, methane ("CH_3H") would be a valid CC (with the fourth hydrogen being the X), but the ability to choose and adapt the interaction parameters of X offers additional flexibility. For the specific parameters used for X in this work, see Section 3.

"*Serial atom insertion*" *is used to avoid the need for customized soft core potentials*. When creating or annihilating particles in a dense environment, such as solvent, the van der Waals endpoint problem occurs.[21] In FES the standard workaround is the introduction of a soft-core potential.[22–24] The corresponding computer code, however, is often interlaced with the FES code, so, as described in the introduction, GPU support may be poor or missing. Therefore, in the present work, we rely on the so-called SAI method to avoid van der Waals endpoint problems.[18] Instead of scaling Lennard–Jones (LJ) interactions as a function of a continuous coupling parameter, the LJ interactions of an atom (the partial charge of which was switched to zero in a previous step) are either fully interacting or completely turned off. Using Bennett's acceptance ratio method (BAR)[25] or its multi-state extension MBAR,[26] the free energy difference between one or even two LJ interactions being turned on/off can be computed reliably. As described in Ref. 18, SAI is incompatible with thermodynamic integration since unmodified LJ potentials are used and the intermediate states are no longer continuous with respect to the coupling parameter λ. It should be stressed that SAI is not essential to the CC approach; however, since it does not require specialized code as the soft-core potential, the combination of CC and SAI makes it possible, in principle, to set up FES on top of any biomolecular MD program.

3 | METHODS

3.1 | Overview of calculations

Our model/benchmark calculations involve the same nine molecules used by Loeffler et al.[15] Rather than computing specific relative solvation free energy differences (between ethane, methanol, neopentane, toluene, 2-methylfurane and 2-methylindole relative to methane, and between 2-cyclopentanylindole (2-CPI) and 7-cyclopentanylindole (7-CPI) directly as in Loeffler et al.), we inserted a methane-like and a cyclopentane-like CC as shown in Figure 1. All relative FES in the CC/SAI framework were carried out with CHARMM[11] using the domain decomposition implementation for GPUs[3] and OpenMM.[27] Calculations were repeated five times and the average and standard deviation of the individual MBAR free energy estimates are reported.

We re-parameterized the solutes using the CGenFF interface at paramchem.org[28–30]; therefore, the relative solvation free energy differences calculated in this work cannot be directly compared to the results of Loeffler et al.[15] To validate the workflow with an established protocol, we computed the absolute solvation free energies for all nine solutes with the PERT module of CHARMM.[11] The use of PERT introduced a subtle complication because in this module only the original CHARMM switching function for LJ interactions (from now on referred to as "vswitch")[31] but not the LJ force switching function ("vfswitch")[32] is supported. OpenMM, on the other hand, neither has native support for CHARMM's "vswitch" nor for "vfswitch." However, when obtaining input scripts for OpenMM through the CHARMM-GUI server,[33,34] a custom energy routine for "vfswitch" is provided. In fact, many of our inputs both for CHARMM and OpenMM are based on scripts generated by CHARMM-GUI to maintain as much consistency as possible between the two programs. Our testing/validation, therefore, proceeded as follows. First, we compared relative solvation free energy differences obtained in the

CC/SAI framework with CHARMM, truncating LJ interactions with "vswitch", and compared these results to the corresponding differences between absolute solvation free energies obtained with PERT. Then, we compared relative solvation free energies calculated with the CC/SAI framework using CHARMM and OpenMM directly with each other, truncating LJ interactions with "vfswitch."

3.2 | Relative alchemical free energy calculations using Transformato

We have developed a Python package named Transformato that performs the steps required by the CC/SAI approach. Transformato takes care of generating the alchemical path, dispatches sampling and post-processing calculations at the alchemical states, and computes the relative free energy difference between physical states and their CC. At present, Transformato can perform these tasks mostly automatically using either OpenMM or CHARMM for sampling and post-processing.

To calculate the free energy from two physical end-states, the following steps have to be performed:

1. Identify the maximum common substructure using a specific node/vertex matching criteria;
2. Generate the alchemical path connecting the molecules to the same CC;
3. Sample the alchemical states;
4. Use the multi-state Bennett acceptance ratio estimator implementation in pymbar[26] to calculate the free energy difference from the alchemical samples obtained in step (3).

3.2.1 | Identifying the maximum common substructure

Transformato identifies the CC substructure using a customized maximum common substructure algorithm based on the chemoinformatics toolkit RDkit,[35] with added checks to avoid ring breaking and enabling user-defined and customized atom matching criteria. In the calculations presented here, a methane-like and a cyclopentane-like CC were used. These consist of a methyl or cyclopentanyl group, respectively, to which a junction LJ particle, denoted as X, is connected, as shown in Figure 1. We computed relative solvation free energies between methane, methanol, ethane, 2-methylfuran, 2-methylindole, neopentane, and toluene to the methane-like CC and for 2-cyclopentanylindole (2-CPI) and 7-cyclopentanylindole (7-CPI) to the cyclopentane-like CC. As described in Section 2, the junction LJ particle serves as the last non-interacting atom connecting the dummy atom region of the molecule with the real region. In both cases, its presence changes a "triple" into a "terminal junction," guaranteeing that the dummy atoms present in the CC states have no influence on the result.[20] Its LJ parameters were set to $\epsilon = -0.15$ kcal/mol and $\sigma = 1.5$ Å, and its partial charge was zero. The bonded parameters involving X and the CC atoms were those of the corresponding hydrogen in methane and cyclopentane, respectively.

3.2.2 | Defining the alchemical path

To generate the alchemical path connecting the physical end-state of two molecules to their CC, at least for one molecule a non-zero number of atoms have to be transformed into dummy atoms. Optionally, CC parameters have to be modified so that the end-state is the same for both transmutations. Transformato always changes a molecule or chemical moiety from the initial physical state to the target CC in the following order. First, we turn off the Coulomb interactions of all atoms not in the CC, including the junction atom X. This step corresponds to linearly scaling a traditional coupling parameter λ from 1 to 0, but Transformato writes explicit charge information (into the PSF file read by both CHARMM and OpenMM for every intermediate step). If the charge manipulation results in a non-integer net charge, a compensating charge of opposite sign is applied to the physical atom (not the junction atom X) that connects the dummy region with the interacting region.

LJ interactions of the atoms not part of the CC are turned off by SAI.[18] In the systems studied here, we scaled the van-der-Waals interactions of all hydrogen to zero in a single step. Next, one or at most two of the remaining heavy atoms were turned off per step. Proceeding in this manner resulted in sufficient overlap with neighboring states.

For the systems studied here, turning off Coulomb and LJ interactions of all atoms not in the CC and if necessary transforming the junction atom X to a LJ particle (cf. above) resulted in CCs which only differed in the bonded parameters involving X. The bonded parameters involving X were those of the original atom. Choosing "methane" (CH_3X) and "cyclopentane-X" (cf. above) as the final end-states (target CCs) as described earlier, one has to change the bonded parameters involving X of all other endpoints; this corresponds to transforming CC_i to CC. In each case, the force field parameters of the bonded terms involved were scaled linearly. In more complex scenarios additional modifications might be necessary to transform the intermediate state CC_i reached after turning off charges and LJ interactions of non CC-atoms into the target CC. For each of the intermediate states parameter and topology information was written out in CHARMM format (PSF/PRM/RTF files) using the ParmEd library.[36]

Figure 2 illustrates the steps just described for the calculation of the free energy difference between toluene and methane; identical steps were used in the gas phase and in aqueous solution. The target CC, CH_3X, is shown at the bottom right of the figure; the junction atom X is colored in red. The steps to transform methane to the CC are shown in the lower half of the figure. The necessary changes in charge are computed in one step (indicated as a single dot in the light blue line), followed by the single change in LJ parameters ($H \rightarrow X$, indicated in light green). By contrast, for toluene most of the benzyl ring needs to be transformed into dummy atoms; the ring atom bound to

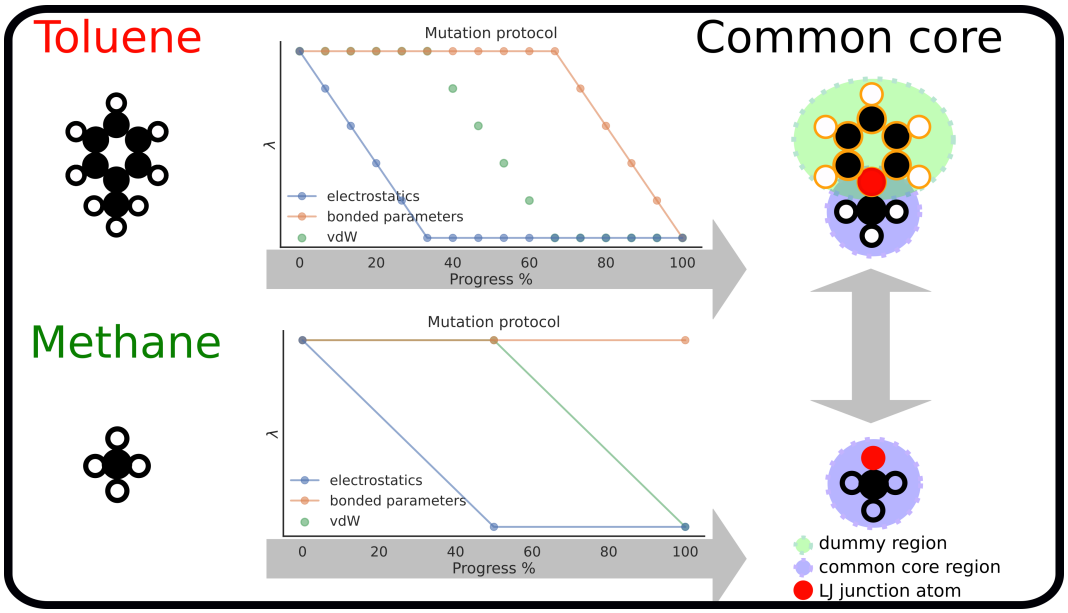

FIGURE 2 Using the SAI/CC approach the physical end-states can be formulated without using dummy atoms. The steps needed to compute the relative solvation free energy between toluene and methane using the CC/SAI approach as realized by Transformato are illustrated. Each of the physical molecules is transformed into a common core, shown on the right. The intermediate steps needed in each case are sketched in the plots in the middle: Changes in electrostatics are indicated in light blue, the transformation of LJ interactions to dummy atoms or into the atom type X are indicated in light green, and common core adjustments involving X (here needed only for toluene) are shown in brown. The same intermediates are used in the gas phase and in aqueous solution. The junction atom X is colored in red. The additional dummy atoms present in the common core obtained starting from toluene (top right) have no effect on the relative solvation free energy difference of interest.[20]

the methyl group becomes the junction atom X. The necessary steps are outlined in the top half of the figure. First, the charges of the benzyl ring are switched off in multiple steps (light blue dots). Then, the LJ interactions of these atoms are turned off in several steps, plus the change C → X is carried out (light green dots). At this point, the bonded parameters involving X are still those of the aromatic carbon. We modify these parameters during additional states (brown dots) into those of a methyl hydrogen interacting with the atoms in the CC. Since the contributions from the dummy atoms present in the CH_3X CC obtained as the end-state of the toluene transformation are identical in the gas phase and aqueous solution, the two CCs shown on the right in Figure 2 are equivalent with respect to the relative solvation free energy difference between toluene and methane, the quantity we want to compute. The detailed number of intermediate states for each of the three stages (turning off electrostatic and vdW interactions, as well as the adjustment of the CC region) and each of the systems is given in Table SI1.

3.2.3 | Sampling of alchemical states

Each alchemical state was sampled using Langevin dynamics[37] at 303.15 K for 2 ns with a 1 fs time step and a friction coefficient of 1/ps; coordinates were saved every 100 steps. Simulations for the solvated system were performed under periodic boundary conditions in a box of TIP3 waters[38,39] with an initial side length of 30 Å in the isothermal–isobaric ensemble at 1 bar. In calculations with CHARMM we used a Langevin piston barostat,[40] for OpenMM a Monte Carlo barostat.[41,42] Waters were kept rigid throughout the simulation utilizing the SHAKE[43] (CHARMM) or the SETTLE[44] (OpenMM) algorithm. In line with the protocols used by Loeffler et al.,[15] the solutes were completely flexible. In the vacuum simulations, no cut-off was applied to the non-bonded interactions. In solution, Coulomb interactions were calculated using the particle-mesh Ewald (PME) method[45] on a 36 × 36 × 36 grid (CHARMM) and a fractional error tolerance of 0.005 (OpenMM). LJ interactions were switched smoothly to zero between 10 and 12 Å. Calculations with CHARMM were carried out both with the "vswitch", as well as the "vfswitch" tapering function for the LJ interactions; in the OpenMM calculations we employed "vfswitch", provided as a custom energy routine from the CHARMM-GUI server[33,34] (cf. Overview of simulations). In addition, we used the switching function implemented in OpenMM ("switch"), as well as a hard truncation of LJ interactions ("no-switch") at 12 Å. No LJ long-range corrections were applied to the simulations.

Starting coordinates for the simulations at each intermediate state were obtained as follows. A NVT equilibration simulation of 125 ps length was carried out for the physical end-state. Before each production simulation of an alchemical state, the coordinates were optimized using the L-BFGS algorithm in OpenMM or the steepest descent and adopted basis Newton–Raphson minimizer in CHARMM. For each state sampling was carried out with OpenMM and CHARMM, using the LJ switching functions as described. Simulations

of each state/condition were repeated five times, using different initial random velocities. A detailed description of the parameters of each system, the mutations, and the input files for each state along the alchemical path can be found in https://github.com/wiederm/Transformato-systems.

3.2.4 | Calculating relative free energy differences

Free energy estimates between each of the solutes shown in Figure 1 and the respective CC in the gas phase and in aqueous solution were calculated using the multi-state Bennett acceptance ratio (MBAR) method as implemented in the pymbar package.[26] Each alchemical state λ_k was simulated for 2 ns. In each simulation trajectories containing 20,000 coordinate sets were saved, of which only every third frame of the final 75% were considered for analysis; that is, for each state 5000 frames were processed by pymbar.

For each configuration sample x, and each alchemical state λ, we computed the reduced potential $u(x,\lambda)$ to form the $N \times K$ matrix of inputs for MBAR, where N is the number of snapshots used and K the number of alchemical states λ_k for a given transformation. Thus, $N = \sum_{k=1}^{K} N_k$, with $N_k = 5000$ snapshots per λ state as just described. To implement this efficiently for CHARMM, a single merged trajectory with all configuration samples x from each alchemical state λ was generated using mdtraj.[46] This facilitated analysis as for each of the five repetitions for each alchemical transformation only a single trajectory had to be post-processed for each of the alchemical states.

Solvation free energy differences between two solutes A, B, and their CC were combined according to Equation 3 to obtain $\Delta\Delta G$ $(A \rightarrow B)$. The individual values for solute A and B used for Equation 3 were obtained using a thermodynamic cycle (i.e., calculating ΔG_{vac} and ΔG_{solv} from the physical end-state to the CC structure as shown in SI Figure SI3). Since calculations were repeated five times the average values for ΔG_{vac} and ΔG_{solv} for solute A and B were used to calculate $\Delta\Delta G(A \rightarrow B)$. The final standard deviation was obtained by Gaussian error propagation.

3.3 | Absolute solvation free energy calculations

Absolute solvation free energies for each of the compounds were computed with the PERT module of CHARMM. Here the soft-core potential implemented in PERT was used.[11] System size, treatment of non-bonded interactions, thermostat, and barostat settings were analogous to the calculations described above. Similarly, each free energy simulation was repeated five times. A total of 21 alchemical states were used for each calculation. At each state, an equilibration phase with 200 ps was followed by 2 ns production phase, during which $\langle \partial U/\partial \lambda \rangle_\lambda$ was evaluated by PERT on the fly. Free energy differences were calculated using thermodynamic integration. The $\langle \partial U/\partial \lambda \rangle_\lambda$ values were fitted using natural cubic splines, which were then integrated analytically. Details, including the calculation of error estimates, can be found in Section 4 of SI of Ref. 20.

4 | RESULTS AND DISCUSSION

All relative solvation free energy differences computed with CHARMM and OpenMM, using various tapering functions for the LJ interactions, as well as reference results are summarized graphically in Figure 3. The raw data from which the plots were generated can be found in Table SI2 of SI. The underlying pymbar framework provides for each individual free energy calculation from physical state to the CC an overlay plot for the alchemical states and an accumulated free energy plot for the transformation; an example is shown in Figure SI1 of SI.

To validate the CC/SAI approach using Transformato we first compared to results obtained with the established PERT module implemented in CHARMM with identical end-state definition and parameter set. In particular, we compare the relative solvation free energies using the CHARMM back end of Transformato utilizing the "vswitch" switching function (shown in Figure 3 in blue) to the $\Delta\Delta G$ values generated with PERT/CHARMM/vswitch (shown in red) as the difference of the two absolute solvation free energies. The two approaches give $\Delta\Delta G$ estimates for all systems which agree well within statistical error estimates, demonstrating the correctness of the CC/SAI approach as implemented in Transformato.

Next, we tested whether the two MD engines currently supported by Transformato, CHARMM and OpenMM, led to results that do agree within statistical error bars. Here we employed the "vfswitch" LJ switching function available in both MD engines. These results are shown in Figure 3 in purple (TF/CHARMM/vfswitch) and green (TF/OpenMM/vfswitch). Again, the relative solvation free energy differences agree within their standard deviation, with a single exception, the free energy difference between methanol and methane. However, for this transformation the net deviation is very small (0.11 kcal/mol) and the standard deviation extremely low (see Table SI2). For completeness we also report the relative solvation free energy differences obtained using the native switching function in OpenMM (TF/OpenMM/switch) (shown in orange in Figure 3) and with a hard cut-off (TF/OpenMM/no-switch) (shown in brown in Figure 3).

Overall, there is surprisingly little variability in the $\Delta\Delta G$ estimates obtained with different switching functions. All relative solvation free energy differences for the pairs ethane/methanol/neopentane/toluene/2-methylfuran to methane and 2-CPI to 7-CPI lie within the ±0.25 kcal/mol interval around the average $\Delta\Delta G$ values (for each system the total average is the average overall values generated with the six different approaches). Only for 2-methylindole the average values of the five runs reach outside this ±0.25 kcal/mol interval. This agreement may in part be the result of fortunate error compensation arising from the use of thermodynamic cycles, as previously observed; for example, Refs. 47, 48 Differences between results obtained with the

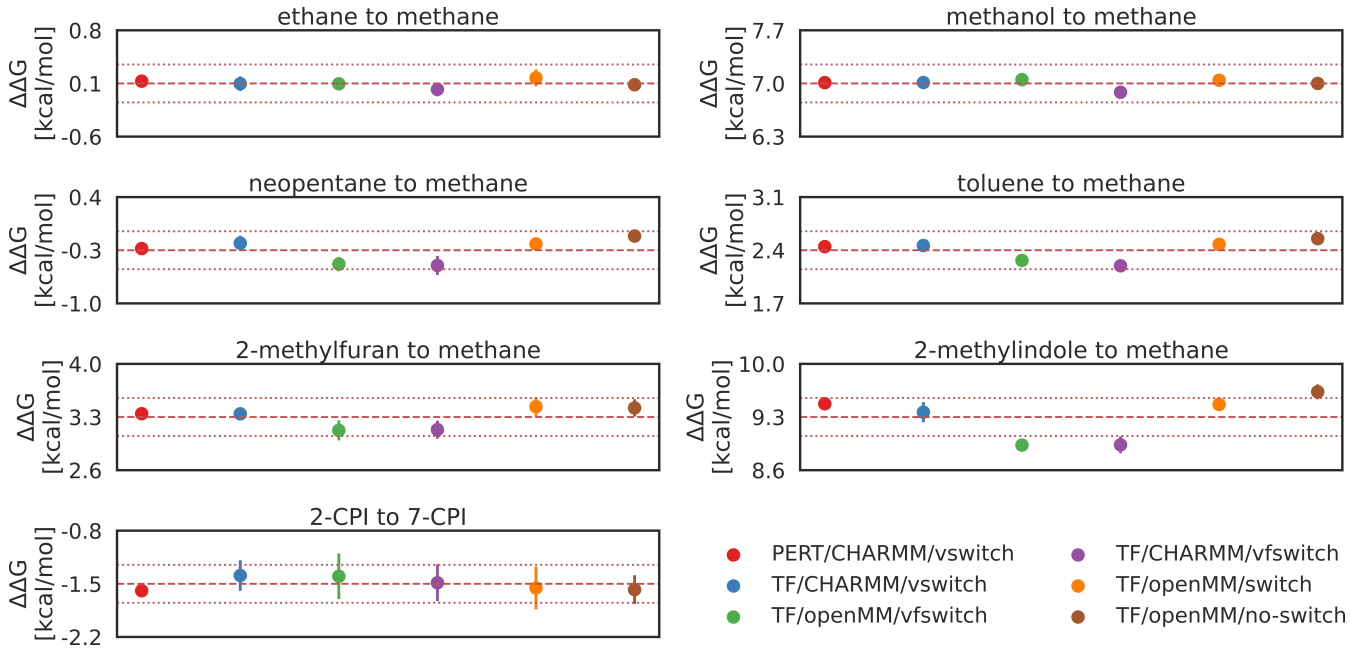

FIGURE 3 Comparing the ΔΔG values for six different approaches described in the methods section show good agreement on the investigated systems. Each of the free energy calculations was repeated five times and the average of the obtained ΔΔG estimates and its standard deviation is plotted. The dashed red line indicates the total average of the six described approaches and the thin, dotted red lines mark the ±0.25 kcal/mol interval around the average. Results generated with Transformato (abbreviated with TF in the legend) and OpenMM used either the OpenMM native switching function (TF/OpenMM/switch), the implementation of the "vfswitch" function (TF/OpenMM/vfswitch) or no-switching function (TF/OpenMM/no-switch), results generated with CHARMM used the "vswitch" (TF/CHARMM/vswitch) or "vfswitch" (TF/CHARMM/vfswitch). In addition to the alchemical path generated using Transformato we also calculated absolute solvation free energies with the PERT module of CHARMM (PERT/CHARMM/vswitch).

various switching methods might be more pronounced for absolute free energies.

While all results for 2-CPI to 7-CPI lie within the ±0.25 kcal/mol interval about the average, their standard deviations are relatively high. Since this is the most complex transformation, this is not completely unexpected. However, compared to the analogous mutation from 2-methylindole to the methane-like CC, we used fewer steps to turn off vdW interactions (see Table SI1 in SI). When repeating the simulations using vfswitch to truncate LJ interactions with the protocol used for 2-methylindole (10 instead of 7 steps to turn off vdW interactions), the overall standard deviation was reduced from 0.24 to 0.17 kcal/mol (CHARMM/vfswitch) and from 0.30 to 0.18 kcal/mol (OpenMM/vfswitch, see Table SI2).

Since different force fields were used, we cannot compare our results directly to those of Loeffler et al.[15] However, it is of interest to take a look at the variability of the free energy estimates obtained with the different software programs used in Ref. 15 and the two supported back ends and different treatments of LJ interactions used in this work. In Figure SI2, we show density plots for the distribution of the relative solvation free energy estimates from Loeffler et al.[15] and our results reported in Figure 3. The offset in the average values is a direct consequence of the force fields used. The variability of the results, reflected by the widths of the density distributions, on the other hand, is quite similar.

5 | CONCLUSION

We presented a Python package called Transformato that is able to generate semi-automatically the alchemical path(s) connecting two or more molecules in a given environment. The results of this work were obtained with an early version of Transformato and serve as the proof of concept of the CC/SAI approach. Transformato is developed as an open source project, see code and data availability below. We validated our methodology and its implementation by using the benchmark set of seven different mutations also used by Loeffler et al.[15] Within statistical error bars we obtained relative solvation free energies that agreed excellently with results of reference calculations using the PERT module of CHARMM.

Using CC/SAI, that is, Transformato, the end-states are the true physical molecules without dummy atoms. In traditional single topology setups of alchemical transformations the correct treatment of dummy atom parameters is not trivial. Systematic errors resulting from non-redundant bonded parameters are a possibility when naively keeping all bonded parameters for the dummy region.[20] Similarly, the CC/SAI approach avoids the need for hybrid topologies at the end-states when two chemical moieties are present at the same time, as is typically the case in certain forms of dual topology setups.

The combination of the CC approach with SAI makes it possible to use, in principle, any biomolecular MD program as the back end for Transformato. CHARMM and OpenMM are the most frequently used

programs in our groups, and adapting input generated by the CHARMM-GUI web server[33,34] is straightforward. At present, Transformato is not tied to CHARMM-GUI's free energy calculator.[49] Currently, Transformato writes inputs for the intermediate states in CHARMM format, in particular the PSF and parameter files, though adding the capability to write other formats would be straightforward. Programs, which have support for CHARMM file formats, such as NAMD, could be supported by Transformato easily. The CC/SAI approach in general and Transformato in particular should not be viewed as a front end to dedicated programs to compute free energy differences, but as a tool to carry out FES with almost any MD program. We do neither require nor use any alchemical FES related functionality of the underlying program. Since only minor modifications to input files for supported MD programs, rather than changes at the code level are needed, extending Transformato's functionality to, for example, the calculation of relative binding free energies, both for globular proteins, as well as membrane proteins is straightforward. Work in this direction is currently ongoing.

ACKNOWLEDGMENTS

Marcus Wieder is grateful for the support of Thierry Langer, who donated significant computational resources to perform much of the calculations/simulations. Marcus Wieder acknowledges support from a FWF Erwin Schrödinger Postdoctoral Fellowship J 4245-N28. Stefan Boresch acknowledges grant P-31024 of the Austrian Science Foundation (FWF), supporting (in part) Markus Fleck and Benedict Braunsfeld.

ENDNOTES

[1] https://github.com/choderalab/openmmtools.

[2] https://github.com/choderalab/perses.

[3] https://academiccharmm.org/documentation/version/c46b1/domdec.

DATA AVAILABILITY STATEMENT

Python package used in this work (release v0.1): https://github.com/wiederm/transformato. Data and notebooks to reproduce the plots/figures (release v0.1): https://github.com/wiederm/transformato-systems.

ORCID

Marcus Wieder https://orcid.org/0000-0003-2631-8415
Markus Fleck https://orcid.org/0000-0002-8648-2164
Benedict Braunsfeld https://orcid.org/0000-0002-0286-8239
Stefan Boresch https://orcid.org/0000-0002-2793-6656

REFERENCES

[1] N. Hansen, W. F. van Gunsteren, *J. Chem. Theory Comput.* **2014**, *10*, 2632.
[2] L. Wang, Y. Wu, Y. Deng, B. Kim, L. Pierce, G. Krilov, D. Lupyan, S. Robinson, M. K. Dahlgren, J. Greenwood, D. L. Romero, C. Masse, J. L. Knight, T. Steinbrecher, T. Beuming, W. Damm, E. Harder, W. Sherman, M. Brewer, R. Wester, M. Murcko, L. Frye, R. Farid, T. Lin, D. L. Mobley, W. L. Jorgensen, B. J. Berne, R. A. Friesner, R. Abel, *J. Am. Chem. Soc.* **2015**, *137*, 2695.
[3] D. L. Mobley, M. K. Gilson, *Annu. Rev. Biophys.* **2017**, *46*, 531.
[4] H.-C. Tsai, Y. Tao, T.-S. Lee, K. M. Merz, D. M. York, *J. Chem. Inf. Model.* **2020**, *60*, 5296.
[5] T. S. Lee, B. K. Allen, T. J. Giese, Z. Guo, P. Li, C. Lin, T. Dwight McGee, D. A. Pearlman, B. K. Radak, Y. Tao, H. C. Tsai, H. Xu, W. Sherman, D. M. York, *J. Chem. Inf. Model.* **2020**, *60*, 5595.
[6] R. O. Dror, R. M. Dirks, J. P. Grossman, H. Xu, D. E. Shaw, *Annu. Rev. Biophys.* **2012**, *41*, 429.
[7] T. J. Giese, D. M. York, *J. Chem. Theory Comput.* **2018**, *14*, 1564.
[8] H. Chen, J. D. C. Maia, B. K. Radak, D. J. Hardy, W. Cai, C. Chipot, E. Tajkhorshid, *J. Chem. Inf. Model.* **2020**, *60*, 5301.
[9] H. H. Loeffler, J. Michel, C. Woods, *J. Chem. Inf. Model.* **2015**, *55*, 2485.
[10] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, R. J. Woods, *J. Comput. Chem.* **2005**, *26*, 1668.
[11] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, M. Karplus, *J. Comput. Chem.* **2009**, *30*, 1545.
[12] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess, E. G. R. O. M. A. C. S. Lindahl, *SoftwareX* **2015**, *1-2*, 19.
[13] J. C. Phillips, D. J. Hardy, J. D. Maia, J. E. Stone, J. V. Ribeiro, R. C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang, R. McGreevy, M. C. Melo, B. K. Radak, R. D. Skeel, A. Singharoy, Y. Wang, B. Roux, A. Aksimentiev, Z. Luthey-Schulten, L. V. Kalé, K. Schulten, C. Chipot, E. Tajkhorshid, *J. Chem. Phys.* **2020**, *153*, 1.
[14] D. A. Pearlman, *J. Phys. Chem.* **1994**, *98*, 1487.
[15] H. H. Loeffler, S. Bosisio, G. D. R. Matos, D. Suh, B. Roux, D. L. Mobley, J. Michel, *J. Chem. Theory Comput.* **2018**, *14*, 5567.
[16] A. Rizzi, T. Jensen, D. R. Slochower, M. Aldeghi, V. Gapsys, D. Ntekoumes, S. Bosisio, M. Papadourakis, N. M. Henriksen, B. L. de Groot, Z. Cournia, A. Dickson, J. Michel, M. K. Gilson, M. R. Shirts, D. L. Mobley, J. D. Chodera, *J. Comput. Aided Mol. Des.* **2020**, *34*, 601.
[17] Woods, C.; Mey, A.; Calabró, G.; Michel, J. Sire molecular simulations framework. **2018**; https://www.siremol.org/.
[18] S. Boresch, S. Bruckner, *J. Comput. Chem.* **2011**, *32*, 2449.
[19] D. C. B. Siebert, M. Wieder, L. Schlener, P. Scholze, S. Boresch, T. Langer, M. Schnürch, M. D. Mihovilovic, L. Richter, M. Ernst, G. F. Ecker, *J. Chem. Inf. Model.* **2018**, *58*, 1682.
[20] M. Fleck, M. Wieder, S. Boresch, *J. Chem. Theory Comput.* **2021**, *17*, 4403.
[21] T. Simonson, *Mol. Phys.* **1993**, *80*, 441.
[22] T. C. Beutler, A. E. Mark, R. C. van Schaik, P. R. Gerber, W. F. van Gunsteren, *Chem. Phys. Lett.* **1994**, *222*, 529.
[23] M. Zacharias, T. P. Straatsma, J. A. McCammon, *J. Chem. Phys.* **1994**, *100*, 9025.
[24] T.-S. Lee, Z. Lin, B. K. Allen, C. Lin, B. K. Radak, Y. Tao, H.-C. Tsai, W. Sherman, D. M. York, *J. Chem. Theory Comput.* **2020**, *16*, 5512.
[25] C. H. Bennett, *J. Comput. Phys.* **1976**, *22*, 245.
[26] M. R. Shirts, J. D. Chodera, *J. Chem. Phys.* **2008**, *129*, 124105.
[27] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, V. S. Pande, *PLoS Comput. Biol.* **2017**, *13*, e1005659.
[28] K. Vanommeslaeghe, E. Hatcher, C. Acharya, S. Kundu, S. Zhong, J. Shim, E. Darian, O. Guvench, P. Lopes, I. Vorobyov, A. D. Mackerell, *J. Comput. Chem.* **2010**, *31*, 671.
[29] K. Vanommeslaeghe, A. D. MacKerell, J., *J. Chem. Inf. Model.* **2012**, *52*, 3144.

[30] K. Vanommeslaeghe, E. P. Raman, A. D. MacKerell, *J. Chem. Inf. Model.* **2012**, *52*, 3155.
[31] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, M. Karplus, *J. Comput. Chem.* **1983**, *4*, 187.
[32] P. J. Steinbach, B. R. Brooks, *J. Comput. Chem.* **1994**, *15*, 667.
[33] J. Lee, X. Cheng, J. M. Swails, M. S. Yeom, P. K. Eastman, J. A. Lemkul, S. Wei, J. Buckner, J. C. Jeong, Y. Qi, S. Jo, V. S. Pande, D. A. Case, C. L. Brooks, A. D. MacKerell, J. B. Klauda, W. Im, *J. Chem. Theory Comput.* **2016**, *12*, 405.
[34] S. Jo, T. Kim, V. G. Iyer, W. Im, *J. Comput. Chem.* **2008**, *29*, 1859.
[35] Landrum, G. RDkit: Open-source Cheminformatics. http://www.rdkit.org.
[36] M. R. Shirts, C. Klein, J. M. Swails, J. Yin, M. K. Gilson, D. L. Mobley, D. A. Case, E. D. Zhong, *J. Comput. Aided Mol. Des.* **2017**, *31*, 147.
[37] D. S. Lemons, A. P. Gythiel, *Am. J. Physiol.* **1997**, *65*, 1079.
[38] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, M. L. Klein, *J. Chem. Phys.* **1983**, *79*, 926.
[39] E. Neria, S. Fischer, M. Karplus, *J. Chem. Phys.* **1996**, *105*, 1902.
[40] S. E. Feller, Y. Zhang, R. W. Pastor, B. R. Brooks, *J. Chem. Phys.* **1995**, *103*, 4613.
[41] K.-H. Chow, D. M. Ferguson, *Comput. Phys. Commun.* **1995**, *91*, 283.
[42] J. Åqvist, P. Wennerström, M. Nervall, S. Bjelic, B. O. Brandsdal, *Chem. Phys. Lett.* **2004**, *384*, 288.
[43] J.-P. Ryckaert, G. Ciccotti, H. J. Berendsen, *J. Comput. Phys.* **1977**, *23*, 327.
[44] S. Miyamoto, P. A. Kollman, *J. Comput. Chem.* **1992**, *13*, 952.
[45] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, L. G. Pedersen, *J. Chem. Phys.* **1995**, *103*, 8577.
[46] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane, V. S. Pande, *Biophys. J.* **2015**, *109*, 1528.
[47] D. L. Mobley, P. V. Klimovich, *J. Chem. Phys.* **2012**, *137*, 230901.
[48] S. Schlund, E. M. Basílio Janke, K. Weisz, B. Engels, *J. Comput. Chem.* **2009**, *30*, 2967.
[49] H. Zhang, S. Kim, T. J. Giese, T.-S. Lee, J. Lee, D. M. York, W. Im, *J. Chem. Inf. Model.* **2021**, *61*, 4145.

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of the article at the publisher's website.

How to cite this article: M. Wieder, M. Fleck, B. Braunsfeld, S. Boresch, *J. Comput. Chem.* **2022**, *43*(17), 1151. https://doi.org/10.1002/jcc.26877